ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ασύρματη Κατανεμημένη Υλοποίηση Νευρο-Ασαφούς Συστήματος Ταξινόμησης (Wireless Distributed Implementation of Fuzzy Neural Classification System)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΩΣΤΑΣ Α. ΤΣΙΩΛΗΣ

ΧΡΗΣΤΟΣ Ι. ΦΕΡΛΕΣ

Επιβλέπων : Ανδρέας - Γεώργιος Ν. Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2003

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ασύρματη Κατανεμημένη Υλοποίηση Νευρο-Ασαφούς Συστήματος Ταξινόμησης (Wireless Distributed Implementation of Fuzzy Neural Classification System)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΩΣΤΑΣ Α. ΤΣΙΩΛΗΣ

ΧΡΗΣΤΟΣ Ι. ΦΕΡΛΕΣ

Επιβλέπων : Ανδρέας - Γεώργιος Ν. Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 6η Νοεμβρίου 2003.

.....	.....	.....
Ανδρέας-Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π.	Στέφανος Κόλλιας Καθηγητής Ε.Μ.Π.	Παναγιώτης Τσανάκας Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2003

.....
ΚΩΣΤΑΣ Α. ΤΣΙΩΛΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

.....
ΧΡΗΣΤΟΣ Ι. ΦΕΡΛΕΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Αντικείμενο της διπλωματικής εργασίας είναι η μελέτη και η ανάλυση των αρχιτεκτονικών και των αλγορίθμων που σχετίζονται με τα νευρο-ασαφή συστήματα ταξινόμησης. Ειδικότερα προτείνεται μια ασύρματη κατανεμημένη λειτουργία ενός συστήματος ταξινόμησης-κατηγοριοποίησης υπό πραγματικές συνθήκες. Απώτερος σκοπός είναι η δημιουργία μιας φορητής συσκευής η οποία θα έχει την δυνατότητα να λαμβάνει πληροφορίες από έναν ασθενή, με αισθητήρες που θα φέρει επάνω του, και θα μπορεί να βγάλει ιατρικής φύσεως συμπεράσματα περί της κατάστασης του. Παράλληλα υποστηρίζεται η δυνατότητα ανανέωσης-επέκτασης των δυνατοτήτων εξαγωγής συμπερασμάτων από την πλευρά της συσκευής.

Πιο συγκεκριμένα το Κεφάλαιο 1 περιλαμβάνει εισαγωγικά στοιχεία που αφορούν στο σύστημα που κατασκευάστηκε. Στο Κεφάλαιο 2 εισάγονται οι έννοιες των ασαφών συνόλων, των νευρωνικών δικτύων και των νευρο-ασαφών συστημάτων. Στο Κεφάλαιο 3 αναφέρονται στοιχεία σχετικά με τα δίκτυα ταξινόμησης, τον αλγόριθμο ανάστροφης διάδοσης με την μέθοδο κλίσης και τα κριτήρια αξιοπιστίας. Στο κεφάλαιο 4 γίνεται ανάλυση του νευρο-ασαφούς συστήματος SuPFuNIS, ενώ παράλληλα αναφέρονται ορισμένες επεκτάσεις τροποποιήσεις που εφαρμόστηκαν σε αυτό. Το Κεφάλαιο 5 αφορά τα ασύρματα δίκτυα και το πρωτόκολλο IEEE 802.11, στοιχεία βάσει των οποίων πραγματοποιήθηκε η ασύρματη λειτουργία του όλου συστήματος. Το Κεφάλαιο 6 παρουσιάζει την πλατφόρμα Java, η οποία αποτελεί την γλώσσα προγραμματισμού που χρησιμοποιήθηκε. Αρχικά στο Κεφάλαιο 7 προτείνεται ένας αλγόριθμός υβριδικής μάθησης, ενώ επίσης προτείνεται η αρχιτεκτονική ενός πολυεπίπεδου ταξινομητή, στην συνέχεια περιγράφεται αναλυτικά η υλοποίηση που πραγματοποιήθηκε σε Java. Τέλος στο Κεφάλαιο 8 εκθέτονται τα συμπεράσματα από τις πειραματικές προσομοιώσεις που έγιναν με σκοπό να επαληθευτεί η επίδοση των χρησιμοποιούμενων μεθόδων, αλγορίθμων και αρχιτεκτονικών.

Λέξεις Κλειδιά

ασαφής συλλογιστική, νευρωνικά δίκτυα, νευρο-ασαφή συστήματα, δίκτυα ταξινόμησης, πολυεπίπεδος ταξινομητής, αλγόριθμος ανάστροφης διάδοσης, αλγόριθμος υβριδικής μάθησης, κριτήρια αξιοπιστίας, SuPFuNIS, ασύρματα τοπικά δίκτυα, IEEE 802.11, J2ME, Personal Profile, PDA

Abstract

The purpose of the present diploma thesis is the study and analysis of the algorithms and architectures that are employed on fuzzy-neural classification systems. A wireless distributed implementation of a classification system is also proposed. The main purpose is programming a mobile device so as to collect data from sensors and simultaneously inferring diagnoses relevant to the condition of a patient. Furthermore supervised and hybrid learning are employed in order to lend adaptability to the classification model.

More precisely, Chapter 1 is an introduction to the major elements of the implemented system. Chapter 2 introduces fuzzy logic, neural networks and fuzzy-neural systems. Chapter 3 contains information on classification systems, the backpropagation algorithm and information relevant to classification reliability measures. In Chapter 4 the SuPFuNIS model is analyzed and a modification on its core learning algorithm is introduced. In Chapter 5 wireless local area networks and the IEEE 802.11 protocol are explained. Chapter 6 presents the Java platform which is the programming language being used. A hybrid learning algorithm and a multi expert system are introduced in Chapter 7. Also in this chapter the software implementation of the proposed model is analyzed in detail. Finally, the results of simulation experiments are presented in Chapter 8 while the performance of certain algorithms and methods is examined at length.

Key Words

fuzzy logic, neural networks, fuzzy-neural systems, classification systems, multi expert systems, backpropagation, hybrid learning, reliability, SuPFuNIS, WLAN, IEEE 802.11, J2ME, Personal Profile, PDA

Ευχαριστίες

Για την εκπόνηση της παρούσας διπλωματικής εργασίας θα θέλαμε να ευχαριστήσουμε κατά κύριο λόγο τον καθηγητή μας κ. Α. Γ. Σταφυλοπάτη, που με την αμέριστη συμπαράσταση τις πάντοτε καίριες και χρήσιμες συμβουλές του υπήρξε ο καθοδηγητής μας στη μελέτη αυτή. Επίσης πρέπει να τονίσουμε ότι η εργασία αυτή δεν θα είχε έλθει εις πέρας χωρίς την πολύτιμη συμβολή και τις παρεμβάσεις των υποψηφίων διδακτόρων Μηνά Περτσελάκη και Δημήτρη Φροσυνιώτη, οι οποίοι μας βοήθησαν σημαντικά στο να ξεπεράσουμε τα σημαντικά προβλήματα που προέκυψαν.

Πίνακας Περιεχομένων

Κεφάλαιο 1

Εισαγωγή

1 Εισαγωγή

Το αρχικό πρόβλημα το οποίο αποτέλεσε την πηγή έμπνευσης για την ανάπτυξη της παρούσας διπλωματικής εργασίας ήταν η δημιουργία μιας φορητής συσκευής η οποία να μπορεί να δέχεται πληροφορίες από αισθητήρες που βρίσκονται επάνω στο σώμα ενός ασθενούς και να είναι σε θέση να εξάγει συμπεράσματα για την ιατρική κατάσταση του.

Γίνεται επομένως αντιληπτό το γεγονός ότι τα δεδομένα που αποστέλλουν οι αισθητήρες προκύπτουν από βιοσήματα (όπως χτύποι καρδιάς, αρτηριακή πίεση), τα οποία επειδή αναφέρονται σε χαρακτηριστικά του ανθρώπινου οργανισμού εξαρτώνται από διάφορους παράγοντες όπως η ηλικία, το φύλο, ή η φυσική του κατάσταση. Πρέπει λοιπόν να χρησιμοποιηθεί η ασαφής συλλογιστική για να περιγράψει τέτοιες συγκεχυμένες έννοιες όπως ΝΕΟΣ ή ΑΣΘΕΝΗΣ. Από την άλλη πλευρά ενδείκνυται η χρήση των νευρωνικών δικτύων ούτως ώστε να γίνει χρήση της προσαρμοστικής ικανότητας τους σε άγνωστα εν γένει προβλήματα. Σημαντική επίσης είναι η επίτευξη διάγνωσης περί της κατάστασης του ασθενούς, επομένως είναι λογικό να αναφερόμαστε σε πρόβλημα ταξινόμησης εφόσον ένας ασθενής πρέπει να ταξινομηθεί σε μια κατηγορία (όπως για παράδειγμα ΥΓΙΗΣ ή ΑΣΘΕΝΗΣ). Τέλος είναι προφανής η αναγκαιότητα λήψης αποφάσεων σε σύντομο χρονικό διάστημα, αφού σε αντίθετη περίπτωση χάνει την ουσία της η οποιαδήποτε διάγνωση.

Η μελέτη των παραπάνω παραγόντων οδήγησε στο σχεδιασμό και στην υλοποίηση του **Wireless Distributed Implementation of Fuzzy Neural Classification System** (*WiDIFuNeCS*).

Το αντικείμενο της παρούσας διπλωματικής εργασίας λοιπόν εστιάζεται στα νευρο-ασαφή συστήματα ταξινόμησης. Μελετώντας και αναλύοντας ένα συγκεκριμένο νευρο-ασαφές σύστημα, το SuPFuNIS, επιχειρείται μια ασύρματη κατανεμημένη υλοποίηση αυτού σε πραγματικό περιβάλλον. Με τον όρο κατανεμημένη λειτουργία τονίζεται ο δομικός αλλά και χωροταξικός διαχωρισμός ανάμεσα στη διαδικασία συλλογής δεδομένων εισόδου, στη διαδικασία ταξινόμησης και στην διαδικασία εκπαίδευσης του νευρο-ασαφούς δικτύου ταξινόμησης. Ο διαχωρισμός έγκειται στο ότι οι διαδικασίες αυτές πραγματοποιούνται σε διαφορετικές συσκευές. Με τον όρο ασύρματη, περιγράφεται η λειτουργία κατά την οποία τα δεδομένα εισόδου συλλέγονται με ασύρματο τρόπο ενώ και η αρχικοποίηση-ανανέωση του διανύσματος των βαρών γίνονται επίσης ασύρματα. Τέλος η έννοια πραγματικό περιβάλλον περιγράφει τις πραγματικές συνθήκες που πρέπει ένα τέτοιο σύστημα να είναι σε θέση να αντιμετωπίσει όπως είναι ο θόρυβος ή οι παρεμβολές.

Το SuPFuNIS μοντέλο πάνω στο οποίο δομήθηκε το WiDIFuNeCS, διαφέρει από τα άλλα νεύρο-ασαφή δίκτυα σε διάφορα σημεία:

- Χρησιμοποιεί έναν εκπαιδευόμενο ασαφοποιητή εισόδου που είναι υπεύθυνος για την ασαφοποίηση των αριθμητικών δεδομένων.
- Όλες οι πληροφορίες που διαδίδονται από το στρώμα εισόδου είναι ασαφείς.
- Το μοντέλο συσσωρεύει τις ενεργοποιήσεις ενός κόμβου χρησιμοποιώντας το ασαφές εσωτερικό γινόμενο, ένα γινόμενο αμοιβαίων υποσυνόλων.
- Οι έξοδοι προκύπτουν με την χρήση της μεθόδου ποσοτικής απο-ασαφοποίησης πράγμα που αποτελεί παραλλαγή της συνηθισμένης τροποποιημένης μεθόδου κέντρου βάρους.

Συνολικά μπορεί κανείς να παρατηρήσει ότι τα στοιχεία-πεδία που συνιστούν το περιβάλλον στο οποίο κινείται η υλοποίηση του WiDIFuNeCS είναι τα κάτωθι:

- ο αλγόριθμος εξαγωγής συμπερασμάτων και η διαδικασία βελτίωσης-επέκτασης των δυνατοτήτων αυτού
- οι ασύρματες διασυνδέσεις του συνόλου των επιμέρους συσκευών
- οι λειτουργίες των αισθητήρων
- οι λειτουργίες της φορητής συσκευής
- οι λειτουργίες της συσκευής εξυπηρετητή

•] Βάσει των όσων αναφέρθηκαν προηγούμενα το SuPFuNIS χρησιμοποιείται ως ταξινομητής, δηλαδή ως σύστημα εξαγωγής συμπερασμάτων σχετικών με την κατηγορία στην οποία ανήκει το εκάστοτε πρότυπο εισόδου. Για την αρχικοποίηση του ταξινομητή, ώστε να είναι σε θέση να κατηγοριοποιεί πρότυπα που ανήκουν σε καινούργια προβλήματα, χρησιμοποιείται ο αλγόριθμος backpropagation με την μέθοδο κλίσης όπως αυτή ορίζεται για το SuPFuNIS. Ακόμη για την επανεκπαίδευση-βελτίωση της ικανότητας ταξινόμησης του συστήματος χρησιμοποιείται ένας πρότυπος υβριδικός αλγόριθμος μάθησης.

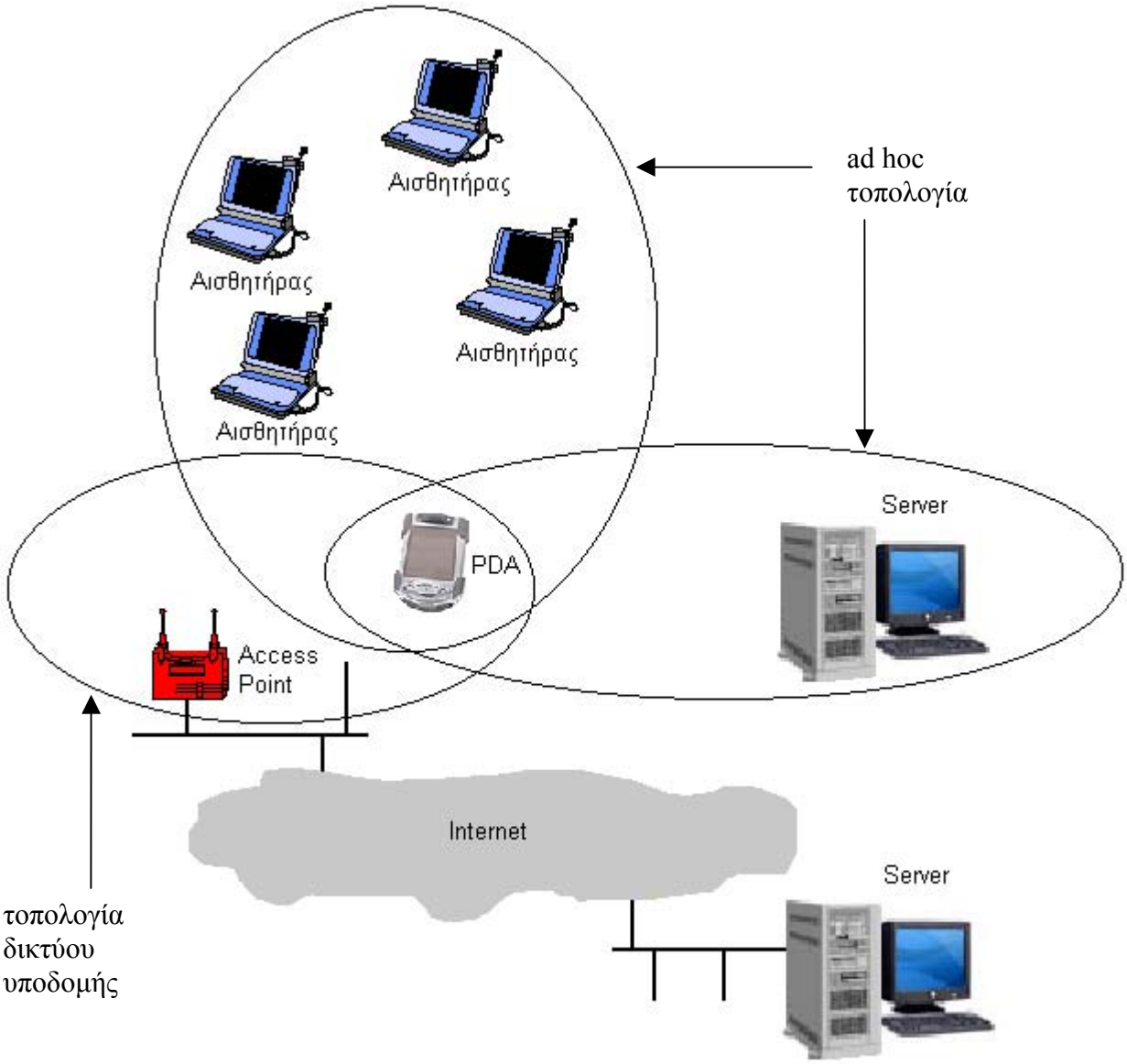
•] Η συλλογή προτύπων εισόδου από τους αισθητήρες για λογαριασμό της φορητής συσκευής γίνεται ασύρματα κάνοντας χρήση ad hoc τοπολογίας. Η επικοινωνία της φορητής συσκευής με τον server ούτως ώστε να γίνει δυνατή η εκπαίδευση-επανεκπαίδευση του συστήματος ταξινόμησης γίνεται ασύρματα με χρήση του πρωτοκόλλου IEEE 802.11 για WLAN. Σε αυτή την περίπτωση όμως η τοπολογία διασύνδεσης των συσκευών μπορεί να είναι ad hoc τοπολογία ή τοπολογία δικτύου υποδομής.

•] Αισθητήρας καλείται μια μικρή συσκευή που έχει την δυνατότητα να λαμβάνει σήματα διαφόρων ειδών και να παράγει αριθμητικές τιμές σχετικές με τα σήματα αυτά.

•] Η φορητή συσκευή αποτελεί τον κρίκο που προσδίδει συνεκτικότητα στην υλοποίηση WiDIFuNeCS. Περιλαμβάνει ένα γραφικό περιβάλλον που επιτρέπει στον χρήστη να επιλέξει μεταξύ των λειτουργιών της ταξινόμησης και της μάθησης του WiDIFuNeCS. Κατά την διάρκεια της λειτουργίας της ταξινόμησης η φορητή συσκευή εξάγει συμπεράσματα σε πραγματικό χρόνο ενώ παράλληλα συλλέγει διανύσματα εισόδου με σκοπό να τα χρησιμοποιήσει στην φάση της επανεκπαίδευσης του συστήματος. Κατά την διάρκεια της λειτουργίας μάθησης παρέχονται δυο δυνατότητες. Η πρώτη είναι να ζητηθεί από τον server ένα σύνολο διανυσμάτων βαρών που να αντιστοιχούν σε ένα εντελώς καινούργιο πρόβλημα ταξινόμησης (αρχικοποίηση). Ενώ η δεύτερη είναι να αποσταλούν τα διανύσματα εισόδων (που έχουν συλλεχθεί) στον server (για περαιτέρω επεξεργασία) και να ανακτηθεί το μέχρι εκείνη την στιγμή βέλτιστο σύνολο βαρών που διαθέτει ο server επί του συγκεκριμένου προβλήματος ταξινόμησης (ανανέωση).

•] Κατά την φάση της εκκίνησης του εξυπηρετητή, ανατίθεται σε αυτόν ένα δεδομένο πρόβλημα ταξινόμησης και στην συνέχεια με παραμετροποιημένη επιβλεπόμενη μάθηση καθορίζονται ορισμένα αρχικά σύνολα βαρών για τον ταξινομητή. Κατά την φάση της κανονικής λειτουργίας ο server δέχεται και επεξεργάζεται αιτήσεις από φορητές συσκευές. Εάν η ληφθείσα αίτηση είναι αίτημα αρχικοποίησης τότε αποστέλλονται στην συσκευή πλήρη διανύσματα βαρών καθώς και ορισμένες πληροφορίες περί του νέου προβλήματος ταξινόμησης που συνδέονται με αυτά τα βάρη. Εάν η αίτηση είναι αίτημα επανεκπαίδευσης τότε αποστέλλονται στην φορητή συσκευή τα μέχρι εκείνη την στιγμή βέλτιστα διανύσματα βαρών, ακολούθως αφού

συλλεχθούν τα πρότυπα εισόδου (τα οποία έχουν σταλεί από την φορητή συσκευή) εκκινείται αυτόματα μια υβριδική διαδικασία επανεκπαίδευσης.



Σχήμα 1.1: Η ασύρματη κατανεμημένη λειτουργία του WiDIFuNeCS.

Κεφάλαιο 2

Νευρο-Ασαφή Συστήματα

2 Νευρο-Ασαφή Συστήματα

2.1 Ασαφή Σύνολα

2.1.1 Εισαγωγή

Η έννοια «σύνολο» κατέχει κεντρική θέση στην επιστήμη των Μαθηματικών και αποτελεί θεμελιώδη σύλληψη πάνω στην οποία δομείται το οικοδόμημα όλων των θετικών επιστημών. Όπως ακριβώς δεν μπορεί κανείς να ορίσει με αυστηρό τρόπο τι είναι ακριβώς ευθεία ή αριθμός, το ίδιο δεν μπορεί να ορίσει αυστηρά και τι είναι σύνολο. Ο άνθρωπος μαθαίνει και κατανοεί τις έννοιες αυτές όχι μόνο μέσω ορισμών αλλά κυρίως μέσω παραδειγμάτων. Έτσι χρειαζόμαστε να επιστρατεύσουμε αυτή ακριβώς τη δυνατότητα για να σκιαγραφήσουμε και να αντιληφθούμε την έννοια του συνόλου. Έστω λοιπόν, σύνολο Χ και χ τυχόν στοιχείο. Τότε είναι προφανές (κατά πόσο θα φανεί στην συνέχεια) ότι μόνο ένα από τα παρακάτω μπορεί να ισχύει:

Το χ ανήκει στο Χ ή το χ δεν ανήκει στο Χ.

Συμβολικά
$\chi \in X \;\; ή \;\; \chi \notin X.$

Τα μαθηματικά στηρίζονται κατά βάση στη συνολοθεωρία και αυτή στηρίζεται εξ' ολοκλήρου σε ένα αξίωμα βασισμένο στη διχοτομία (ανήκει ή δεν ανήκει, είναι εντός ή εκτός του συνόλου). Αμφισβητώντας τη διχοτομία, η κλασική συνολοθεωρία καταρρέει εκ θεμελίων και στη θέση της αναδύεται μια άλλη προσέγγιση, η θεωρία των ασαφών συνόλων (theory of fuzzy sets) [1].

2.1.2 Βασικοί Ορισμοί και Ορολογία

Ένα **κλασικό σύνολο** είναι ένα σύνολο με σαφή όρια και περιγράφεται με δύο τρόπους:

Α. Απαριθμώντας τα στοιχεία του, αν το πλήθος είναι πεπερασμένο

$$E=\{\alpha_1,\alpha_2,\ldots,\alpha_v\}$$

Β. Περιγράφοντας μια κοινή ιδιότητα που διαθέτουν τα στοιχεία του, αν το πλήθος τους είναι μη πεπερασμένο

π.χ. $E=\{x \mid x>6\}$

Ορισμός 1

Θεωρούμε ένα κλασικό σύνολο Χ, το οποίο ονομάζουμε **υπερσύνολο αναφοράς**, και έστω Α υποσύνολο του Χ. Το Α καλείται **ασαφές υποσύνολο** του Χ τότε και μόνο τότε αν :

$$A=\{(x\ ,\ \mu_A(x)\ )\ |\ x\in X\ \}$$

όπου μ_A μία συνάρτηση από το X στο [0,1] η οποία ονομάζεται **συνάρτηση συμμετοχής**

Είναι φανερό ότι στην ειδική περίπτωση που έχουμε {0,1} αντί [0,1] το ασαφές υποσύνολο Α εκφυλίζεται στο κλασικό υποσύνολο Α.

Συμβολισμός: Αν το Α είναι *πεπερασμένο*, τότε συμβολίζεται με:

$$A=\mu_A(x_1)/x_1+\ \dots\ +\mu_A(x_n)/x_n=\sum_{i=1}^{n}\ \mu_A(x_i)/x_i$$

Αν το είναι *μη-πεπερασμένο*, τότε συμβολίζεται με:

$$A=\int \mu_A(x)/x$$

Τα σύμβολα +, Σ , $\int$ υποδηλώνουν 'ένωση' και το σύμβολο '/' δεν δηλώνει διαίρεση.

Ορισμός 2

Ασαφές δυναμοσύνολο, F(X) του υπερσυνόλου αναφοράς X, ονομάζεται το σύνολο όλων των ασαφών υποσυνόλων του X.

Μια οικογένεια ασαφών υποσυνόλων του X θα λέγεται **ασαφής διαμέριση $P^n(X)$** του X, τάξης n($n\in N$) και θα συμβολίζεται με $\mathbf{A^n=\{A_1,A_2,\dots,A_n\}}$ αν και μόνο αν

$$A_i\neq Aj\ ,\ \forall\ i,j\in\ N_n\ (\ i\neq j\ )\quad 0<\sum_{k=1}^{m}A_i(x_k)<m\ ,\forall i\in N_n$$

Τα στοιχεία $A_i,i\in N_n$ της A^n θα λέγονται **κλάσεις** της ασαφούς διαμέρισης [2].

Ορισμός 3

Έστω X υπερσύνολο αναφοράς και Α ασαφές υποσύνολο του X. Τότε:

Α. Το κλασικό υποσύνολο **Supp(A)** του X, $Supp(A)\subset X$ καλείται **στήριγμα** (**support**) του ασαφούς υποσυνόλου όταν και μόνο όταν

$$Supp(A)=\{\ x\in\ X\ :\ \mu_A(x)\geq 0\ \}$$

Β. Το κλασικό υποσύνολο $L_a A$ του X καλείται **α-τομή (a-cut)** όταν και μόνο όταν:

$$L_a A=\{x\in X\ :\ \mu_A(x)\geq \alpha\ \}$$

Γ. Εάν $A,B\subseteq X$ τότε:

I. $\mathbf{A=B}$ όταν και μόνο όταν $\mu_A(x)=\mu_B(x)$ για όλα τα $x\in X$

II $\mathbf{A\subseteq B}$ όταν και μόνο όταν $\mu_A(x)\leq \mu_B(x)$ για όλα τα $x\in X$.

III.$\mathbf{A\subset B}$ όταν και μόνο όταν $\mu_A(x)\leq \mu_B(x)$ για όλα τα $x\in X$ και υπάρχει $y\in X$ τέτοιο ώστε $\mu_A(y)<\mu_B(y)$

Μια α-τομή L_a A ενός ασαφούς συνόλου A είναι ένα κοφτό σύνολο (δηλαδή όλα του τα στοιχεία έχουν βαθμό συμμετοχής στο L_a A ίσο με 1). Έτσι το L_a A μπορεί να γραφτεί με τη βοήθεια ενός τελεστή $C_a[.]$ ως:

$$L_a A = C_a[A] = \{(x,1) | \mu_A(x) \geq \alpha, x \in X\}$$

Προφανώς ο τελεστής $C_a[A]$ ανυψώνει στο 1 τους βαθμούς συμμετοχής όλων των $x \in X$, $\mu_A(x) \geq \alpha$ και μειώνει στο 0 τους βαθμούς όλων των άλλων στοιχείων.

Ορισμός 4

Έστω X υπερσύνολο αναφοράς και A ασαφές υποσύνολο του X, Τότε:

A. Το A καλείται **κυρτό** όταν και μόνο όταν:

$$\mu_A(\lambda x_1 + (1-\lambda)x_2 \geq \min(\mu_A(x_1), \mu_A(x_2)), \forall x_1, x_2 \in X, \lambda \in [0,1]$$

B. Το A καλείται **κανονικό** (**normal**) όταν και μόνο όταν υπάρχει τουλάχιστον ένα x στο υπερσύνολο αναφοράς για το οποίο ισχύει $\mu_A(x)=1$.

Γ. Ως **μέτρο** (**cardinality**) του A ορίζεται η ποσότητα :

$$|A| = \Sigma \mu_A(x) \text{ για κάθε } \chi \text{ που ανήκει στο } X$$

Δ. Ως **σχετικό μέτρο** (**relative cardinality**) του A ορίζεται η ποσότητα

$$||A|| = \frac{|A|}{|X|}$$

Ορισμός 5

Αντίστοιχα προς τις πράξεις 'ένωση', 'τομή', ΄συμπλήρωμα' των κλασικών συνόλων, τα ασαφή σύνολα έχουν παρόμοιες πράξεις.
Έστω X υπερσύνολο αναφοράς και A,B ασαφή υποσύνολα του X.
Ορίζουμε ως:

A. Τομή: $C=A \cap B = \{(x,\mu_C(x)) \mid x \in X, \mu_C(x)=\min(\mu_A(x), \mu_B(x))\}$

B. Ένωση : $D=A \cup B = \{(x,\mu_D(x)) \mid x \in X, \mu_D(x)=\max(\mu_A(x), \mu_B(x))\}$

Γ. Συμπλήρωμα: $A^C = \{(x,\mu_A{}^c(x)) \mid x \in X, \mu_A{}^c(x)=1-\mu_A(x)\}$
Σχετικά στο [1].

2.1.3 Συνάρτηση Συμμετοχής

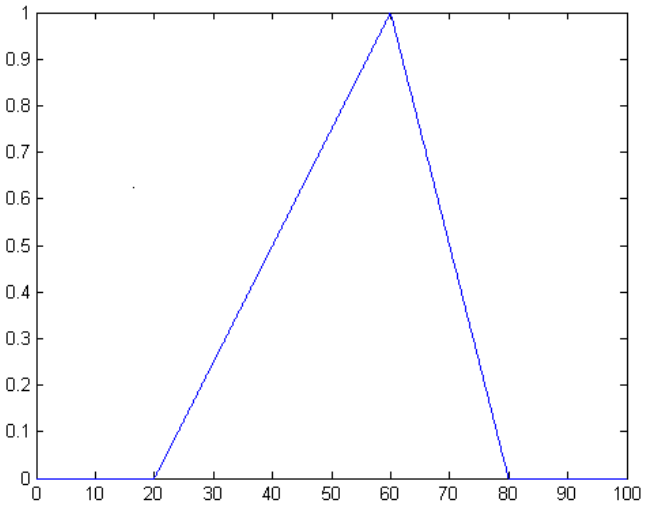
Όπως προκύπτει από τον ορισμό της, η συνάρτηση συμμετοχής (ΣΜ) καθορίζει ένα ασαφές σύνολο πλήρως. Επειδή τα περισσότερα ασαφή σύνολα που χρησιμοποιούνται έχουν ως υπερσύνολο αναφοράς το σύνολο των πραγματικών αριθμών, θα ήταν ανέφικτο να καταγραφούν όλα τα ζευγάρια που καθορίζουν μια ΣΜ. Ένας πιο συνοπτικός τρόπος για να οριστεί μια ΣΜ είναι να εκφραστεί με το μαθηματικό της τύπο.

Παραδείγματα κυριότερων ΣΜ

- Τριγωνική

Η τριγωνική ΣΜ καθορίζεται από τρεις παραμέτρους {a,b,c} που καθορίζουν τις συντεταγμένες των τριών γωνιών του τριγώνου που δημιουργεί η ΣΜ :

$$\text{Τριγωνική}(a,b,c)=\begin{cases} 0 & , x<a \\ \dfrac{x-a}{b-a} & , a \le x \le b \\ \dfrac{c-x}{c-b} & , b \le x \le c \\ 0 & , c \le x \end{cases}$$

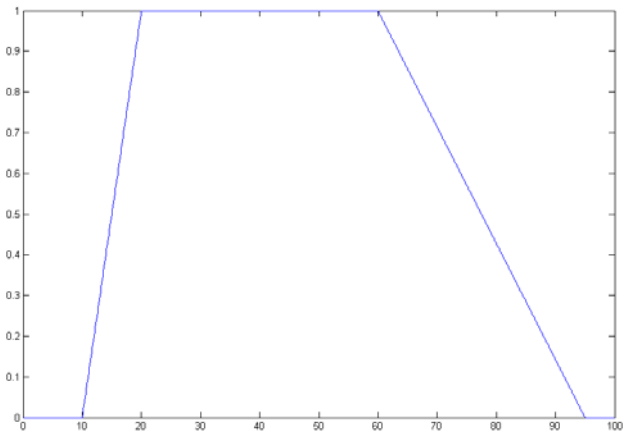


Σχήμα 2.1.3.1: Παράδειγμα τριγωνικής με a=20,b=60 ,c=80

- Τραπεζοειδής

Η τραπεζοειδής καθορίζεται από τέσσερις παραμέτρους{a,b,c,d} που καθορίζουν τις συντεταγμένες των τεσσάρων γωνιών του τραπεζίου που δημιουργεί η ΣΜ:

$$\text{Τραπεζοειδής}(a,b,c)=\begin{cases} 0 & , x<a \\ \dfrac{x-a}{b-a} & , a \le x \le b \\ 1 & , b \le x \le c \\ \dfrac{d-x}{d-c} & , c \le x \le d \\ 0 & , d \le x \end{cases}$$

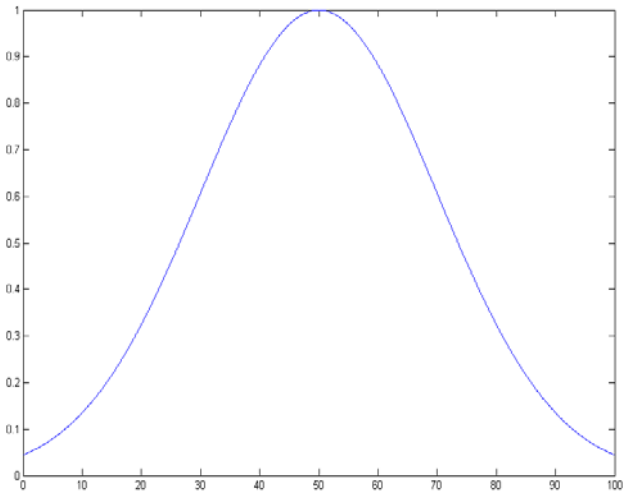


Σχήμα 2.1.3.2: Παράδειγμα τραπεζοειδούς με a=10,b=20,c=60,d=95

- Gauss (Γκαουσσιανή)

Η Γκαουσσιανή καθορίζεται από δύο παραμέτρους το κέντρο c και τη διασπορά σ

$$\text{Gauss}(c,\sigma) = \mathbf{e}^{-\frac{1}{2}\frac{(x-c)^2}{\sigma}}$$



Σχήμα 2.1.3.3: Παράδειγμα Gauss με c=50,σ=20

Για περισσότερα βλέπε στο [3]

2.1.4 Ασαφής ή Προσεγγιστική Συλλογιστική

2.1.4.1 Εισαγωγή

Ασαφής ή Προσεγγιστική συλλογιστική ονομάζεται η μεθοδολογία εξαγωγής ασαφών ή προσεγγιστικών συμπερασμάτων. Τρεις θεμελιώδεις έννοιες της συλλογιστικής αυτής είναι οι έννοιες της «γλωσσικής μεταβλητής», του «γενικευμένου κανόνα του θέτειν/αναιρείν» και της «ασαφούς σχέσης».

2.1.4.2 Γλωσσική Μεταβλητή

Γλωσσική μεταβλητή είναι μια μεταβλητή της οποίας οι τιμές δεν είναι αριθμοί αλλά λέξεις ή φράσεις σε μια φυσική ή τεχνητή γλώσσα. Οι τιμές αυτές παριστάνονται με ασαφή σύνολα και γι' αυτό οι γλωσσικές μεταβλητές ονομάζονται επίσης «ασαφείς μεταβλητές» ή «ασαφή κατηγορήματα» και συνήθως δηλώνονται ως:

'Χ είναι Α'

Ορισμός 6

Μαθηματικά, η Γλωσσική μεταβλητή ορίζεται ως η πεντάδα **<x,T(X),U,G,M>** όπου:
x: το όνομα της γλωσσικής μεταβλητής
T(X): το ασαφές σύνολο των τιμών της
U: το υπερσύνολο αναφοράς πάνω στο οποίο δομείται το σύνολο τιμών Τ(Χ)
G: ένας συντακτικός κανόνας που παράγει τα ονόματα των τιμών της x δηλαδή τα ονόματα των ασαφών συνόλων
M: ένας σημασιολογικός κανόνας που αποδίδει νόημα στα ονόματα

Μια γλωσσική μεταβλητή καλείται **δομημένη** όταν το σύνολο των τιμών της Τ(Χ) και το σύνολο των εννοιών Μ(x) που αποδίδονται σε αυτές μπορούν να καθοριστούν αλγοριθμικά.

Γλωσσικός διαμορφωτής (**τροποποιητής**) είναι ένας τελεστής που εφαρμοζόμενος σε μια τιμή (ασαφές σύνολο) μιας γλωσσικής μεταβλητής μεταβάλλει το νόημα της. Έτσι αν Α είναι το σύνολο των τιμών της γλωσσικής μεταβλητής τότε εφαρμαζόντας στα στοιχεία αυτού τον γλωσσικό τροποποιητή m προκύπτει ένα άλλο σύνολο τιμών B=m(A).

2.1.4.3 Γενικευμένος Κανόνας του Θέτειν και του Αναιρείν

Είναι γνωστό ότι η κλασική λογική στηρίζει τη διαδικασία απόδειξης εξ' ολοκλήρου σε «λογικές ταυτολογίες». Οι κυριότερες λογικές ταυτολογίες ,στις οποίες ανάγονται και όλες οι υπόλοιπες, είναι οι ακόλουθες :

Modus ponens:
$\{A \wedge (A \rightarrow B)\} \rightarrow B$ (**Κανόνας του Θέτειν**)

Modus tollens:
$\{(A \rightarrow B) \wedge \sim B\} \rightarrow \sim A$ (**Κανόνας του αναιρείν**)
Συλλογισμός:
$\{(A \rightarrow B) \wedge (B \rightarrow C)\} \rightarrow (A \rightarrow C)$ (**Αλυσίδα**)
Αντιθετοαντιστροφή:
$(A \rightarrow B) \rightarrow (\sim B \rightarrow \sim A)$

Επεκτείνοντας τους κανόνες modus ponens, modus tollens, ώστε να συμπεριλάβουν και τα ασαφή σύνολα, προκύπτουν οι γενικευμένοι κανόνες του θέτειν και του αναιρείν

Ο τροποποιημένος (ασαφής) κανόνας Modus Ponens (MP) που αναφέρεται και ως **Generalized Modus Ponens (GMP) (γενικευμένος κανόνας του θέτειν**) έχει την ακόλουθη δομή.

Συνεπαγωγή : EAN x=A TOTE y=B
Γεγονός : x=A′
Συμπέρασμα : y=B′

όπου τα A, B, A′ και B′ είναι ασαφή σύνολα.

Κριτήριο	**Γεγονός: το x είναι A′**	**Συμπέρασμα: το y είναι B′**
1	το x είναι A	το y είναι B
2-1	το x είναι πολύ A	το y είναι πολύ B
2-2	το x είναι πολύ A	το y είναι B
3-1	το x είναι περίπου A	το y είναι περίπου B
3-2	το x είναι περίπου A	το y είναι B
4-1	το x είναι όχι A	το y είναι άγνωστο
4-2	το x είναι όχι A	το y είναι όχι B

Πίνακας 2.1.4.3.1 Εμπειρικά κριτήρια εφαρμογής του κανόνα του **GMP**

Ο κανόνας GMP ονομάζεται επίσης «**ασαφής κανόνας του θέτειν**»(fuzzy modus ponens).

Μια επέκταση του GMP είναι η εξής:

Συνεπαγωγή:
Εάν x_1 είναι A_1 και … και x_n είναι A_n TOTE y είναι B
Γεγονότα(Δεδομένα):
x_1 είναι A'_1 και x_2 είναι A'_2 και x_n είναι A'_n
Συμπέρασμα: y είναι B′
Όπου A_i, A'_i (i=1,2,…,n) ,B και B′ είναι ασαφή σύνολα.

Αντίστοιχα ο **γενικευμένος (ασαφής) κανόνας του αναιρείν** (**Generalized Modus Tollens GMT**) έχει τη μορφή:

Συνεπαγωγή : EAN x είναι A TOTE y είναι B′
Γεγονός : y είναι B′
Συμπέρασμα : x είναι A′

Κριτήριο	Γεγονός: το y είναι Β′	Συμπέρασμα: το x είναι Α′
5	το y είναι όχι Β	το x είναι όχι Α
6	το y είναι όχι πολύ Β	το x είναι όχι πολύ Α
7	το y είναι περίπου Β	το x είναι περίπου Α
8-1	το y είναι Β	το x είναι άγνωστο
8-2	το y είναι Β	το x είναι Α

Πίνακας 2.1.4.3.1 Εμπειρικά κριτήρια εφαρμογής του κανόνα του **GMT**

2.1.4.4 Ασαφείς Σχέσεις

Οι ασαφείς σχέσεις αποτελούν γενίκευση των συνήθων (κοφτών) σχέσεων και μας δίνουν τη δυνατότητα να χειρισθούμε προβλήματα στα οποία υπάρχει αβεβαιότητα ή αμφιβολία. Από μαθηματική σκοπιά οι ασαφείς σχέσεις χρησιμοποιούνται για τη μοντελοποίηση (παράσταση) ασαφών συνεπαγωγών καθώς και τη συλλογιστική με αυτές. Βρίσκουν εφαρμογή σε όλες τις επιστημονικές περιοχές ασαφούς / προσεγγιστικής συλλογιστικής . Οι κοφτές (συνήθεις) σχέσεις αποτελούν στοιχεία του Καρτεσιανού γινομένου ορισμένων χώρων.

Ορισμός 7

Έστω X και Y υπερσύνολο αναφοράς . Τότε με τον όρο «**ασαφής σχέση R**» εννοούμε ένα ασαφές σύνολο στο Καρτεσιανό γινόμενο X x Y = {(x,y)/ x∈ X, y∈ Y} το οποίο σύνολο χαρακτηρίζεται από τη συνάρτηση συμμετοχής μ_R:

$$\mu_R: X \times Y \rightarrow [0,1]$$

Στην περίπτωση όπου X =Y έχουμε μια ασαφή σχέση επί του X. Η συνάρτηση συμμετοχής $\mu_R(x ,y)$ παριστά για κάθε ζεύγος (x ,y) το βαθμό σύνδεσης ανάμεσα στα x και y.

Ορισμός 8

Οι πράξεις μεταξύ των ασαφών σχέσεων ορίζονται κατά αναλογία των πράξεων των ασαφών συνόλων . Έτσι για τις ασαφείς σχέσεις R_1 και R_2 στο X x Y έχουμε:

Τομή $R_1 \cap R_2 : \mu_{R1 \cap R2}(x ,y) = \min \{\mu_{R1}(x ,y), \mu_{R2}(x ,y)\}$

Ένωση $R_1 \cup R_2 : \mu_{R1 \cup R2}(x ,y) = \max \{\mu_{R1}(x ,y), \mu_{R2}(x ,y)\}$

Συμπλήρωμα $R_1^c : \mu_{R1c} (x ,y) = 1- \mu_{R1} (x ,y)$

Έγκλειση $R_1 \subseteq R_2 : \mu_{R1}(x ,y) \leq \mu_{R2}(x ,y)$

Ορισμός 9

Εάν τα υπερσύνολα αναφοράς είναι πεπερασμένα, δηλαδή $X=\{x_1,x_2,\ldots,x_m\}$ και $Y=\{y_1,y_2,\ldots,y_n\}$ τότε μια ασαφής σχέση R στο X x Y παριστάνεται και χρησιμοποιείται ως μια **m**x**n** μήτρα.

$$\underline{R}=\lfloor \mu_R (x_i ,y_j) \rfloor$$

με στοιχεία $\mu_R(x_i, y_j)$, i=1,2,…,m και y=1,2,…,n

Η μήτρα ονομάζεται **ασαφής μήτρα** ή **ασαφής σχεσιακή μήτρα** ή **μήτρα μιας ασαφούς σχέσης** και τα στοιχεία της έχουν τιμές μεταξύ 0 και 1.

Ορισμός 10

Ένας *ασαφής κανόνας* (**συνεπαγωγή**) :

$$K: \text{EAN A TOTE B} \quad (\text{ή } A \rightarrow B)$$

όπου τα A και B είναι τα ασαφή σύνολα

$$A=\{(x_i, \mu_A(x_i)) / x_i \in X\}, \quad B=\{(y_j, \mu_B(y_j)) / y_j \in Y\}$$

μπορεί να παρασταθεί με μια ασαφή σχέση

$$R=\{((x_i, y_j), \mu_R(x_i, y_j)) / x_i \in X, y_j \in Y\}$$

όπου το (i,j) στοιχείο του $\mu_R(x_i, y_j)$ της σχεσιακής της μήτρας μπορεί να υπολογισθεί με τη χρήση κανόνων όπως:

- Αριθμητικός κανόνας Zadeh
  $\mu_R(x_i, y_j) = \min\{1, 1 - \mu_A(x_i) + \mu_B(y_j)\}$

- Κανόνας Ελαχίστου (Mamdani)
  $\mu_R(x_i, y_j) = \min\{\mu_A(x_i), \mu_B(y_j)\}$

- Κανόνας Γινομένου
  $\mu_R(x_i, y_j) = \mu_A(x_i)\, \mu_B(y_j)$

Ορισμός 11

Η **σύνθεση «max-min»** είναι ένα από τα σημαντικότερα αποτελέσματα της ασαφούς (προσεγγιστικής) συλλογιστικής γιατί παρέχει ένα συγκεκριμένο (μαθηματικό) τρόπο υλοποίησης του γενικευμένου κανόνα modus ponens

Δίνονται τα ασαφή σύνολα

$$A=\{(x_i, \mu_A(x_i)) / x_i \in X\}, \quad B=\{(y_j, \mu_B(y_j)) / y_j \in Y\}$$

και μια ασαφή σχέση επί του X x Y

$$R=\{((x_i, y_j), \mu_R(x_i, y_j)) / x_i \in X, y_j \in Y\}$$

Τότε με δεδομένα τα A και R , το B δίνεται από τη σχέση :

$$B = A \circ R$$

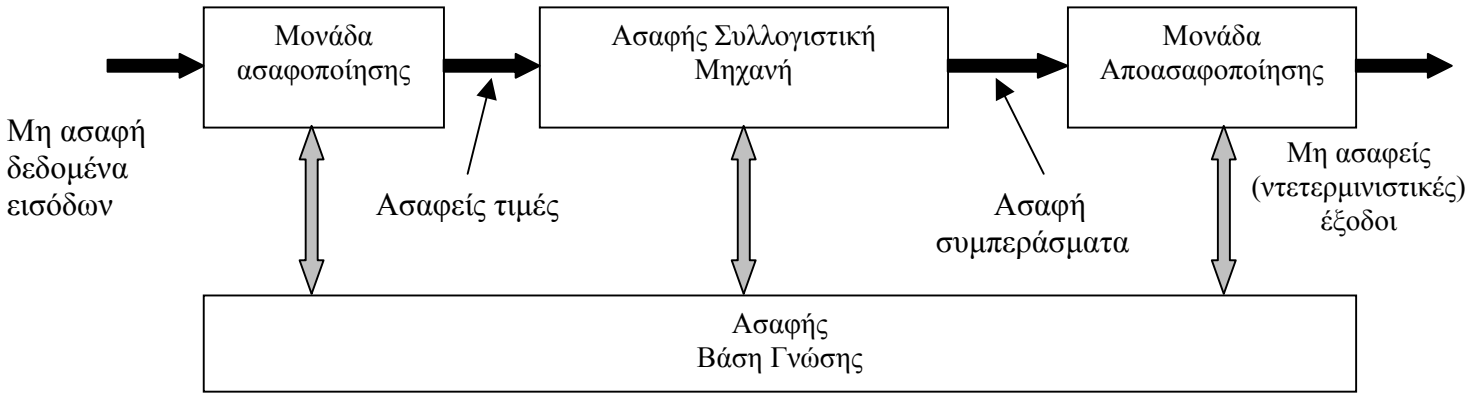
με $\mu_B(x) = \max_x \{\min[\mu_A(x), \mu_R(x,y)]\}$

όπου το ‘$\circ$’ συμβολίζει τον τελεστή της σύνθεσης max-min [1]

2.1.5 Ασαφή Συστήματα (Fuzzy Inference Systems/FIS)

2.1.5.1 Εισαγωγή

Τα ασαφή συστήματα περιλαμβάνουν κάθε σύστημα απόφασης και ελέγχου που λειτουργεί σε αβέβαιο περιβάλλον και μοντελοποιείται με ασαφείς μεταβλητές. Τα ασαφή συστήματα ανήκουν στην κατηγορία των ευφυών συστημάτων και βρίσκουν εφαρμογή ολοένα και σε περισσότερα πρακτικά προβλήματα.



Σχήμα 2.1.5.1.1 Γενική αρχιτεκτονική ασαφούς συστήματος

Η γενική αρχιτεκτονική (δομή) των ασαφών συστημάτων παρουσιάζεται στο σχήμα και περιλαμβάνει τέσσερις μονάδες:

1. Μία βάση ασαφών κανόνων της μορφής EAN-TOTE (Ασαφής βάση Γνώσης).
2. Μία ασαφή συλλογιστική μηχανή (μηχανισμό εξαγωγής ασαφών συμπερασμάτων).
3. Μία μονάδα ασαφοποίησης (ασαφοποιητική μονάδα διεπαφής) η οποία μετατρέπει τα δεδομένα εισόδου σε ασαφή σύνολα.
4. Μία μονάδα απο-ασαφοποίησης (από-ασαφοποιητική μονάδα διεπαφής) η οποία μετατρέπει τα ασαφή συμπεράσματα / αποφάσεις σε σαφώς καθορισμένη μορφή.

Η *ασαφής βάση γνώσης* περιέχει συνήθως εκτός από τους ασαφείς (γλωσσικούς) κανόνες και ένα τμήμα βάσης αριθμητικών δεδομένων τα οποία απαιτούνται για τη διαδικασία εξαγωγής των αποτελεσμάτων. Οι κανόνες της βάσης γνώσης λαμβάνονται συνήθως από εμπειρογνώμονες και από διαδικασίες προσομοίωσης.

Η *ασαφής συλλογιστική μηχανή* αποτελεί τον πυρήνα του ασαφούς συστήματος και περιέχει τη λογική λήψης αποφάσεων

Η *μονάδα ασαφοποίησης (ασαφοποιητής)* εκτελεί τις παρακάτω εργασίες:

- Μετράει (παραλαμβάνει) τις (μη ασαφείς) τιμές των εισόδων του συστήματος
- Απεικονίζει τις περιοχές μεταβολής των τιμών εισόδου σε κατάλληλα υπερσύνολα αναφοράς

- Ασαφοποιεί τις εισερχόμενες τιμές των εισόδων, δηλαδή τις μετατρέπει σε ασαφή γλωσσική μορφή.

Η *μονάδα απο-ασαφοποίησης (από-ασαφοποιητής)* εκτελεί τις εξής δύο εργασίες

- Απεικονίζει τις περιοχές μεταβολής των μεταβλητών εξόδου σε αντίστοιχα υπερσύνολα αναφοράς
- Απο-ασαφοποιεί τα αποτελέσματα που δίνει η ασαφής συλλογιστική μηχανή, δηλαδή τα μετατρέπει σε «ντετερμινιστική»(μη-ασαφής) μορφή για παραπέρα χρήση από επόμενα συστήματα ή διεργασίες απόφασης.

2.1.5.2 Μέθοδοι Ασαφοποίησης

Ο ασαφοποιητής υλοποιεί μία απεικόνιση από το σύνολο πραγματικών τιμών εισόδου $x=\{x1,x2,\ldots,x_n\}\in X$ στο ασαφές σύνολο A του υπερσυνόλου αναφοράς X. Δύο επιλογές αυτής της απεικόνισης είναι οι ακόλουθες :

Μονότιμος ασαφοποιητής

Στον μονότιμο ασαφοποιητή (singleton fuzzifier) το ασαφές σύνολο A έχει στήριγμα (support), δηλαδή suppA=x ,ήτοι:

$$\mu_A(x')=\begin{cases} 1 \text{ ,εάν } x'=x \\ 0 \text{ ,εάν } x'\neq x \end{cases}$$

δηλαδή μοναδιαία συνάρτηση συμμετοχής .

Μη μονότιμος ασαφοποιητής

Στην περίπτωση αυτή θεωρούμε ότι $\mu_A(x)=1$ και ότι το $\mu_A(x')$ μικραίνει όσο το x' απομακρύνεται από το x.

Για παράδειγμα :

$$\mu_A(x')=\exp\left[-\frac{(x'-x)^T(x'-x)}{\sigma^2}\right]$$

όπου σ^2 είναι μια παράμετρος η οποία καθορίζει τη μορφή του $\mu_A(x')$.

Υπάρχει λοιπόν μια τιμή του x που έχει τη μέγιστη συνάρτηση συμμετοχής (ίση με 1) στο ασαφές σύνολο A, αλλά όσο πιο πολύ απομακρυνόμαστε από την τιμή αυτή τόσο μειώνεται η τιμή της συνάρτησης συμμετοχής στο A.

Στην πράξη χρησιμοποιείται συνήθως ο μονότιμος ασαφοποιητής ενώ ο μη-μονότιμος ασαφοποιητής είναι καταλληλότερος όταν οι είσοδοι υπόκεινται σε θόρυβο.

2.1.5.3 Μέθοδοι Απο-ασαφοποίησης

Απο-ασαφοποίηση είναι η διαδικασία μετατροπής ενός ασαφούς συνόλου B σε μια τιμή w_o, η οποία είναι και η έξοδος του ασαφούς συστήματος. Οι κυριότερες μέθοδοι απο-ασαφοποίησης είναι οι εξής:

Μέθοδος κέντρου βάρους

Στη μέθοδο αυτή που είναι γνωστή ως μέθοδος COG (Center Of Gravity) η τιμή w_o δίνεται από τη σχέση:

$$w_o = \frac{\sum_i w_i \mu_B(w_i)}{\sum_i \mu_B(w_i)}$$

Μέθοδος Μέσης Τιμής των Μεγίστων

Εδώ η τιμή w_o δίνεται από τη σχέση:

$$w_o = \frac{\sum_{j=1}^{m} w_j}{m}$$

όπου η τιμή w_o είναι η τιμή που αντιστοιχεί στο j μέγιστο της συνάρτησης συμμετοχής. Η μέθοδος είναι γνωστή ως μέθοδος MOM (Mean of Maxima).

Τροποποιημένη Μέθοδος Κέντρου Βάρους

$$w_o = \frac{\sum_i w_i [\mu_B(w_i) / \delta_i]}{\sum_i \mu_B(w_i) / \delta_i}$$

όπου το δ_i χαρακτηρίζει το σχήμα της συνάρτησης συμμετοχής.

2.1.5.4 Ασαφής Βάση Γνώσης

Η ασαφής βάση γνώσης αποτελείται από μια συλλογή κανόνων EAN-TOTE της παρακάτω μορφής:

$$R^l: EAN\ x_1\ είναι\ A^l_1\ KAI\ \ldots\ KAI\ x_n\ είναι\ A^l_n\ TOTE\ y\ είναι\ B^l$$

όπου τα A^l_1 και B^l είναι ασαφή σύνολα επί των $X_i \subset \Re$ και $Y \subset \Re$ αντίστοιχα και $x=[x_1,x_2,\ldots,x_n]^T$ και y είναι γλωσσικές μεταβλητές

Υπάρχουν δύο βασικές μέθοδοι κατασκευής ασαφών κανόνων:

- Από εμπειρογνώμονες
- Από αλγόριθμους εκπαίδευσης βασισμένους σε δεδομένα μετρήσεων

Για τον προσδιορισμό της μορφής των A^l_1 και B^l έχουμε τις ακόλουθες μεθόδους:

- Αν οι κανόνες καθορίζονται από κάποιο εμπειρογνώμονα τότε αυτός πρέπει να καθορίσει και τη μορφή τους.
- Αν οι κανόνες καθορίζονται με βάση δεδομένα μετρήσεων, τότε χρησιμοποιούμε κάποια αυθαίρετη μορφή με πιο συνηθισμένες τις τριγωνικές, τις τραπεζοειδείς και τις Gauss. Κάθε τέτοια συνάρτηση συνοδεύεται και από κάποιες παραμέτρους οι οποίες πρέπει να καθοριστούν με βάση τις δεδομένες μετρήσεις.

2.1.5.5 Μηχανισμός Ασαφούς Συλλογισμού

Αν στη βάση κανόνων υπάρχουν συνολικά n κανόνες δηλαδή R_I , i=1,2,…,n τότε η βάση μπορεί να θεωρηθεί πως αποτελεί μια μοναδική σχέση

$$R= \cup R_i \ \ (i=1,2,\ldots,n)$$

Το ζητούμενο είναι πως θα συνδυαστούν τα δεδομένα με τους κανόνες που υπάρχουν στη βάση κανόνων, ώστε να εξαχθεί το τελικό συμπέρασμα [4].

Οι μηχανισμοί ασαφούς συλλογισμού (ή μηχανισμοί εξαγωγής συμπερασμάτων) αναφέρονται στην μέθοδο με την οποία παράγονται συμπεράσματα από τους κανόνες και πως αυτά τα αποτελέσματα-συμπεράσματα συνδυάζονται ώστε να παραχθεί το ολικό αποτέλεσμα.

Δηλαδή πρέπει να καθοριστεί

- ο τρόπος με τον οποίο υπολογίζεται η προσαρμοστικότητα (δύναμη) του αριστερού μέλους του κανόνα για κάθε είσοδο
- ο τρόπος με τον οποίο εφαρμόζεται η προσαρμοστικότητα αυτή στα ασαφή σύνολα του δεξιού μέλους κάθε κανόνα για να λάβουμε το συμπέρασμα κάθε κανόνα (συμπέρασμα εξόδου κάθε κανόνα)
- ο τρόπος με τον οποίο συνδυάζονται αυτά τα αποτελέσματα ώστε να προκύψει το συνολικό (τελικό) αποτέλεσμα

Παράδειγμα μηχανισμού ασαφούς συλλογισμού είναι η μέθοδος Mamdani η οποία έχει:

- *Κανόνες της μορφής*:

  ΕΑΝ x είναι A_i ΚΑΙ y είναι B_i ΤΟΤΕ z είναι C_i (i=1,..n όπου n=κανόνες)

- *Προσαρμοστικότητα κανόνα I:*

  $\mu_i=\min\{\mu_{Ali}(x_i),\ldots,\mu_{Alm-1}(x_{m-1}), \mu_{Bm}(y_m)\}$
  όπου $A^l_1,\ldots, A^l_{m-1}$ ασαφή σύνολα εισόδου και B_m ασαφές σύνολο εξόδου

- Συμπέρασμα κανόνα και Ολικό Αποτέλεσμα:
  Κανόνας Ελαχίστου(Mamdani) ή Κανόνας Γινομένου

Οι μέθοδοι μηχανισμών ασαφούς συλλογισμού είναι:
- η μέθοδος Takagi-Sugeno
- η μέθοδος Tsukamoto

Η λογική των ασαφών συστημάτων αντιστοιχεί στην λογική «διαίρει και βασίλευε». Το μέλος της υπόθεσης (το αριστερό μέλος) του κανόνα καθορίζει μια τοπική ασαφή περιοχή ενώ το μέλος του συμπεράσματος (το δεξιό μέλος) περιγράφει την συμπεριφορά εντός της τοπικής περιοχής δια μέσου διαφόρων συνιστωσών. Η συνιστώσα μπορεί να είναι μια συνάρτηση συμμετοχής (Mamdani, Tsukamoto ασαφή συστήματα), μια σταθερή τιμή (Takagi-Sugeno μηδενικού βαθμού) ή μια γραμμική εξίσωση (Takagi-Sugeno πρώτου βαθμού). Άρα κανόνες με διαφορετικά δεξιά μέλη καταλήγουν σε διαφορετικά ασαφή συστήματα, αν και μπορεί να έχουν το ίδιο μέλος υπόθεσης. Περισσότερα στα [1], [3].

2.1.6 Μέτρο Ομοιότητας

Το μέτρο ομοιότητας E(A,B) δύο ασαφών συνόλων A και B δείχνει το βαθμό ομοιότητας του A με το B και ορίζεται ως:

$$\mathbf{E(A,B)= Βαθμός\ (A=B)}$$
$$\mathbf{= Βαθμός\ (A\subseteq B\ και\ B\subseteq A)\ ,\ E\in[0,1]}$$

Προφανώς E(A,B)=1 αν A=B

Αλγεβρικά προκύπτει:
$\mathbf{E(A,B)=|A\cap B|\ /\ |A\cup B|}$
Περισσότερα στο [1].

2.2 Νευρωνικά Δίκτυα

2.2.1 Εισαγωγικό Σημείωμα

Ο ανθρώπινος εγκέφαλος αποτελεί μια πηγή φυσικής νοημοσύνης και έναν αξιοσημείωτο παράλληλο υπολογιστή. Ο εγκέφαλος επεξεργάζεται ελλιπείς πληροφορίες οι οποίες συλλέγονται από τους μηχανισμούς της αντίληψης, με εξαιρετικά υψηλούς ρυθμούς. Τα νευρικά κύτταρα λειτουργούν σε ταχύτητες περίπου 10^6 φορές χαμηλότερες από τις σύγχρονες ηλεκτρονικές πύλες, αλλά παρόλα αυτά ο εγκέφαλος επεξεργάζεται ακουστικές και οπτικές πληροφορίες πολύ ταχύτερα από ότι οι σύγχρονοι υπολογιστές.

Εμπνευσμένοι από τα βιολογικά νευρικά συστήματα, πολλοί ερευνητές εξερευνούν την περιοχή των *τεχνητών νευρωνικών δικτύων*, μια καινοτόμα μη αλγοριθμική προσέγγιση στο πρόβλημα της επεξεργασίας της πληροφορίας. Ο εγκέφαλος μοντελοποιείται ως ένα συνεχούς-χρόνου μη γραμμικό δυναμικό σύστημα με ποικίλες αρχιτεκτονικές διασύνδεσης το οποίο αναμένεται να μιμείται τους μηχανισμούς του εγκεφάλου και να προσεγγίζει ευφυή συμπεριφορά. Οι διασυνδέσεις υλοποιούνται ως κατανεμημένες αναπαραστάσεις με την μορφή βαρών μεταξύ ενός μεγάλου αριθμού διασυνδεδεμένων νευρώνων.

Η περιοχή των *νευρωνικών δικτύων* εγκαινιάζεται με την εργασία των Mc Culloh και Pitts οι οποίοι μελέτησαν το μοντέλο του βασικού κυττάρου του ανθρώπινου εγκεφάλου. Το μοντέλο αυτό αποτελείται από μεταβλητές αντιστάσεις και αθροιστικούς ενισχυτές οι οποίοι αναπαριστούν τα συναπτά βάρη που συνδέουν τα *νευρόνια* μεταξύ τους. Αργότερα ο Rosenblatt ανέπτυξε την έννοια του *Perceptron* ως μια νέα λύση στο πρόβλημα της αναγνώρισης προτύπων και απέδειξε το αντίστοιχο θεώρημα σύγκλισης του αλγορίθμου μάθησης του Perceptron. Στην συνέχεια οι Widrow και Hoff θεμελίωσαν τον αλγόριθμο μάθησης *μέσων ελαχίστων τετραγώνων* τον οποίο χρησιμοποίησαν στο μοντέλο *Adaline*. Από τους Werbos, Parker και Rumelhart εισάγεται η ιδέα του δικτύου *Backpropagation*.

Κατά την διάρκεια της δεκαετίας του 80’ υπάρχει αναζωπύρωση της έρευνας γύρω από το αντικείμενο των νευρωνικών δικτύων. Έτσι ο Hopfield αναπτύσσει τα *νευρωνικά δίκτυα Hopfield* και χρησιμοποιεί την ιδέα της συνάρτησης ενέργειας για την ανάλυσή τους. Ακολούθως εισάγεται η διαδικασία του *simulated annealing* για την επίλυση προβλημάτων βελτιστοποίησης. Οι Broomhead και Lowe μελετούν εκτενώς τα νευρωνικά δίκτυα *ακτινικών συναρτήσεων βάσης*. Στον πίνακα που ακολουθεί εμφανίζονται τα κυριότερα μοντέλα νευρωνικών δικτύων καθώς και οι ερευνητές που τα ανέπτυξαν.

Έτος	**Νευρωνικό Δίκτυο**	**Ερευνητές**
1942	Νευρόνιο Mc Culloh / Pitts	Mc Culloh, Pitts
1957	Perceptron	Rosenblatt
1960	Adaline / Madaline	Widrow
1974	Backpropagation (BP)	Werbos, Parker, Rumelhart
1978	Neocognitron	Fukushima
1980	Adaptive Resonance Theory	Kohonen
1980	Self Organizing Map	Kohonen

1982	Νευρωνικό Δίκτυο Hopfield	Hopfield
1985	Μηχανή Boltzmann	Hinton, Sejnowsky
1988	Κυτταρικό Νευρωνικό Δίκτυο	Chua, Yang
1988	Δίκτυα Radial Basis Functions	Broohead, Lowe

Τα νευρωνικά δίκτυα έχουν φθάσει σε ένα υψηλό επίπεδο ανάπτυξης και θα συνεχίσουν να αναπτύσσονται προς διάφορες κατευθύνσεις συνδυαζόμενα με ασαφή συστήματα αλλά και με άλλες τεχνικές ανάλυσης και σχεδίασης ευφυών συστημάτων.

2.2.2 Βασικές Έννοιες

Τα νευρωνικά δίκτυα (*ΝΔ*) εκτελούν υπο-συμβολική επεξεργασία πληροφορίας η οποία βασίζεται σε μοντέλα του ανθρωπίνου εγκεφάλου τα οποία εμπνέονται από την βιολογία και τη νευροφυσιολογία. Για την χρήση των μοντέλων αυτού του είδους διατίθενται μέθοδοι που υλοποιούν πολύπλοκες συναρτήσεις και λειτουργίες. Για την εφαρμογή των μεθόδων αυτών δεν απαιτείται ρητή γνώση σε αντίθεση με ότι ισχύει κατά την εφαρμογή συμβολικών μεθόδων της τεχνητής νοημοσύνης. Στην υποσυμβολική προσέγγιση δεν δίνεται η υπό εξέταση σχέση ρητά αλλά κωδικοποιείται στη δομή ενός νευρωνικού δικτύου.

Τα νευρωνικά δίκτυα είναι συστήματα μεγάλης κλίμακας τα οποία περιέχουν ένα μεγάλο αριθμό μη γραμμικών επεξεργαστών ειδικού τύπου οι οποίοι καλούνται νευρόνια. Κάθε ΝΔ χαρακτηρίζεται από μια κατάσταση, ένα σύνολο εισόδων με βάρη που προέρχονται από άλλα νευρόνια και μια εξίσωση η οποία περιγράφει τη δυναμική λειτουργία του ΝΔ. Τα βάρη του ΝΔ ανανεώνονται μέσω μιας διαδικασίας μάθησης (εκπαίδευσης) η οποία πραγματοποιείται με την ελαχιστοποίηση κάποιας συνάρτησης κόστους (σφάλματος) ανανεώνοντας ακολουθιακά τα βάρη. Οι βέλτιστες τιμές των βαρών αποθηκεύονται (ως τιμές των διασυνδέσεων μεταξύ των νευρονίων) και χρησιμοποιούνται κατά την εκτέλεση της εργασίας για την οποία προορίζεται το ΝΔ.

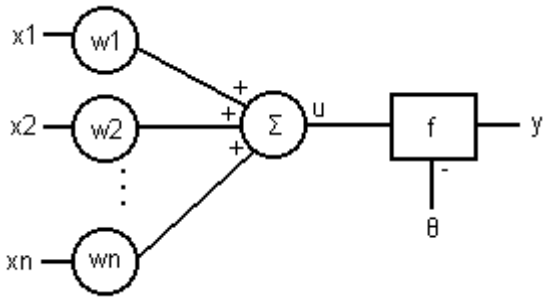
Κατά έναν ισοδύναμο ορισμό ένα ΝΔ είναι μια αρχιτεκτονική δομή αποτελούμενη από ένα πλήθος διασυνδεδεμένων μονάδων (νευρόνια). Κάθε μονάδα χαρακτηρίζεται από εισόδους και εξόδους και υλοποιεί τοπικά έναν απλό υπολογισμό. Κάθε σύνδεση μεταξύ δυο μονάδων χαρακτηρίζεται από μια τιμή βάρους. Οι τιμές των βαρών των συνδέσεων αποτελούν την γνώση που είναι αποθηκευμένη στο δίκτυο και καθορίζουν την λειτουργικότητά του. Η έξοδος κάθε μονάδας καθορίζεται από τον τύπο της μονάδας, την διασύνδεση με τις υπόλοιπες μονάδες και πιθανώς κάποιες εξωτερικές εισόδους. Πέρα από μια πιθανή δεδομένη (εκ κατασκευής) λειτουργική ικανότητα ενός δικτύου, συνήθως ένα δίκτυο αναπτύσσει μια συνολική λειτουργικότητα μέσω μιας μορφής εκπαίδευσης.

Η συνολική λειτουργικότητα ενός ΝΔ καθορίζεται από την τοπολογία του δικτύου, τα χαρακτηριστικά των νευρώνων, την μέθοδο εκπαίδευσης και από τα δεδομένα με τα οποία γίνεται η εκπαίδευση. Συχνά ο υπολογισμός που εκτελεί κάθε νευρόνιο είναι απλός και κοινός για όλα τα νευρόνια. Επειδή οι νευρώνες λειτουργούν παράλληλα (ταυτόχρονα) και ο αριθμός τους μπορεί να είναι πολύ μεγάλος, τα ΝΔ αποτελούν χαρακτηριστικό παράδειγμα παράλληλου μαζικού υπολογισμού.

Τα νευρωνικά δίκτυα είναι κατάλληλα για προβλήματα στα οποία ο συνήθης υπολογισμός δεν είναι αποδοτικός ή δεν γίνεται να υλοποιηθεί αναλυτικά. Τα σημαντικότερα χαρακτηριστικά των νευρωνικών δικτύων που τα διαφοροποιούν από άλλα ευφυή συστήματα είναι η δυνατότητα *μάθησης* (*learning*) και *προσαρμογής* (*adaptation*) σε διαφορετικούς χώρους προβλημάτων.

2.2.3 Μοντέλο Τεχνητού Νευρονίου (Νευρώνας)

Το μοντέλο αυτό στηρίζεται στο μοντέλο Mc Culloh και Pitts και έχει την μορφή του σχήματος 2.2.3.1.



Σχήμα 2.2.3.1: Μοντέλο τεχνητού νευρονίου.

Παρατηρείται ότι ο νευρώνας είναι μια θεμελιακή μονάδα επεξεργασίας πληροφορίας η οποία αποτελείται από τρία συστατικά στοιχεία:

- Ένα σύνολο κλάδων διασύνδεσης (συνάψεων).
- Ένα κόμβο άθροισης.
- Μια συνάρτηση ενεργοποίησης.

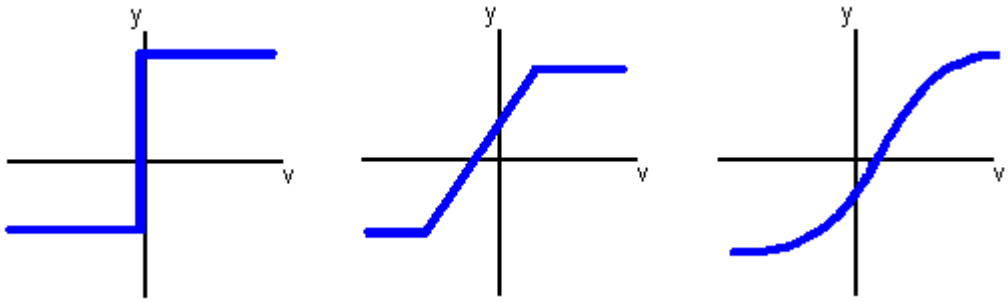
Κάθε κλάδος διασύνδεσης έχει ένα βάρος (weight) το οποίο είναι θετικό εάν η σύναψη είναι διεγερτικού τύπου (excitatory) και αρνητικό εάν η σύναψη είναι απαγορευτικού τύπου (inhibitory). Ο κόμβος άθροισης αθροίζει τα σήματα εισόδου πολλαπλασιαζόμενα με τα αντίστοιχα βάρη των συνάψεων. Συνεπώς ο κόμβος άθροισης είναι μια μονάδα γραμμικού συνδυασμού. Η συνάρτηση ενεργοποίησης (squashing function) περιορίζει το επιτρεπτό πλάτος του σήματος εξόδου σε κάποια πεπερασμένη τιμή (συνήθως στα κανονικοποιημένα διαστήματα [0,1], [-1,1]). Τέλος το μοντέλο του νευρονίου περιέχει επίσης ένα κατώφλι θ που εφαρμόζεται εξωτερικά και πρακτικά υποβιβάζει την καθαρή είσοδο στην συνάρτηση ενεργοποίησης. Συνεπώς η περιγραφή του νευρώνα γίνεται από τις παρακάτω εξισώσεις:

$$u = \sum_{i=1}^{n} w_i x_i$$

$$y = f(u-\theta),\ \theta>0$$

όπου x_i = (i = 1,2,…,n) είναι τα σήματα εισόδου, w_i (i = 1,2,..,n) είναι τα συναπτικά βάρη του νευρονίου, u είναι η έξοδος του γραμμικού συνδυαστή, θ είναι το κατώφλι, f είναι η συνάρτηση ενεργοποίησης και y είναι το σήμα εξόδου του νευρονίου. Σε γενικές γραμμές η συνάρτηση ενεργοποίησης μπορεί να έχει μια από τις παρακάτω μορφές:

- Συνάρτηση κατωφλίου.
- Κατά τμήματα γραμμική συνάρτηση.
- Συνεχής σιγμοειδής συνάρτηση.

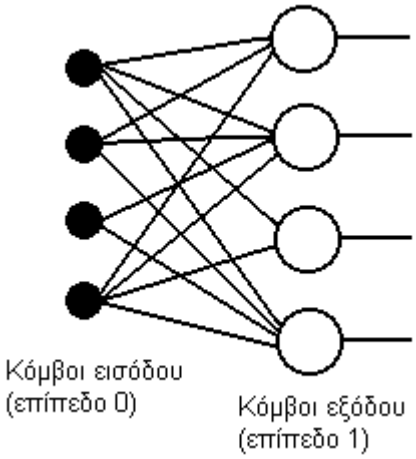


Σχήμα 2.2.3.2: Συναρτήσεις ενεργοποίησης νευρώνα.

2.2.4 Αρχιτεκτονικές Νευρωνικών Δικτύων

Η αρχιτεκτονική των νευρωνικών δικτύων αφορά την τοπολογική διάταξη αλλά και την μεθοδολογία δόμησης πολλαπλών νευρονίων. Τα χαρακτηριστικά που καθορίζουν την αρχιτεκτονική ενός ΝΔ είναι το πλήθος των στρωμάτων (layers) και οι συνδέσεις ανάμεσα στους νευρώνες. Ένα επιπρόσθετο χαρακτηριστικό το οποίο σχετίζεται με άμεσο τρόπο με τον τρόπο δόμησης των νευρονίων είναι ο αλγόριθμος μάθησης που χρησιμοποιείται για την εκπαίδευση του νευρωνικού δικτύου.

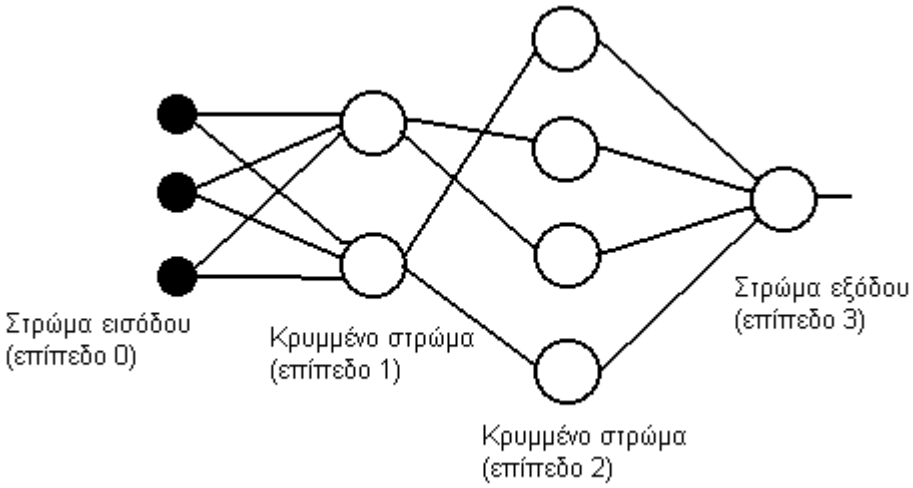
ΝΔ Προσοτροφοδότησης: Τα νευρωνικά δίκτυα προσοτροφοδότησης αποτελούνται από νευρόνια οργανωμένα σε στρώματα. Περιλαμβάνουν ένα στρώμα εισόδου από κόμβους πηγής (source nodes) το οποίο προβάλλεται πάνω σε ένα στρώμα νευρονίων εξόδου (output nodes) αλλά όχι αντίστροφα. Το ΝΔ αυτό καλείται νευρωνικό δίκτυο προσοτροφοδότησης ενός μοναδικού στρώματος, όπου το μοναδικό στρώμα είναι το στρώμα νευρονίων εξόδου.



Σχήμα 2.2.4.1: Μονοστρωματικό ΝΔ προσοτροφοδότησης.

Στην γενική περίπτωση, ένα ΝΔ προσοτροφοδότησης περιέχει ένα ή περισσότερα κρυμμένα-ενδιάμεσα στρώματα των οποίων οι υπολογιστικοί κόμβοι (hidden nodes) παρεμβαίνουν μεταξύ των εξωτερικών εισόδων και των εξόδων του νευρωνικού δικτύου. Στα δίκτυα αυτά τα οποία ονομάζονται πολυστρωματικά ΝΔ προσοτροφοδότησης τα στοιχεία του διανύσματος εισόδου που προέρχονται από τους κόμβους εισόδου εισέρχονται στο πρώτο κρυμμένο στρώμα υπολογιστικών κόμβων.

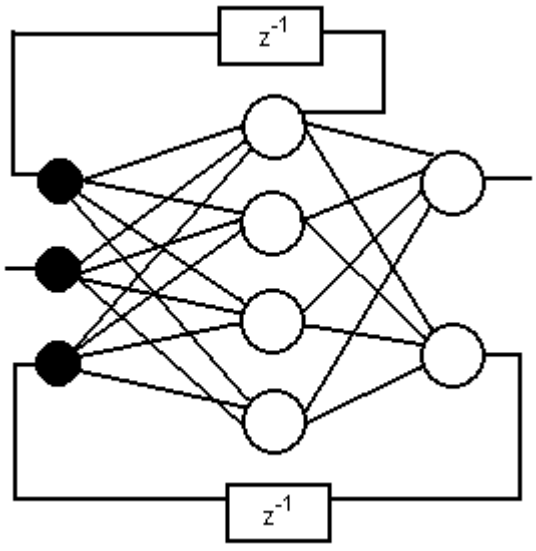
Ομοίως οι έξοδοι του πρώτου κρυμμένου στρώματος εισέρχονται, ως είσοδοι πλέον, στους κόμβους του δεύτερου κρυμμένου στρώματος. Αυτή η διαδικασία συνεχίζει μέχρι το τελικό στρώμα κόμβων οι οποίοι δίνουν την συνολική απόκριση στα πρότυπα εισόδου.



Σχήμα 2.2.4.2: Πολυστρωματικό ΝΔ προσοτροφοδότησης (μερικά διασυνδεδεμένο).

Τέλος αξίζει να σημειωθεί ότι ένα νευρωνικό δίκτυο ονομάζεται πλήρως διασυνδεδεμένο εάν κάθε κόμβος οποιουδήποτε στρώματος συνδέεται με όλους τους κόμβους του γειτονικού προς τα εμπρός στρώματος. Εάν αυτό δεν ισχύει δηλαδή εάν λείπουν μια ή περισσότερες συναπτικές συνδέσεις τότε το νευρωνικό δίκτυο καλείται ΝΔ μερικά διασυνδεδεμένο.

<u>ΝΔ Ανατροφοδότησης:</u> Εάν ένα ΝΔ περιέχει τουλάχιστον ένα βρόχο ανατροφοδότησης ο οποίος ανακυκλώνει πληροφορία μέσω του ιδίου ή προηγούμενων στρωμάτων τότε το ΝΔ καλείται αναδρομικό νευρωνικό δίκτυο.



Σχήμα 2.2.4.3: Πολυστρωματικό ΝΔ ανατροφοδότησης.

Το αποτέλεσμα της ανατροφοδότησης είναι ότι όταν ένα πρότυπο εισόδου εισέρχεται στο αναδρομικό ΝΔ, δεν παράγει ένα πρότυπο εξόδου σε πεπερασμένο

αριθμό χρονικών βημάτων, αλλά δρα με ένα κυκλικό τρόπο, όπου τα ίδια στρώματα ενεργοποιούνται επαναληπτικά. Εάν το ΝΔ είναι ευσταθές πιθανά να ταλαντωθεί για κάποιο χρονικό διάστημα, προτού φθάσει σε μια σταθερή κατάσταση στην οποία οι ενεργοποιήσεις των νευρώνων θα σταματήσουν να αλλάζουν με αποτέλεσμα να παραχθεί μια σταθερή έξοδος. Διαφορετικά εάν το ΝΔ είναι ασταθές οι ταλαντώσεις θα συνεχίσουν αδιάκοπα. Συνεπώς η εκπαίδευση του εν λόγω αναδρομικού ΝΔ σκοπό έχει την εύρεση των συναπτικών βαρών που του επιτρέπουν να σταθεροποιηθεί στις επιθυμητές τιμές εξόδου.

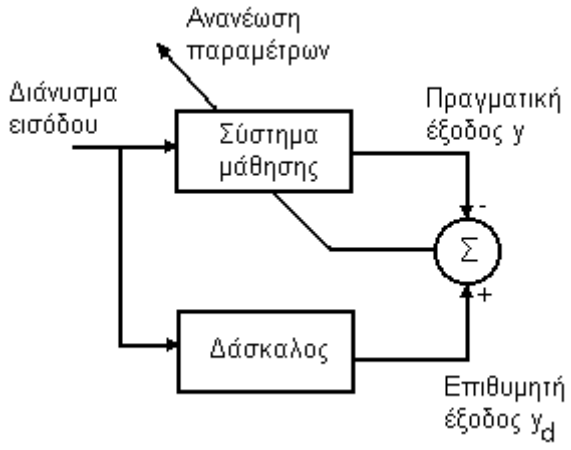
2.2.5 Μάθηση Νευρωνικών Δικτύων

Η ιδιότητα της προσαρμογής των ΝΔ σε μεταβαλλόμενους χώρους προβλημάτων σχετίζεται με την ικανότητα μάθησης τους. Η μάθηση είναι μια θεμελιακή ικανότητα των νευρωνικών δικτύων η οποία τους παρέχει την ικανότητα να μαθαίνουν από το περιβάλλον τους και να βελτιώνουν τη συμπεριφορά τους με το πέρασμα του χρόνου. Ειδικότερα στα ΝΔ η μάθηση αναφέρεται στην διεργασία επίτευξης μιας επιθυμητής συμπεριφοράς μέσω ενημέρωσης των τιμών των συναπτικών βαρών. Έτσι ένα ΝΔ μαθαίνει για το περιβάλλον του μέσω μιας επαναληπτικής διαδικασίας ανανέωσης των συναπτικών βαρών και κατωφλίων.

Σε γενικές γραμμές μπορεί να λεχθεί ότι αλγόριθμος μάθησης είναι κάθε προκαθορισμένο σύνολο καλά ορισμένων κανόνων επίλυσης του προβλήματος εκπαίδευσης του νευρωνικού δικτύου. Κάθε αλγόριθμος μάθησης προσφέρει έναν άλλο τρόπο προσαρμογής των συναπτικών βαρών. Γενικά υπάρχουν πολλοί αλγόριθμοι μάθησης στα ΝΔ, καθένας από τους οποίους παρουσιάζει πλεονεκτήματα αλλά και μειονεκτήματα. Τα προβλήματα μάθησης τα οποία επιλύουν οι αντίστοιχοι αλγόριθμοι εξαρτώνται και από το περιβάλλον στο οποίο εργάζεται κάθε ΝΔ. Έτσι διαφορετικά μοντέλα του περιβάλλοντος οδηγούν σε διαφορετικά μοντέλα εκπαίδευσης:

- Επιβλεπόμενη (ενεργή) μάθηση.
- Ενισχυτική μάθηση.
- Μη επιβλεπόμενη (αυτό-οργανούμενη) μάθηση.

Επιβλεπόμενη Μάθηση: Σχηματικά η δομή της επιβλεπόμενης μάθησης παρουσιάζεται στην συνέχεια. Παρατηρείται ότι στην επιβλεπόμενη μάθηση συνυπάρχουν δυο βασικές συνιστώσες, το σύστημα εκμάθησης και ο δάσκαλος.



Σχήμα 2.2.5.1: Δομή της επιβλεπόμενης μάθησης.

Το κύριο χαρακτηριστικό της επιβλεπόμενης μάθησης είναι η ύπαρξη του εξωτερικού δασκάλου ο οποίος με βάση την γνώση που είναι αποθηκευμένη σε αυτόν είναι σε θέση να διδάξει στο σύστημα μάθησης τις επιθυμητές εξόδους για το σύνολο εισόδων εκπαίδευσης. Όταν ο δάσκαλος και το ΝΔ λαμβάνουν ένα διάνυσμα εισόδου εκπαίδευσης, ο δάσκαλος δίνει στο νευρωνικό δίκτυο μια επιθυμητή έξοδο η οποία αναπαριστά την βέλτιστη δράση που πρέπει να εμφανίζει το ΝΔ. Οι παράμετροι του ΝΔ ανανεώνονται βάσει του διανύσματος εκπαίδευσης και του διανύσματος σφάλματος (δηλαδή της διαφοράς πραγματικής και επιθυμητής απόκρισης του δικτύου). Ορίζοντας μια συνάρτηση κόστους της μορφής:

$$J(w) = \frac{1}{2} E[(y(t)-y_d(t))^2]$$

όπου w είναι το διάνυσμα των προς επιλογή ελεύθερων παραμέτρων του συστήματος μάθησης (δηλαδή του ΝΔ), η ανανέωση των παραμέτρων παίρνει την μορφή του αλγορίθμου διόρθωσης σφάλματος. Η σταδιακή ανανέωση των παραμέτρων κάνει τελικά το ΝΔ μάθησης να μιμείται την δεδομένη επιθυμητή συμπεριφορά. Δύο περιπτώσεις αλγορίθμών επιβλεπόμενης μάθησης είναι ο αλγόριθμος ελαχίστου μέσου τετραγώνου και η γενίκευσή του που είναι γνωστή ως αλγόριθμος ανάστροφης διάδοσης. Αμφότεροι αυτοί οι αλγόριθμοι θα εξεταστούν στο επόμενο κεφάλαιο.

Ενισχυτική Μάθηση: Σε αυτή την περίπτωση το ΝΔ τροφοδοτείται και πάλι με δειγματικά πρότυπα εισόδου αλλά δεν τροφοδοτείται με τις επιθυμητές αποκρίσεις σε αυτές τις εισόδους. Σε αυτή την περίπτωση χρησιμοποιείται ένα συνολικό μέτρο της επάρκειας της προκύπτουσας απόκρισης το οποίο μπορεί να οδηγήσει το νευρωνικό δίκτυο στην επιθυμητή συμπεριφορά. Το μέτρο αυτό είναι γνωστό ως ενισχυτικό σήμα (reinforcement signal) και ανατροφοδοτείται στο ΝΔ έτσι ώστε να επιβραβεύσει τις ορθές συμπεριφορές και να τιμωρήσει τις λανθασμένες.

Η ενισχυτική μάθηση διακρίνεται σε συσχετιστική και μη συσχετιστική ενισχυτική μάθηση. Στην πρώτη περίπτωση το περιβάλλον τροφοδοτεί πέρα από το ενισχυτικό σήμα και άλλες μορφές πληροφορίας από τις οποίες το ΝΔ πρέπει να αποτυπώσει μια απεικόνιση συσχέτισης αιτίου-αποτελέσματος. Στην δεύτερη περίπτωση η μόνη πληροφορία που δίδεται από το περιβάλλον είναι το ενισχυτικό σήμα και ο προορισμός του ΝΔ είναι να επιλέξει μια μοναδική βέλτιστη ενέργεια. Επιγραμματικά η ενισχυτική μάθηση λειτουργεί ως εξής:

1] Το ΝΔ υπολογίζει τις εξόδους που παράγονται από την τρέχουσα είσοδο με τις παρούσες τιμές των βαρών.

2] Το σύστημα αξιολογεί την έξοδο και το ενισχυτικό σήμα τροφοδοτείται στο δίκτυο.

3] Τα βάρη ανανεώνονται με βάση το ενισχυτικό σήμα, αυξάνοντας τις τιμές των βαρών που συνέβαλλαν σε ορθή συμπεριφορά ή μειώνοντας τις τιμές των βαρών που προκάλεσαν αποκλίνουσα συμπεριφορά.

4] Το νευρωνικό δίκτυο ψάχνει να βρει ένα σύνολο βαρών τα οποία να τείνουν να αποφύγουν αρνητικά ενισχυτικά σήματα.

Η βασική διαφορά ανάμεσα στην ενισχυτική και την επιβλεπόμενη μάθηση είναι ότι στην ενισχυτική μάθηση το σύστημα μάθησης βελτιώνεται χρησιμοποιώντας ένα κριτήριο συμπεριφοράς οι τιμές του οποίου δίνονται από το περιβάλλον, ενώ στην επιβλεπόμενη μάθηση το κριτήριο συμπεριφοράς (συνάρτηση σφάλματος) καθορίζεται εσωτερικά με βάση τις επιθυμητές αποκρίσεις.

Μη Επιβλεπόμενη Μάθηση: Σε αυτό τον τύπο μάθησης που καλείται αυτό-οργανούμενη μάθηση δεν χρησιμοποιείται εξωτερικός δάσκαλος ούτε μια βάση γνώσης για να επιβλέψει την εκπαίδευση του ΝΔ, το μόνο στοιχείο που μπορεί να

χρησιμοποιηθεί για την υλοποίηση της εκπαίδευσης είναι τα διανύσματα εισόδου. Ένα σύστημα μη επιβλεπόμενης μάθησης εξελίσσεται με τέτοιο τρόπο ώστε να εξάγει χαρακτηριστικά ή κανονικότητες από τα παρουσιαζόμενα πρότυπα, χωρίς ωστόσο να έχει την πληροφορία για το ποιες έξοδοι ή ποιες κατηγορίες συσχετίζονται με τα χαρακτηριστικά εισόδου. Με άλλα λόγια το σύστημα μάθησης εντοπίζει ή κατηγοριοποιεί τα διανύσματα εισόδου χωρίς καμία εκ των προτέρων πληροφόρηση από το περιβάλλον. Εξ' αιτίας αυτών η μη επιβλεπόμενη μάθηση συχνά χρησιμοποιείται σε προβλήματα ομαδοποίησης, εξαγωγής εσωτερικών χαρακτηριστικών και ανίχνευσης συμμετριών.

Τα νευρωνικά δίκτυα μη επιβλεπόμενης μάθησης εκπαιδεύονται έτσι ώστε να αποκρίνονται σε διαφορετικά διανύσματα εισόδου με διαφορετικά τμήματα του δικτύου. Το ΝΔ εκπαιδεύεται με τέτοιο τρόπο ώστε να αυξάνει την πυροδότηση του ως απόκριση σε συχνά εμφανιζόμενες εισόδους, γι' αυτό και συχνά ονομάζεται εκτιμητής πιθανοτήτων (probability estimator). Κατά αυτό τον τρόπο το νευρωνικό δίκτυο αναπτύσσει συγκεκριμένες εσωτερικές αναπαραστάσεις οι οποίες κωδικοποιούν τα διάφορα διανύσματα εισόδου.

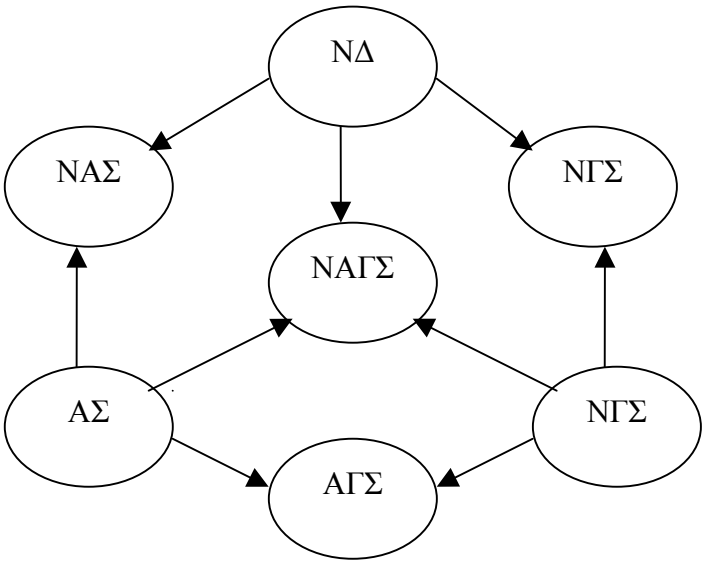
2.3 Νευρο-Ασαφή Συστήματα

2.3.1 Υβριδικά Συστήματα Υπολογιστικής Νοημοσύνης

Η Υπολογιστική Νοημοσύνη (ΥΝ) είναι η περιοχή της πληροφορικής, η οποία περιλαμβάνει υπολογισμούς και συλλογιστική υπό συνθήκες ανακρίβειας, αβεβαιότητας και μερικής αλήθειας και πετυχαίνει σταθερότητα και χαμηλό κόστος λύσεων. Τα τρία βασικά συστατικά πεδία της: Νευρωνικά Δίκτυα (ΝΔ), Ασαφής Συλλογιστική (ΑΣ) και Γεννητικοί Αλγόριθμοί (ΓΑ) έχουν το καθένα ιδιαίτερες ιδιότητες και πλεονεκτήματα [1]. Συγκεκριμένα:

- Τα ΝΔ επιτρέπουν στο σύστημα να μαθαίνει
- Η ΑΣ επιτρέπει την εμφύτευση στο σύστημα εμπειρικής γνώσης
- Οι ΓΑ καθιστούν το σύστημα ικανό να αυτο-βελτιώνεται

Συνδυάζοντας τα συστατικά αυτά πεδία μπορούμε να σχεδιάσουμε και να κατασκευάσουμε υβριδικά (μικτά) συστήματα ΥΝ ικανά να επιλύουν με υψηλή απόδοση πολύπλοκα πρακτικά προβλήματα. Τα υβριδικά συστήματα ΥΝ συνδυάζουν τις ιδιότητες καθενός πεδίου και παρακάμπτουν τους περιορισμούς ή τα μειονεκτήματα τους. Οι υβριδικές τεχνικές οδηγούν στην πραγματοποίηση ευφυών συστημάτων τα οποία βρίσκουν ποικίλες εφαρμογές. Οι τέσσερις δυνατοί συνδυασμοί των ΝΔ, ΑΣ και ΓΑ για την ανάπτυξη υβριδικών τεχνικών και συστημάτων ΥΝ φαίνονται στο σχήμα.



Σχήμα 2.3.1.1 Υβριδικά συστήματα υπολογιστικής νοημοσύνης

Επεξήγηση συμβολισμών *ΝΑΣ: Νευρο-ασαφή συστήματα*
ΝΓΣ: Νευρό-γενετικά συστήματα
ΑΓΣ: Ασαφο-γενετικά συστήματα
ΝΑΓΣ: Νευρο-ασαφο-γενετικά συστήματα

Τα ΝΔ χρησιμοποιούνται στα ΝΑΣ για να μάθουν τις συναρτήσεις συμμετοχής ή / και να καθορίζουν τη δομή των ασαφών συστημάτων. Οι ΓΑ

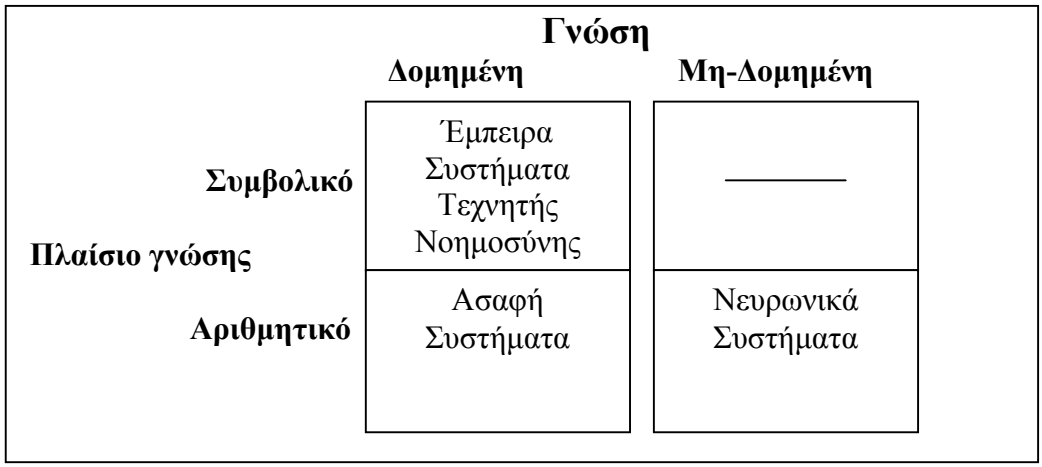
χρησιμοποιούνται στα ΑΓΣ για την αναζήτηση μιας βέλτιστης δομής και ακολούθως για τη ρύθμιση των παραμέτρων. Αντίστροφα, η ασαφής λογική μπορεί να χρησιμοποιηθεί για τη βελτίωση της συμπεριφοράς των ΓΑ. Οι γενετικοί αλγόριθμοι χρησιμοποιούνται στα ΝΓΣ για την αυτοματοποίηση της σχεδίασης νευρωνικών δικτύων μέσω της γενετικής εκπαίδευσης ή μέσω της γενετικής επιλογής της τοπολογίας αυτών ή μέσω της βέλτιστης επιλογής των παραμέτρων μάθησης. Τέλος συνδυάζοντας κατά ποικίλους τρόπους τα ΝΔ, την ΑΣ και τους ΓΑ μπορούμε να σχεδιάσουμε νευρο-ασαφή-γενετικά συστήματα (ΝΑΓΣ) τα οποία ολοκληρώνουν τα πλεονεκτήματα και τις ιδιότητες όλων αυτών με στόχο τη βελτιστοποίηση της απόδοσης τους.

2.3.2 Νευρο-Ασαφή Συστήματα

Τα καθαρά συστήματα ασαφούς συλλογιστικής έχουν τα εξής δύο μειονεκτήματα:

- Δεν διαθέτουν μια συγκεκριμένη μέθοδο για τον προσδιορισμό των συναρτήσεων συμμετοχής
- Δεν διαθέτουν μια συνιστώσα μάθησης ή προσαρμοστικότητας.

Τα παραπάνω μειονεκτήματα εξαλείφονται αν χρησιμοποιηθούν νευρωνικά δίκτυα για την καθοδήγηση της ασαφούς συλλογιστικής. Πραγματικά τα ΝΔ μπορούν να εκπαιδευθούν να επιλέγουν τις συναρτήσεις συμμετοχής (δηλαδή τα ασαφή σύνολα) κατά αυτόματο τρόπο, όπως επίσης και να επιλέγουν τον αριθμό ή / και τη μορφή των ασαφών κανόνων. Για το σκοπό αυτό έχουν αναπτυχθεί ποικίλες τεχνικές με σχετικές διακυμάνσεις στη γενικότητα, απλότητα και εφαρμοστικότητα τους. Ιδιαίτερη σημασία έχει το γεγονός ότι τόσο τα ΝΔ όσο και η ΑΣ χαρακτηρίζονται από αυξημένο βαθμό παραλληλίας.

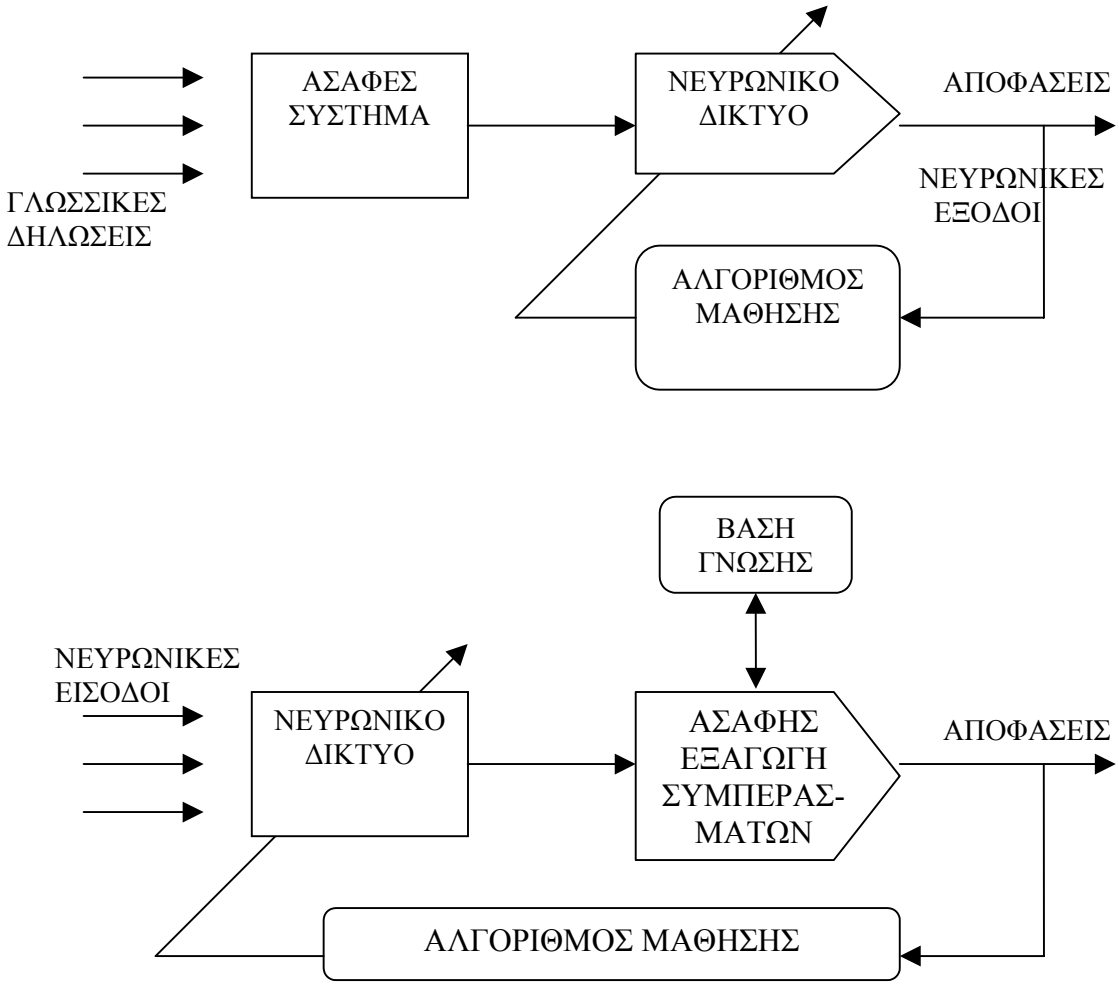


Σχήμα 2.3.2.1 Ταξινόμηση των έμπειρων, ασαφών και νευρωνικών συστημάτων

Στο σχήμα φαίνεται η θέση των ασαφών και των νευρωνικών συστημάτων σύμφωνα με την ταξινόμηση ως προς το πλαίσιο της γνώσης (συμβολικό, αριθμητικό) και του τύπου της (δομημένου ή μη), από το οποίο προκύπτει η μεγάλη ποικιλία συνδυασμών ασαφών και νευρωνικών συστημάτων .

Στην υλοποίηση των ασαφών συστημάτων τα ΝΔ μπορούν να χρησιμοποιηθούν στους εξής τομείς:

1. Υπολογισμός των συναρτήσεων συμμετοχής
2. Ασαφοποίηση των εισόδων
3. Υλοποίηση συναρτήσεων συμμετοχής
4. Συνδυασμός συναρτήσεων συμμετοχής
5. Αποασαφοποίηση των ασαφών ποσοτήτων ώστε να έχουμε αριθμητικές εξόδους



Σχήμα 2.3.2.2 Διάφορες υλοποιήσεις ασαφών νευρωνικών δικτύων

2.3.3 Νευρωνικά Μέρη ενός Ασαφούς Συστήματος

Τα ασαφή συστήματα μπορούν να υλοποιηθούν χρησιμοποιώντας πολυεπίπεδα δίκτυα πρόσθιας τροφοδότησης όπου κάθε επίπεδο υλοποιεί τους υπολογισμούς που απαιτούνται σε κάθε στάδιο ενός ασαφούς συστήματος.

1. Το πρώτο επίπεδο υπολογίζει τις συναρτήσεις συμμετοχής (ασαφοποίηση)
2. Το δεύτερο επίπεδο υλοποιεί τους ασαφείς κανόνες και συνδυάζει τους ασαφείς κανόνες χρησιμοποιώντας την συνάρτηση ελαχίστου (ή μια προσέγγιση της)

3. Το τρίτο επίπεδο συνδυάζει τις ασαφείς τιμές που προκύπτουν χρησιμοποιώντας τη συνάρτηση μεγίστου
4. Το επίπεδο εξόδου υλοποιεί την απο-ασαφοποίηση.

Υπάρχουν και άλλες παραλλαγές. Στόχος της απεικόνισης ενός ασαφούς συστήματος σε ένα ΝΔ είναι η χρήση αλγορίθμων εκπαίδευσης για τον καθορισμό των παραμέτρων συμμετοχής με χρήση δεδομένων εκπαίδευσης. Όμως οι συναρτήσεις ελαχίστου και μεγίστου που χρησιμοποιούνται από ένα ασαφές σύστημα δεν είναι παραγωγίσιμες. Μια λύση που ακολουθείται συχνά είναι η προσέγγιση αυτών των συναρτήσεων με παραγωγίσιμες συναρτήσεις.

Κεφάλαιο 3

Ταξινομητές Προτύπων

3 Ταξινομητές Προτύπων

3.1 Νευρωνικά Δίκτυα Ταξινόμησης

3.1.1 Το Πρόβλημα της Ταξινόμησης Προτύπων

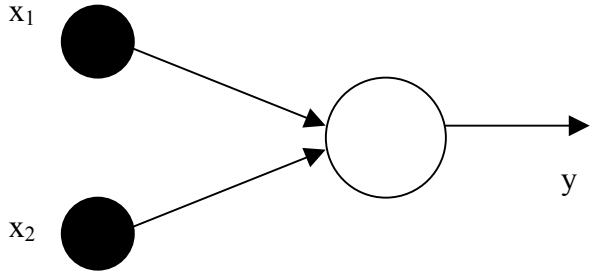
Το πρόβλημα της ταξινόμησης προτύπων είναι να ταξινομηθεί ένας δεδομένος αριθμός εισόδων (προτύπων, ερεθισμών, ενεργοποιήσεων) σε ένα σταθερό σύνολο δεδομένων κατηγοριών (κλάσεων). Το πρόβλημα μπορεί να λυθεί τόσο με επιβλεπόμενη όσο και με μη-επιβλεπόμενη μάθηση. Στην πρώτη περίπτωση το νευρωνικό δίκτυο εκπαιδεύεται με ένα σύνολο ζευγών «πρότυπο εισόδου-κατηγορία» και ακολούθως καλείται να ταξινομήσει πρότυπα (διανύσματα) που δεν έχει δει προηγουμένως (με την υπόθεση βεβαίως ότι τα πρότυπα ανήκουν στον ίδιο πληθυσμό προτύπων που χρησιμοποιήθηκαν για την εκπαίδευση του δικτύου). Μη επιβλεπόμενη μάθηση μπορεί να χρησιμοποιηθεί όταν δεν διατίθεται προγενέστερη (a-priori) γνώση των κατηγοριών στις οποίες πρόκειται να ταξινομηθούν τα πρότυπα (διανύσματα) εξόδου, οπότε και έχουμε πρόβλημα ομαδοποίησης (clustering) [1].

3.1.2 Το Απλό Perceptron

Το απλό perceptron είναι το απλούστερο νευρωνικό δίκτυο που χρησιμοποιείται για την ταξινόμηση γραμμικά διαχωρίσιμων προτύπων δηλαδή προτύπων τα οποία διαχωρίζονται από ένα υπερεπίπεδο. Αποτελείται από ένα απλό νευρώνιο με προσαρμόσιμα βάρη το οποίο ακολουθείται από μια διπολική συνάρτηση ενεργοποίησης της μορφής:

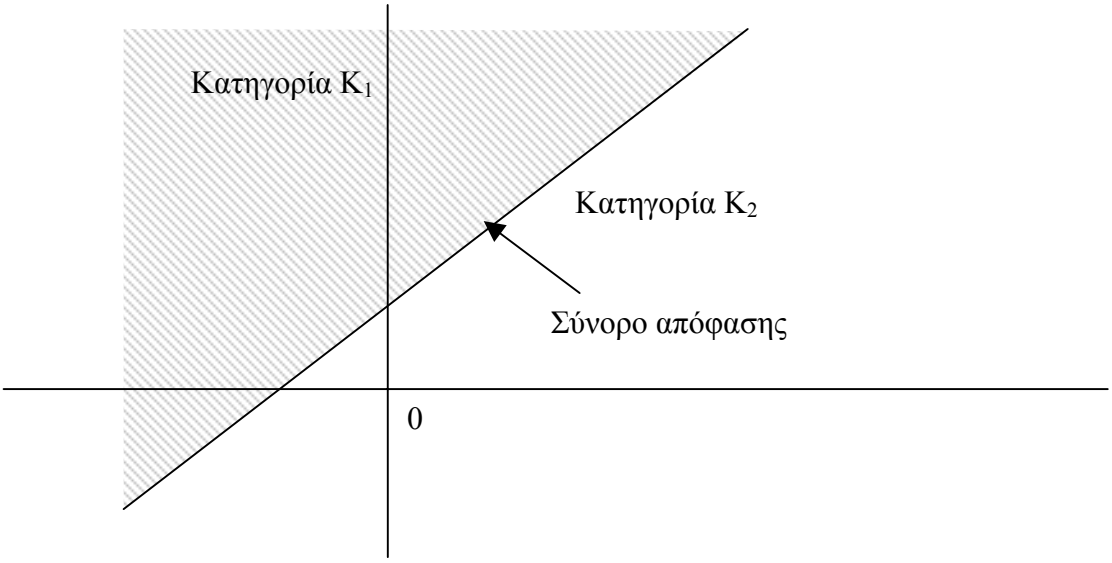
$$f(u)=\begin{cases} -1 , & u<0 \\ 0 , & u=0 \\ +1 , & u>0 \end{cases}$$

Αυτό μπορεί να πραγματοποιήσει ταξινόμηση στην περίπτωση που έχουμε μόνο δύο κατηγορίες. Για να είναι δυνατή η ταξινόμηση περισσότερων γραμμικά διαχωρίσιμων κατηγοριών (κλάσεων) το στρώμα εξόδου χρειάζεται να έχει περισσότερα από ένα νευρώνια.



Σχήμα *3.1.2.1 Απλό perceptron*

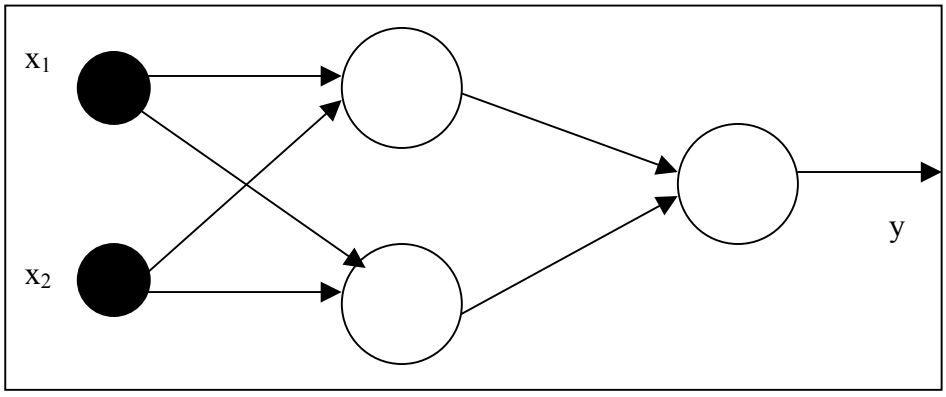
Ο σκοπός του perceptron είναι να ταξινομήσει τα εξωτερικά πρότυπα σε μια από τις δύο κατηγορίες K_1 ή K_2. Για να κατανοήσουμε πως λειτουργεί ένας ταξινομητής προτύπων θεωρούμε την περίπτωση δύο μεταβλητών x_1, x_2 όπου το σύνορο απόφασης (διαχωρισμού) ανάμεσα στις κατηγορίες K_1 και K_2 είναι μια ευθεία γραμμή. Οποιοδήποτε σημείο βρίσκεται πάνω από την διαχωριστική γραμμή ταξινομείται στην κατηγορία K_1 διαφορετικά αποδίδεται στην K_2 [1].



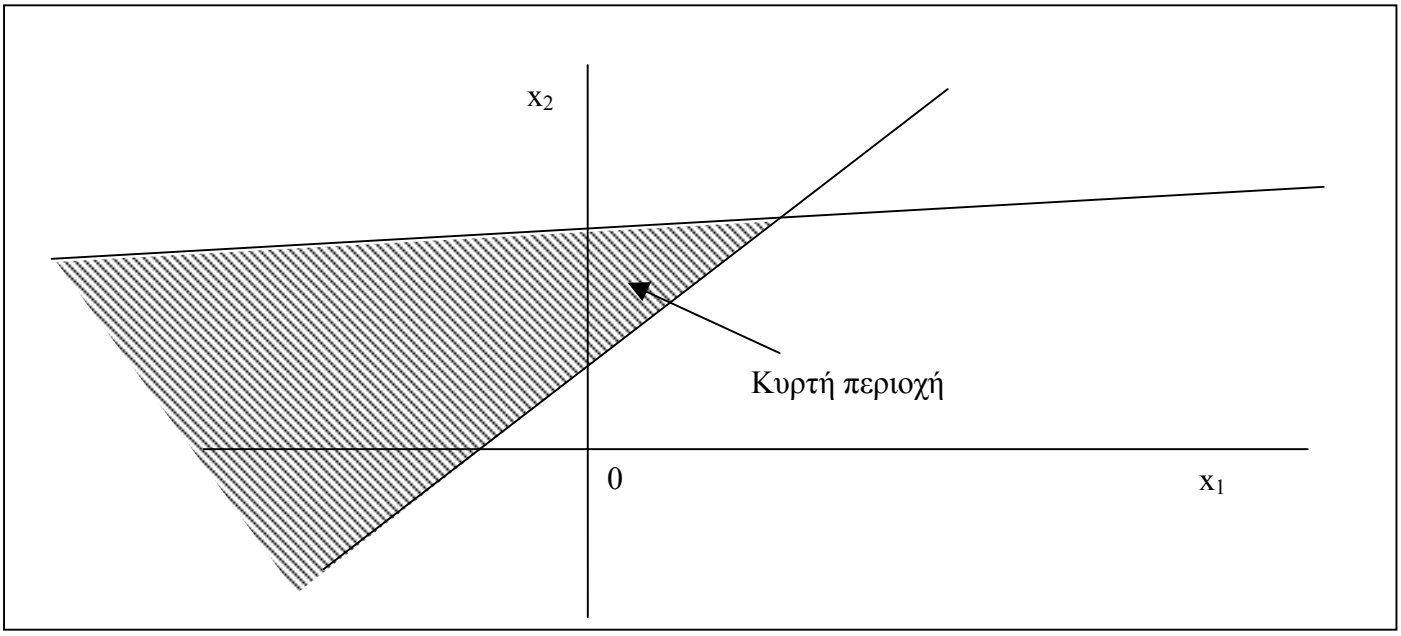
Σχήμα 3.1.2.2 Γραμμικός διαχωρισμός δύο κλάσεων K_1, K_2

3.1.3 Πολυεπίπεδο Perceptron

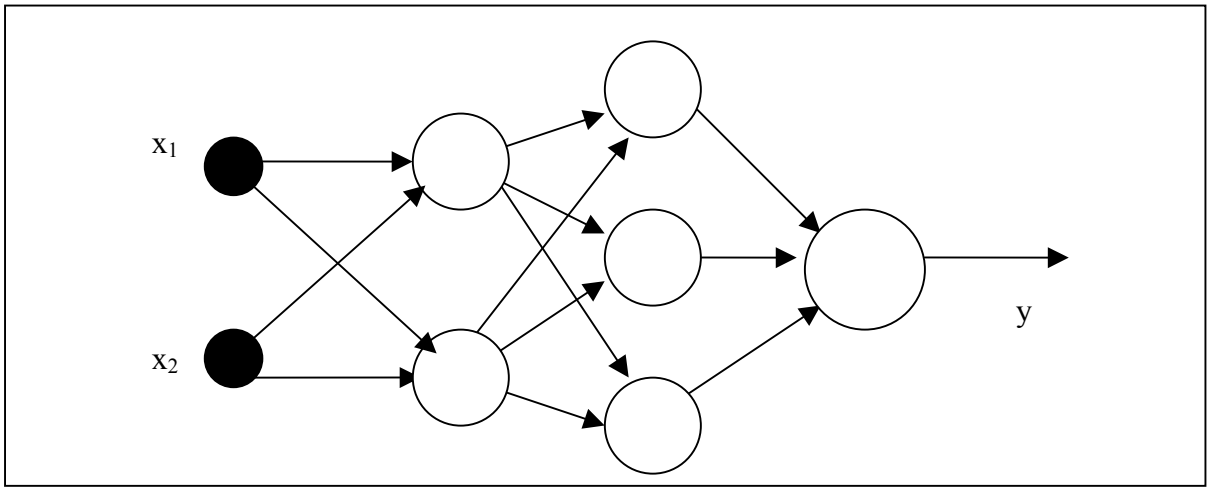
Ένα Νευρωνικό Δίκτυο με δύο κρυμμένα στρώματα μπορεί να διακρίνει (ταξινομήσει) περιοχές οποιουδήποτε αυθαίρετου σχήματος. Πραγματικά το πρώτο κρυμμένο στρώμα μπορεί να παράγει γραμμές (επίπεδα) ταξινόμησης στο χώρο προτύπων, οι οποίες δεν μπορούν να υπερβούν το πλήθος των κόμβων του. Οι γραμμές αυτές συνδυάζονται από τα νευρόνια εξόδου και παράγουν κυρτές περιοχές. Προσθέτοντας ένα ακόμα κρυμμένο στρώμα κόμβων υπολογισμού τούτο συνδυάζει τις γραμμές (επίπεδα) που δέχεται από τους κόμβους του πρώτου κρυμμένου στρώματος και στέλνει κυρτές περιοχές στα νευρόνια του στρώματος εξόδου τα οποία παράγουν περιοχές στο χώρο προτύπων οποιουδήποτε σχήματος. Η πολυπλοκότητα του σχήματος των περιοχών αυτών περιορίζεται μόνο από τον αριθμό των κόμβων του δικτύου [1].



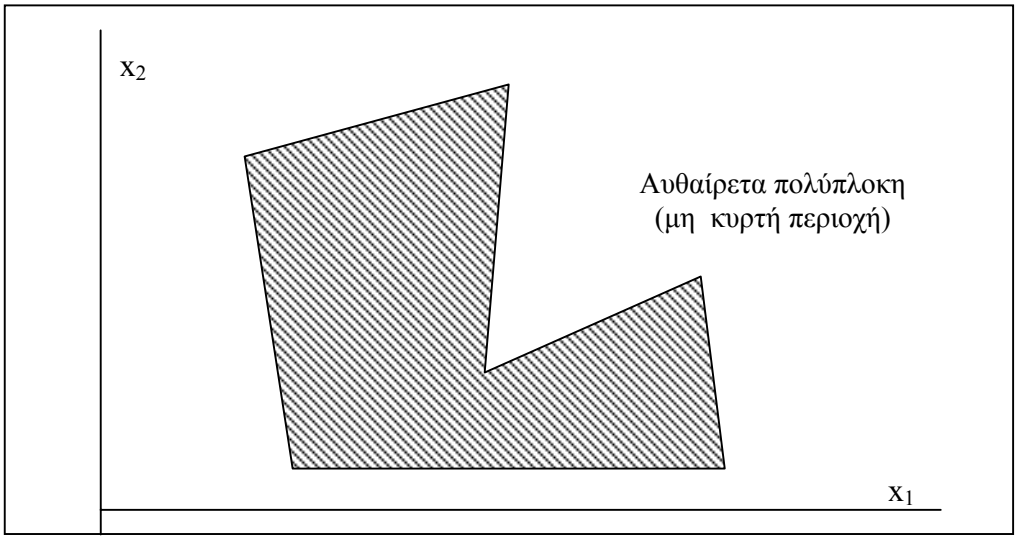
Σχήμα *3.1.3.1 Perceptron με ένα κρυμμένο στρώμα*



Σχήμα *3.1.3.2 Γραμμικός διαχωρισμός για Perceptron με ένα κρυμμένο στρώμα*



Σχήμα *3.1.3.3 Perceptron με δύο κρυμμένα στρώματα*



Σχήμα *3.1.3.4 Γραμμικός διαχωρισμός Perceptron με δύο κρυμμένα στρώματα*

Προκειμένου να εκπαιδευτεί το πολυεπίπεδο Perceptron απαιτείται μια διαδικασία κωδικοποίησης των κατηγοριών. Σκοπός της διαδικασίας αυτής είναι η μετατροπή του προβλήματος ταξινόμησης σε πρόβλημα απεικόνισης μέσω της αντιστοίχισης κάθε κατηγορίας σε κάποιο διάνυσμα εξόδου. **Απεικόνιση** *είναι η διαδικασία αντιστοίχισης ενός διανύσματος πραγματικών εισόδων σε ένα διάνυσμα πραγματικών εξόδων*. Με τον τρόπο αυτό το αρχικό σύνολο εκπαίδευσης μετασχηματίζεται ώστε να περιέχει ζεύγη της μορφής (πρότυπο, διάνυσμα εξόδου) και να μπορεί να χρησιμοποιηθεί το πολυεπίπεδο Perceptron για την υλοποίηση της απεικόνισης.

Ο ευρύτερα χρησιμοποιούμενος τρόπος κωδικοποίησης των κατηγοριών είναι η κωδικοποίηση 1-από–K (1-out of-K) για ένα πρόβλημα κατηγοριών. Στην κωδικοποίηση αυτή, το διάνυσμα εξόδου έχει συνιστώσες $(t_1,t_2,\ldots,t_k)$ και η κατηγορία C_k κωδικοποιείται θέτοντας $t_k=1$ και $t_i=0$ $i\neq k$. Η ταξινόμηση ενός προτύπου γίνεται εφαρμόζοντας το πρότυπο ως είσοδο στο δίκτυο και επιλέγοντας την κατηγορία που αντιστοιχεί στην έξοδο με την μεγαλύτερη τιμή . Όσο πιο κοντά στο 1 είναι η έξοδος και κοντά στο 0 οι υπόλοιπες έξοδοι, τόσο πιο αξιόπιστη είναι η ταξινόμηση. Ειδικά για την περίπτωση των δύο κατηγοριών χρησιμοποιείται και η κωδικοποίηση με μια έξοδο: αντιστοιχίζουμε την έξοδο t=0 στην μια κατηγορία (C_1) και την έξοδο t=1 στην άλλη κατηγορία (C_2). Στην περίπτωση αυτή, η ταξινόμηση ενός προτύπου (αφού γίνει η εκπαίδευση) γίνεται ως εξής: αν η έξοδος είναι μεγαλύτερη από 0.5 τότε το πρότυπο ταξινομείται στην κατηγορία (C_2) αλλιώς στην κατηγορία (C_1) [2].

3.1.4 Μάθηση και Γενίκευση

Στην περίπτωση που χρησιμοποιούμε τα νευρωνικά δίκτυα ταξινόμησης υπάρχει το πρόβλημα της επιλογής του αριθμού των κρυμμένων μονάδων, που πρέπει να χρησιμοποιηθούν. Στόχος της εκπαίδευσης δεν είναι η ακριβής μάθηση ολόκληρου του συνόλου εκπαίδευσης, αλλά η επίδοση του μοντέλου στην ταξινόμηση νέων δεδομένων, δηλαδή η ικανότητα γενίκευσης.

Χαρακτηριστικό παράδειγμα αποτελούν τα πολυεπίπεδα Perceptron. Χρησιμοποιώντας μεγάλο αριθμό κρυμμένων μονάδων, μπορούν να μάθουν πλήρως όλα τα δεδομένα ενός συνόλου εκπαίδευσης. Ωστόσο ένας τέτοιος ταξινομητής έχει γενικά κακές ικανότητες γενίκευσης, διότι στην ουσία απομνημονεύει τα δεδομένα εκπαίδευσης και δεν παρουσιάζει καλές επιδόσεις. Από την άλλη ένα πολυεπίπεδο Perceptron με λίγες κρυμμένες μονάδες δεν έχει αρκετή ευελιξία ώστε να μπορεί να ορίσει πολύπλοκες περιοχές απόφασης. Υπάρχει λοιπόν ένας βέλτιστος αριθμός παραμέτρων (βαρών και πολώσεων) ενός δικτύου για το οποίο το εκπαιδευμένο μοντέλο χαρακτηρίζεται από τις καλύτερες επιδόσεις γενίκευσης.

Υπάρχει λοιπόν η ανάγκη τεχνικών εύρεσης της βέλτιστης πολυπλοκότητας ενός δικτύου για την οποία προκύπτει η καλύτερη γενικευτική ικανότητα. Ισχύει το δίλημμα πόλωσης-μεταβλητότητας σύμφωνα με το οποίο το σφάλμα γενίκευσης ενός ταξινομητή μπορεί να γραφτεί σαν άθροισμα δύο παραγόντων: της πόλωσης και της μεταβλητότητας. Ένα μοντέλο που είναι πολύ απλό έχει μεγάλη πόλωση, ενώ ένα μοντέλο με πολλές παραμέτρους έχει μεγάλη μεταβλητότητα. Η καλύτερη γενίκευση προκύπτει όταν έχουμε το βέλτιστο συνδυασμό τιμών των δύο παραπάνω ποσοτήτων.

Ουσιαστικά το δίλημμα πόλωσης-μεταβλητότητας μας λέει ότι στην περίπτωση που προσπαθούμε να ελαττώσουμε την πόλωση ενός ταξινομητή, δηλαδή να απομνημονεύσουμε ολόκληρο το σύνολο εκπαίδευσης χρησιμοποιώντας ένα πολύ ευέλικτο μοντέλο το τίμημα που πληρώνουμε είναι ότι αυξάνουμε τη μεταβλητότητα

και κατά συνέπεια μειώνουμε την ικανότητα γενίκευσης του μοντέλου σε άγνωστα δεδομένα [2].

3.1.5 Δομική Προσέγγιση

Για να πετύχουμε τη βέλτιστη ισορροπία μεταξύ της πόλωσης και μεταβλητότητας συνήθως χρησιμοποιούμε την μέθοδο της *δομικής προσέγγισης* κατά την οποία ξεκινάμε από ένα μικρό δίκτυο (λίγες παράμετροι) το οποίο βαθμιαία επεκτείνουμε (αυξάνουμε τον αριθμό των κρυμμένων μονάδων Μ) μέχρι να πετύχουμε βέλτιστες επιδόσεις. Συγκεκριμένα θα παρατηρήσουμε ότι υπάρχει κάποια τιμή του Μ πέρα από την οποία το σφάλμα γενίκευσης αρχίζει να αυξάνεται. Το φαινόμενο αυτό λέγεται ***υπερεκπαίδευση*** του δικτύου και σημαίνει ότι ο ταξινομητής έχει πολύ μεγάλο αριθμό παραμέτρων και στην ουσία έχει απομνημονεύσει τα δεδομένα, μειώνοντας έτσι τη γενικευτική ικανότητα. Από τη στιγμή που παρατηρούμε ελάττωση της γενικευτικής ικανότητας σταματάμε να αυξάνουμε των αριθμό των παραμέτρων και θεωρούμε ότι φτάσαμε σε βέλτιστο μοντέλο.

Η παραπάνω διαδικασία μπορεί να εφαρμοστεί και με αντίστροφη φορά: εκπαιδεύουμε αρχικά ένα μοντέλο με μεγάλο αριθμό παραμέτρων. Το μοντέλο αυτό λόγω μεγάλης μεταβλητότητας έχει μικρή πόλωση και μικρή γενικευτική ικανότητα. Σταδιακά μειώνουμε τον αριθμό των κρυμμένων μονάδων οπότε μειώνεται το σφάλμα γενίκευσης, μέχρι να φτάσουμε σε κάποιο μέγεθος δικτύου πέρα από το οποίο το σφάλμα γενίκευσης αρχίζει να αυξάνει, οπότε σταματάμε την όλη διαδικασία [2].

3.1.6 Τεχνικές Εκτίμησης Σφάλματος Ταξινόμησης

Από τη στιγμή που η αξιολόγηση ενός ταξινομητή θα πρέπει να γίνεται με βάση την επίδοση του στην ταξινόμηση άγνωστων δεδομένων, η στρατηγική που χρησιμοποιείται για την εκτίμηση του σφάλματος ταξινόμησης κάποιας μεθόδου διαιρεί το σύνολο των διαθέσιμων προτύπων σε δύο τμήματα: στο **σύνολο εκπαίδευσης (training set)** που χρησιμοποιείται για την κατασκευή του ταξινομητή και στο **σύνολο ελέγχου (testing set)** που χρησιμοποιείται για τον υπολογισμό του σφάλματος γενίκευσης. Οι τεχνικές εκτίμησης σφάλματος διαφέρουν μεταξύ τους κυρίως στον τρόπο που γίνεται η διάσπαση των δεδομένων στα δύο σύνολα. Θα πρέπει να σημειωθεί ότι οι τεχνικές αυτές δεν μπορούν να χρησιμοποιηθούν για την αξιολόγηση ενός συγκεκριμένου ταξινομητή (ενός συγκεκριμένου συνόλου παραμέτρων), διότι βασίζονται στην κατασκευή πολλών ταξινομητών (πολλών συνόλων παραμέτρων) για την εκτίμηση σφάλματος. Οι λόγοι για τους οποίους χρησιμοποιούνται είναι:

a) για τη σύγκριση διαφορετικών τεχνικών
b) για τη μελέτη της επίδρασης των διαφόρων χαρακτηριστικών εισόδου στο σφάλμα ταξινόμησης
c) *για την μελέτη της επίδρασης του αριθμού των παραμέτρων στο σφάλμα γενίκευσης, όταν χρησιμοποιείται συγκεκριμένο μοντέλο*.

Οι κυριότερες *τεχνικές εκτίμησης σφάλματος* είναι οι ακόλουθες:

- **Holdout**: Καθορίζεται το ποσοστό των προτύπων εκπαίδευσης και ελέγχου. Δημιουργούνται τυχαία τα σύνολα εκπαίδευσης και ελέγχου με βάση τα παραπάνω ποσοστά και κατασκευάζεται ο ταξινομητής χρησιμοποιώντας το σύνολο εκπαίδευσης. Στη συνέχεια υπολογίζεται το σφάλμα ταξινόμησης χρησιμοποιώντας το σύνολο ελέγχου. Επαναλαμβάνεται αρκετές φορές η παραπάνω διαδικασία (δημιουργία συνόλων εκπαίδευσης και ελέγχου, εκπαίδευση του ταξινομητή και υπολογισμός του σφάλματος ταξινόμησης) και η τελική εκτίμηση σφάλματος προκύπτει ως ο μέσος όρος των επιμέρους σφαλμάτων ταξινόμησης που υπολογίστηκαν.

- **Leave-one-out:** Για κάθε διαθέσιμο πρότυπο κατασκευάζεται ένας ταξινομητής, θεωρώντας ως σύνολο εκπαίδευσης ολόκληρο το σύνολο δεδομένων εκτός από το συγκεκριμένο πρότυπο. Στη συνέχεια ελέγχεται η δυνατότητα του ταξινομητή να ταξινομεί σωστά το πρότυπο που αγνοήθηκε κατά την εκπαίδευση, και αυτή η διαδικασία επαναλαμβάνεται για όλα τα διαθέσιμα πρότυπα, οπότε τελικά μετράται το ποσοστό των προτύπων που ταξινομήθηκαν λάθος.

- **Rotation**: Διαιρείται το σύνολο προτύπων σε υποσύνολα και για κάθε υποσύνολο κατασκευάζεται ένας ταξινομητής θεωρώντας ως σύνολο εκπαίδευσης τα πρότυπα των υπολοίπων υποσυνόλων, οπότε και υπολογίζεται το ποσοστό των σφαλμάτων ταξινόμησης χρησιμοποιώντας ως σύνολο ελέγχου τα πρότυπα του υποσυνόλου. Η διαδικασία επαναλαμβάνεται ίσες φορές με τον αριθμό των υποσυνόλων και υπολογίζεται το τελικό σφάλμα εκτίμησης ως μέσος όρος των επιμέρους σφαλμάτων.

- **Bootstrap**: Επιλέγουμε με τυχαίο τρόπο N ακέραιους αριθμούς από 1 έως N, όπου N ο αριθμός των προτύπων του συνόλου δεδομένων. Είναι πιθανό μερικοί αριθμοί να επιλεγούν περισσότερες από μία φορές ενώ άλλοι καθόλου. Τα πρότυπα που αντιστοιχούν στους αριθμούς που δεν επιλέχθηκαν αποτελούν το σύνολο ελέγχου και τα υπόλοιπα στο σύνολο εκπαίδευσης. Κατασκευάζεται ο αντίστοιχος ταξινομητής και υπολογίζεται το σφάλμα ταξινόμησης. Επαναλαμβάνεται την παραπάνω διαδικασία αρκετές φορές και υπολογίζεται το τελικό σφάλμα ταξινόμησης ως μέσος όρος των επιμέρους σφαλμάτων [2].

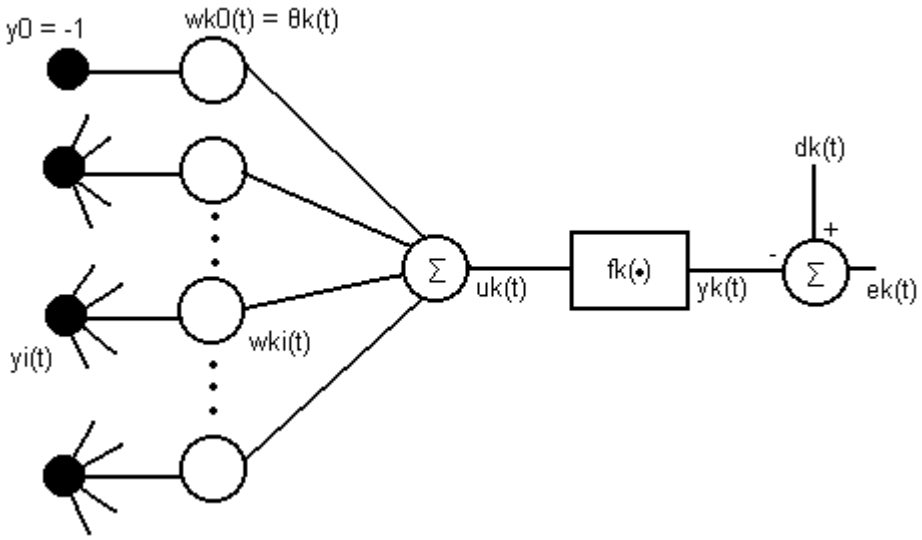
3.2 Ο Αλγόριθμος Ανάστροφης Διάδοσης

3.2.1 Περιγραφή του Αλγορίθμου Ανάστροφης Διάδοσης

Ο αλγόριθμος *ανάστροφης διάδοσης* (*Backpropagation*) είναι γενίκευση του αλγορίθμου *μέσου ελαχίστου τετραγώνου* (*Least Mean Square*) και αποτελεί ίσως τον πιο διαδεδομένο αλγόριθμο επιβλεπόμενης μάθησης [1], [2]. Ως τέτοιος προσαρμόζει τα συναπτικά βάρη έτσι ώστε να ελαχιστοποιηθεί το μέσο τετραγωνικό σφάλμα μεταξύ των επιθυμητών και πραγματικών αποκρίσεων μετά από την παρουσίαση του κάθε προτύπου στην είσοδο του νευρωνικού δικτύου. Η επέκταση από τον αλγόριθμο μέσου ελαχίστου τετραγώνου συνίσταται στην χρήση της πραγματικής εξόδου κάθε νευρονίου μετά την εφαρμογή της συνεχούς συνάρτησης ενεργοποίησης. Έτσι, τα συναπτικά βάρη ενημερώνονται με στόχο την ελαχιστοποίηση του κριτηρίου:

$$E_p(t) = \frac{1}{2}\sum_k e^2{}_k(t) \ , \ e_k(t) = d_k(t)\text{-}y_k(t) \qquad (3.2.1.1)$$

όπου η άθροιση εκτείνεται σε όλα τα νευρόνια του στρώματος εξόδου μετά την παρουσίαση κάθε διανύσματος εκπαίδευσης στην είσοδο του ΝΔ. Αυτό σημαίνει ότι η ελαχιστοποίηση πρέπει να γίνει διαδοχικά καθώς διαφορετικά πρότυπα εκπαίδευσης εφαρμόζονται στις εισόδους του νευρωνικού δικτύου.



Σχήμα 3.2.1.1: Το k-οστό νευρόνιο εξόδου.

Το σήμα $u_k(t)$ της εσωτερικής δραστηριότητας του νευρονίου είναι:

$$u_k = \sum_{i=0}^{n} w_{ki}(t)y_i(t) \qquad (3.2.1.2)$$

$$y_k(t) = f_k(u_k(t)) \qquad (3.2.1.3)$$

$$e_k(t) = d_k(t)\text{-}f_k(u_k(t)) \qquad (3.2.1.4)$$

βάσει του τύπου της απότομης κατάβασης οι διορθώσεις των βαρών $\Delta w_{ki}(t)$ είναι:

$$\Delta w_{ki}(t) = -\gamma\frac{\partial E_p(t)}{\partial w_{ki}(t)} \qquad (3.2.1.5)$$

όπου γ είναι η παράμετρος μάθησης και η μερική παράγωγος $\partial E_p(t)/\partial w_{ki}(t)$ δίδεται από την σχέση:

$$\frac{\partial E_p(t)}{\partial w_{ki}(t)} = \frac{\partial E_p(t)}{\partial e_k(t)} \cdot \frac{\partial e_k(t)}{\partial y_k(t)} \cdot \frac{\partial y_k(t)}{\partial u_k(t)} \cdot \frac{\partial u_k(t)}{\partial w_{ki}(t)}$$

$$= e_k(t)(-1)f'_k(u_k(t))y_i(t) \qquad (3.2.1.6)$$

όπου $f'_k(u_k(t)) = df_k(u_k(t))/du_k(t)$

επομένως τελικά προκύπτει ο γενικευμένος κανόνας δέλτα:

$\Delta w_{ki}(t) = \gamma\delta_k(t)y_i(t)$ (3.2.1.7)

όπου $\delta_k(t) = e_k(t)f'_k(u_k(t))$

Τούτο το γινόμενο το οποίο ονομάζεται τοπική κλίση μπορεί να υπολογιστεί για όλους τους νευρώνες του στρώματος εξόδου αλλά και για τους νευρώνες των κρυμμένων στρωμάτων.

<u>Στρώμα Εξόδου:</u> Σε αυτό το επίπεδο θα ισχύει:

$\Delta w_{ki}(t) = \gamma[d_k(t)-y_k(t)]f'_k(u_k(t))$ (3.2.1.8)

όπου ο δείκτης k εκτείνεται σε όλα τα νευρόνια εξόδου και ο δείκτης i σε όλα τα νευρόνια του τελευταίου κρυμμένου στρώματος.

<u>Κρυμμένα Στρώματα:</u> Για τους νευρώνες j των κρυμμένων στρωμάτων δεν υπάρχουν συγκεκριμένες επιθυμητές αποκρίσεις. Συνεπώς τα αντίστοιχα σφάλματα πρέπει να υπολογιστούν έμμεσα χρησιμοποιώντας τα σφάλματα όλων των νευρονίων με τα οποία κάθε κρυμμένο νευρόνιο συνδέεται. Η τοπική κλίση $\delta_j(t)$ είναι ίση με:

$$\delta_j(t) = -\frac{\partial E_p(t)}{\partial y_j(t)} \cdot \frac{\partial y_j(t)}{\partial u_j(t)} = \frac{\partial E_p(t)}{\partial y_j(t)} f'_j(u_j(t)) \qquad (3.2.1.9)$$

όπου

$$\frac{\partial E_p(t)}{\partial y_j(t)} = \sum_m e_m(t) \frac{\partial e_m(t)}{\partial y_j(t)}$$

$$= \sum_m e_m(t) \frac{\partial e_m(t)}{\partial u_m(t)} \frac{\partial u_m(t)}{\partial y_j(t)}$$

$$= \sum_m e_m(t) f'_m(u_m(t)) w_{mj}(t) \qquad (3.2.1.10)$$

και ο δείκτης m εκτείνεται σε όλα τα νευρόνια του στρώματος εξόδου. Η σχέση (3.2.1.10) προκύπτει ομοίως με προηγούμενα (σχέσεις (3.2.1.1) → (3.2.1.4)):

$$E_p(t) = \frac{1}{2}\sum_m e^2{}_m(t) \ , \ e_m(t) = d_m(t)-y_m(t) \qquad (3.2.1.11)$$

$$u_m = \sum_{j=0}^{n} w_{mj}(t) y_j(t) \qquad (3.2.1.12)$$

$$y_m(t) = f_m(u_m(t)) \qquad (3.2.1.13)$$

όπου n είναι ο συνολικός αριθμός εισόδων. Βάσει της (3.2.1.7) η (3.2.1.10) γράφεται:

$$\frac{\partial E_p(t)}{\partial y_j(t)} = -\sum_m \delta_m(t) w_{mj}(t) \qquad (3.2.1.14)$$

άρα το δέλτα του j κρυμμένου νευρονίου θα είναι:

$$\delta_j(t) = f'_j(u_j(t)) \sum_m \delta_m(t) w_{mj}(t) \qquad (3.2.1.15)$$

όπου $\delta_m(t)$ m = 1,2,.. είναι τα δέλτα των νευρονίων εξόδου. Τελικά η διόρθωση προκύπτει ότι είναι:

$$\Delta w_{ji}(t) = \gamma\delta_j(t)y_i(t) = \gamma y_i(t) \sum_m \delta_m(t) w_{mj}(t) \qquad (3.2.1.16)$$

όπου ο δείκτης i αναφέρεται στα νευρόνια του στρώματος που προηγείται του υπό θεώρηση κρυμμένου στρώματος. Συνολικά τα βήματα του αλγορίθμου backpropagation (BP) είναι τα εξής:

1] Επιλογή των αρχικών βαρών και κατωφλίων χρησιμοποιώντας μικρές θετικές τυχαίες τιμές.
2] Παρουσιάζεται στο ΝΔ το πρότυπο διάνυσμα εκπαίδευσης $\mathbf{x}(t)=[x_0(t),x_1(t),\ldots,x_n(t)]^T$ και το επιθυμητό διάνυσμα εξόδου $\mathbf{d}(t)=[d_1(t),d_2(t),\ldots,d_M(t)]^T$.
3] Ευθύ πέρασμα, υπολογίζονται τα σήματα εξόδου όλων των νευρονίων του δικτύου προς τα εμπρός χρησιμοποιώντας τις τρέχουσες τιμές των συναπτικών βαρών δηλαδή

$$u_j = \sum_i w_{ji}(t) y_i(t) \qquad (3.2.1.17)$$

$$y_j(t) = f_j(u_j(t)) \qquad (3.2.1.18)$$

όπου $y_i(t)$ είναι η i είσοδος του j νευρονίου (δηλαδή η έξοδος του i νευρονίου) και w_{ji} είναι το συναπτικό βάρος που συνδέει το i νευρόνιο με το j νευρόνιο. Για τα νευρόνια j του πρώτου κρυμμένου στρώματος ισχύει $y_i(t) = x_i(t)$, i = 1,2,…,n όπου η i είσοδος είναι στοιχείο του διανύσματος διέγερσης. Για τους νευρώνες j του στρώματος εξόδου, το $y_j(t)$ είναι η j πραγματική έξοδος του νευρωνικού δικτύου.
4] Ανάστροφο πέρασμα, ανανεώνονται τα βάρη αρχίζοντας από τα νευρόνια εξόδου και προχωρώντας ανάποδα προς το στρώμα εξόδου, χρησιμοποιώντας τον κανόνα:

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) \qquad (3.2.1.19)$$

όπου το $\Delta w_{ji}(t)$ δίνεται από την (3.2.1.8) όταν το θεωρούμενο νευρόνιο ανήκει στο στρώμα εξόδου και από την (3.2.1.16) όταν ανήκει σε κρυμμένο στρώμα. Σημειώνεται ότι $w_{ji}(t)$ είναι το συναπτικό βάρος που συνδέει το νευρόνιο j ενός στρώματος με το νευρόνιο i του αμέσως προηγούμενου στρώματος.
5] Επαναλαμβάνεται η όλη διαδικασία από το βήμα 2.

3.2.2 Σημειώσεις στον Αλγόριθμο Ανάστροφης Διάδοσης

Αρχικά, αξίζει να σημειωθεί ότι η επιλογή των αρχικών τιμών για τα συναπτικά βάρη είναι πολύ σημαντική. Συνήθως σαν αρχικές τιμές των βαρών δίνονται μικρές θετικές τυχαίες τιμές ομοιόμορφα κατανεμημένες σε μια μικρή περιοχή. Σημειώνεται επίσης ότι δεν υπάρχει θεώρημα που να εξασφαλίζει ότι ο αλγόριθμος ανάστροφης διάδοσης ο οποίος βασίζεται στην μέθοδο gradient descent συγκλίνει σε ένα ολικό ελάχιστο. Έτσι ένα νευρωνικό δίκτυο που εκπαιδεύεται με τον αλγόριθμο BP παγιδεύεται συνήθως σε κάποιο τοπικό ελάχιστο.

Στην λειτουργία ομάδας προτύπων (batch mode) η εκκίνηση γίνεται με τυχαίες αρχικές τιμές για το διάνυσμα βαρών $\mathbf{w}^{(0)}$. Στην συνέχεια ενημερώνεται το διάνυσμα βαρών έτσι ώστε στο βήμα τ να γίνεται μετακίνηση προς την κατεύθυνση του μεγαλύτερου ρυθμού μείωσης της συνάρτησης σφάλματος:

$$\Delta \boldsymbol{w}^{(\tau)} = -\eta \nabla E|_{\boldsymbol{w}}^{(\tau)} \qquad (3.2.2.1)$$

Κατά την λειτουργία απλού προτύπου (pattern mode) η μερική παράγωγος της συνάρτησης σφάλματος αποτιμάται για κάθε ένα διάνυσμα εισόδου την φορά.

$$\Delta \boldsymbol{w}^{(\tau)} = -\eta \nabla E^n|_{\boldsymbol{w}}^{(\tau)} \qquad (3.2.2.2)$$

Τα διανύσματα πρότυπα υπάρχει η δυνατότητα να επιλέγονται με την σειρά ή τυχαία.

Η παράμετρος $\boldsymbol{\eta}$ (ή γ) καλείται ρυθμός μάθησης (learning rate), θεωρώντας ότι η τιμή της είναι αρκούντως μικρή αναμένεται ότι κατά την batch mode λειτουργία η τιμή του E θα μειώνεται σε κάθε διαδοχικό βήμα οδηγώντας σε ένα διάνυσμα βαρών το οποίο θα ικανοποιεί την συνθήκη $\nabla E = 0$. Από την άλλη πλευρά κατά την pattern mode λειτουργία για μικρές τιμές του η θα υπάρχει σταθερή μείωση του σφάλματος

αφού οι επιμέρους μεταβολές του διανύσματος των βαρών θα προσεγγίζουν ένα ολικό ελάχιστο.

Η εξασφάλιση σύγκλισης μπορεί να γίνει εάν ο ρυθμός μάθησης μειώνεται σε κάθε βήμα του αλγορίθμου σε αντιστοιχία με τις απαιτήσεις του θεωρήματος [1]. Αυτές ικανοποιούνται επιλέγοντας $\eta^{(\tau)} \propto 1/\tau$ παρότι μια τέτοια επιλογή καθυστερεί εμφανώς την σύγκλιση. Πρακτικά επιλέγεται μια σταθερή τιμή για το η μιας και αυτό παρέχει αρκετά καλά αποτελέσματα παρότι δεν υπάρχει εξασφάλιση για την σύγκλιση. Εάν το η είναι πολύ μεγάλο ο αλγόριθμος μπορεί να αποκλίνει οδηγώντας σε αύξηση του Ε, δημιουργώντας έτσι ταλαντώσεις οι οποίες κάνουν τον αλγόριθμο ασταθή. Από την άλλη πλευρά εάν το η επιλεγεί πολύ μικρό η εκπαίδευση προχωράει με μικρούς ρυθμούς οδηγώντας έτσι σε μεγάλους υπολογιστικούς χρόνους.

Ένα σημαντικό πλεονέκτημα της λειτουργία απλού προτύπου έναντι της λειτουργίας ομάδας προτύπων, εμφανίζεται όταν το σύνολο των διανυσμάτων εισόδων περιέχει πλεονάζουσα πληροφορία. Ως απλό παράδειγμα μπορεί να θεωρηθεί ένα σύνολο προτύπων εκπαίδευσης το οποίο προήλθε από ένα μικρότερο σύνολο προτύπων εκπαίδευσης επαναλαμβάνοντας 10 φορές τα πρότυπα του μικρότερου συνόλου. Σε αυτή την περίπτωση κάθε αποτίμηση του Ε χρειάζεται δεκαπλάσιο χρόνο και έτσι ο αλγόριθμος σε batch mode θα χρειαστεί ομοίως δεκαπλάσιο χρόνο για να οδηγηθεί σε μια δεδομένη λύση. Αντίθετα ο αλγόριθμος σε λειτουργία απλού προτύπου, ανανεώνοντας το διάνυσμα βαρών ανά πρότυπο, μένει ανεπηρέαστος από την επανάληψη των προτύπων. Ένα εξίσου σημαντικό πλεονέκτημα του pattern έναντι του batch mode είναι ότι από την στιγμή που ο αλγόριθμος BP (σε λειτουργία απλού προτύπου) είναι στοχαστικός έχει την δυνατότητα να ξεφύγει από τοπικά ελάχιστα.

Ο αλγόριθμος ανάστροφης διάδοσης δίνει μια προσέγγιση της τροχιάς των βαρών που υπολογίζονται με την μέθοδο της απότομης καθόδου. Όσο μικρότερος είναι ο ρυθμός μάθησης τόσο μικρότερη η μεταβολή του διανύσματος των βαρών σε κάθε επανάληψη και ομαλότερη η τροχιά της καμπύλης των βαρών. Το κόστος για βελτιωμένη μάθηση όμως είναι ο αργός ρυθμός εκπαίδευσης. Αντίθετα, αν χρησιμοποιηθεί υψηλός ρυθμός μάθησης για να επιτευχθεί επιτάχυνση της διαδικασίας τότε οι μεγάλες μεταβολές σε κάθε επανάληψη προκαλούν τον κίνδυνο της αστάθειας του αλγορίθμου εκπαίδευσης. Μια απλή μέθοδος της αύξησης του ρυθμού, με αποφυγή των παραπάνω αρνητικών φαινομένων, είναι η τροποποίηση του κανόνα ενημέρωσης των βαρών με την εισαγωγή ενός όρου ορμής (momentum term).

$$\Delta w_{ji}(t) = \alpha \Delta w_{ji}(t-1) + \eta \delta_j(t) y_i(t) \qquad (3.2.2.3)$$

όπου $\boldsymbol{\alpha}$ ένας (συνήθως θετικός) αριθμός που ονομάζεται σταθερά ορμής (momentum). Για να γίνει εμφανής η επίπτωση του α στον αλγόριθμο της εκπαίδευσης [2] αναδιατυπώνεται η προηγούμενη εξίσωση ως μια χρονική ακολουθία με δείκτη t:

$$\Delta w_{ji}(t) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t) \qquad (3.2.2.4)$$

ισοδύναμα από την σχέση (3.2.1.5) προκύπτει ότι:

$$\Delta w_{ji}(t) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E_p(t)}{\partial w_{ji}(t)} \qquad (3.2.2.5)$$

Βάσει αυτής της σχέσης μπορούν να γίνουν ορισμένες παρατηρήσεις.

- Η τρέχουσα μεταβολή $\Delta w_{ji}(t)$ αντιπροσωπεύει το άθροισμα μιας εκθετικής χρονικής ακολουθίας. Για να συγκλίνει αυτή η ακολουθία πρέπει η σταθερά α να ανήκει στο διάστημα $0 \le |\alpha| < 1$. Όταν $\alpha = 0$ ο αλγόριθμος λειτουργεί χωρίς momentum. Η σταθερά α μπορεί να παίρνει και αρνητικές τιμές αν και αυτό δεν συνηθίζεται.

- Όταν η μερική παράγωγος $\frac{\partial E_p(t)}{\partial w_{ki}(t)}$ έχει το ίδιο πρόσημο σε συνεχόμενες επαναλήψεις τότε το $\Delta w_{ji}(t)$ μεγαλώνει σημαντικά και το βάρος $w_{ji}(n)$ μεταβάλλεται σε μεγάλο βαθμό. Επομένως η εισαγωγή του παράγοντα ορμής τείνει να επιταχύνει σε σταθερή κατεύθυνση την κάθοδο.
- Όταν η μερική παράγωγος $\frac{\partial E_p(t)}{\partial w_{ki}(t)}$ έχει αντίθετο πρόσημο σε συνεχόμενες επαναλήψεις, τότε το $\Delta w_{ji}(t)$ μειώνεται σημαντικά και το βάρος $w_{ji}(n)$ μεταβάλλεται σε μικρό βαθμό. Επομένως η εισαγωγή του παράγοντα ορμής έχει σταθεροποιητικό αποτέλεσμα

Συνεπώς η ενσωμάτωση του όρου ορμής στον κανόνα ενημέρωσης των βαρών μπορεί να προσδώσει σημαντικά οφέλη αναφορικά με την διαδικασία μάθησης. Παράλληλα αποτρέπει τον τερματισμό της διαδικασίας σε ένα τοπικό ελάχιστο της επιφάνειας σφάλματος.

Τελευταία έχουν υπάρξει προσπάθειες για την βελτίωση της επίδοσης του αλγορίθμου ανάστροφης διάδοσης κάνοντας διάφορες κατά περίπτωση τροποποιήσεις του αλγορίθμου. Ένα εμφανές μειονέκτημα του αλγορίθμου είναι ότι περιέχει δυο παραμέτρους η και α, των οποίων οι τιμές επιλέγονται κάνοντας ορισμένες σειρές δοκιμών. Οι βέλτιστες τιμές αυτών των παραμέτρων εξαρτώνται από το εκάστοτε πρόβλημα και επιπλέον μεταβάλλονται κατά την διάρκεια της εκπαίδευσης. Επομένως είναι λογικό να αναζητάται μια διαδικασία η οποία θα επιλέγει τις τιμές των η και α αυτόματα κατά την εξέλιξη του αλγορίθμου BP. Μια τέτοια προσέγγιση αποτελεί η τεχνική bold driver [3].

Αντιμετωπίζοντας το πρόβλημα χωρίς την ύπαρξη του όρου ορμής η ιδέα που κρύβεται πίσω από την τεχνική bold driver είναι να ελέγχεται το κατά πόσο μειώθηκε η τιμή της συνάρτησης λάθους μετά από κάθε πέρασμα του αλγορίθμου ανάστροφης διάδοσης. Εάν αυτή η τιμή έχει αυξηθεί τότε ο BP έχει ξεπεράσει το προηγούμενο ελάχιστο, πράγμα που σημαίνει ότι η τιμή του ρυθμού μάθησης ήταν πολύ μεγάλη. Σε αυτή την περίπτωση η μεταβολή του διανύσματος των βαρών αναιρείται και το η μειώνεται. Αυτή η διαδικασία επαναλαμβάνεται μέχρι να ανιχνευθεί μια μείωση στην συνάρτηση σφάλματος. Εάν από την άλλη πλευρά το σφάλμα μειώνεται σε κάποιο βήμα του αλγορίθμου τότε η μεταβολή του διανύσματος βαρών διατηρείται. Εντούτοις η τιμή του ρυθμού μάθησης μπορεί να ήταν πολύ μικρή οπότε αυξάνεται.το η. Συνολικά η τιμή του η μεταβάλλεται όπως φαίνεται στην συνέχεια:

$$\eta_{new} = \begin{cases} \rho\eta_{old}\ ,\ \Delta E<0 \\ \\ \sigma\eta_{old}\ ,\ \Delta E>0 \end{cases} \qquad (3.2.2.6)$$

Η παράμετρος ρ επιλέγεται να είναι λίγο μεγαλύτερη της μονάδας (μια τυπική τιμή μπορεί να είναι ρ = 1.1) ώστε να αποφεύγονται συχνές εμφανίσεις αύξησης του σφάλματος, αφού στις εν λόγω περιπτώσεις σπαταλάται υπολογιστική ισχύς. Η παράμετρος σ επιλέγεται σαφώς μικρότερη της μονάδας (μια τυπική τιμή μπορεί να είναι σ = 0.5), έτσι ώστε ο αλγόριθμος να ανακάμπτει γρήγορα βρίσκοντας ένα διάνυσμα βαρών που μειώνει το σφάλμα (γεγονός που και πάλι εξοικονομεί υπολογιστική ισχύ). Διάφορες τροποποιήσεις μπορούν να γίνουν όπως η αύξηση του η γραμμικά (κατά μια σταθερή τιμή) αντί εκθετικά (κατά έναν σταθερό παράγοντα). Εάν συμπεριληφθεί ο όρος ορμής, αυτός μπορεί να τεθεί σε μια σταθερή τιμή. Για περισσότερες πληροφορίες πάνω στον αλγόριθμο ανάστροφης διάδοσης [4].

3.3 Αξιοπιστία στην Ταξινόμηση

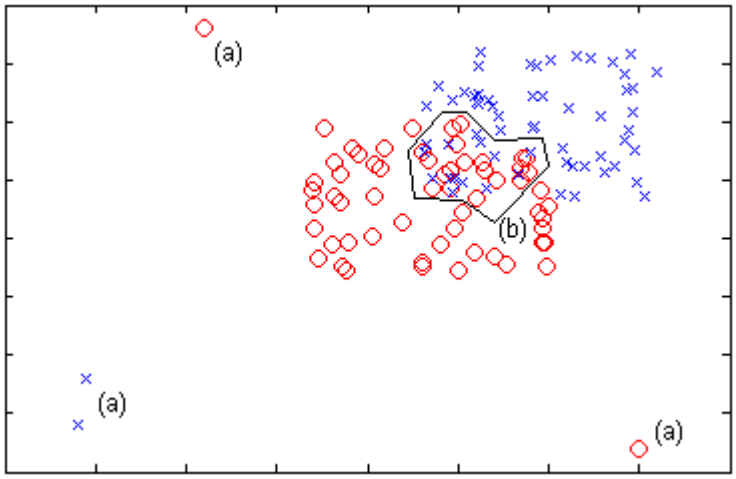
3.3.1 Εισαγωγικό Σημείωμα

Μια προσέγγιση για τον έλεγχο του κατά πόσο ένας ταξινομητής έχει παράγει μια ικανοποιητική λύση είναι να εξεταστούν αυτές καθ΄ αυτές οι τιμές που έχουν παραχθεί από τον ταξινομητή. Σε πρακτικά προβλήματα ένας μπορεί να διαθέτει έναν ταξινομητή, χωρίς να διαθέτει όμως καμία επιπλέον πληροφόρηση για την δομή, την αρχιτεκτονική ή τον τρόπο που έχει εκπαιδευτεί ο εν λόγω ταξινομητής. Συνεπώς για τα πρότυπα που ταξινομούνται από το νευρωνικό δίκτυο ταξινόμησης χρειάζεται ένα μέτρο αξιοπιστίας-εμπιστοσύνης που να παρέχει την πληροφόρηση του κατά πόσο ένας μπορεί να είναι βέβαιος για το αποτέλεσμα που έχει παράγει ο ταξινομητής. Εάν η απόδοση του ταξινομητή είναι περιορισμένη η αξιοπιστία για την ταξινόμηση ενός προτύπου αναμένεται να έχει μικρή τιμή. Αντίθετα για έναν ταξινομητή με βέλτιστη απόδοση, η τιμή του μέτρου εμπιστοσύνης (για το ίδιο πρότυπο εισόδου) θα εμφανίζεται αυξημένη.

Η τιμή του μέτρου αξιοπιστίας μπορεί να έχει ποικίλες χρήσεις και εφαρμογές. Μια πρώτη εφαρμογή είναι η βελτίωση της απόδοσης ενός ταξινομητή χρησιμοποιώντας απόρριψη. Με την χρήση κανόνων απόρριψης ένα πρότυπο με χαμηλή τιμή αξιοπιστίας μπορεί να αποκλειστεί από την φάση της ταξινόμησης, βελτιώνοντας έτσι την συνολική ικανότητα ορθής ταξινόμησης ενός ταξινομητή. Μια άλλη εφαρμογή είναι η κατασκευή ταξινομητών πολλών επιπέδων. Αρχικά υποθέτουμε ότι υπάρχει ένας αριθμός διαθέσιμων ταξινομητών για την ταξινόμηση ενός προτύπου. Κατά την φάση της ταξινόμησης εκτός της κατηγορίας υπολογίζεται η τιμή του μέτρου εμπιστοσύνης για κάθε έναν ταξινομητή. Οι τιμές της αξιοπιστίας στην συνέχεια χρησιμοποιούνται για την κατασκευή ενός σύνθετου ταξινομητή ο οποίος εμφανίζει υψηλότερη απόδοση από ότι οι επιμέρους ταξινομητές.

Σύμφωνα με τα όσα αναφέρθηκαν προηγούμενα πιο αποδοτικές λύσεις για το πρόβλημα της ταξινόμησης μπορούν να επιτευχθούν εισάγοντας έναν όρο αξιοπιστίας ο οποίος θα παρουσιάζει την ακρίβεια της ταξινόμησης κάθε ενός προτύπου. Ο υπολογισμός του μέτρου της αξιοπιστίας προϋποθέτει ότι καταστάσεις που εμφανίζονται στον χώρο των εισόδων και που είναι επιρρεπείς σε μη αξιόπιστες ταξινομήσεις πρέπει να απεικονίζονται με κάποιο τρόπο στον χώρο των εξόδων του ταξινομητή. Η χαμηλή αξιοπιστία μιας ταξινόμησης δύναται να εντοπιστεί σε μια από τις δυο ακόλουθες καταστάσεις στον χώρο των εισόδων.

(a) Το πρότυπο προς ταξινόμηση είναι χαρακτηριστικά διαφορετικό από αυτά που εμφανίζονταν κατά την φάση της εκπαίδευσης του ταξινομητή. Για παράδειγμα οι τιμές του διανύσματος του είναι απομακρυσμένες από τις τιμές των διανυσμάτων του συνόλου εκπαίδευσης.
(b) Το πρότυπο προς ταξινόμηση εμφανίζει ομοιότητες με πρότυπα που εμφανίζονταν κατά την φάση της εκπαίδευσης και που ανήκουν σε διαφορετικές κατηγορίες. Για παράδειγμα οι τιμές του διανύσματος του προτύπου βρίσκονται σε περιοχή που συνυπάρχουν πρότυπα από δυο ή περισσότερες διαφορετικές κλάσεις.



Σχήμα 3.3.1.1: Δυο περιπτώσεις μη αξιόπιστης ταξινόμησης. (a) Τα δείγματα είναι χαρακτηριστικά διαφορετικά από τα πρότυπα του συνόλου εκπαίδευσης, (b) Τα δείγματα εντοπίζονται σε μια περιοχή που συνυπάρχουν πρότυπα δυο κατηγοριών.

Για να γίνεται εύκολη διάκριση μεταξύ των μη αξιόπιστων ταξινομήσεων τύπου (a) και τύπου (b) εισάγονται δυο παράμετροι αξιοπιστίας ψ_a και ψ_b οι τιμές των οποίων γενικά βρίσκονται στην περιοχή τιμών [0,1]. Επιπλέον θεωρείται ότι παράμετροι με τιμές κοντά στο 1 αντιστοιχούν σε αξιόπιστες ταξινομήσεις ενώ χαμηλές τιμές αντιστοιχούν σε λιγότερο αξιόπιστες ταξινομήσεις. Οι δυο αυτές παράμετροι σχετίζονται με το διάνυσμα των εξόδων του ταξινομητή. Μια συνολική παράμετρος αξιοπιστίας είναι δυνατόν να υπολογιστεί με χρήση των ψ_a και ψ_b:

$$\psi = min(\psi_a, \psi_b) \qquad (3.3.1.1)$$

Μια τέτοια επιλογή είναι εμφανώς συντηρητική αφού υπονοεί ότι μια ταξινόμηση είναι μη αξιόπιστη από την στιγμή που έστω και μια παράμετρος από τις ψ_a και ψ_b λαμβάνει χαμηλή τιμή ανεξαρτήτως της τιμής που λαμβάνει η άλλη παράμετρος. Προφανώς ένας διαφορετικός συνδυαστικός τελεστής μπορεί να χρησιμοποιηθεί αναλόγως των απαιτήσεων που υπάρχουν σε κάθε επιμέρους εφαρμογή ταξινόμησης

3.3.2 Εκτιμητές Αξιοπιστίας

Όταν ένα νευρωνικό δίκτυο χρησιμοποιείται για την ταξινόμηση ενός αγνώστου προτύπου ένα από τα ζητούμενα είναι κάποιο μέγεθος το οποίο να εμπεριέχει την πληροφορία την ποιότητας αυτής της ταξινόμησης, δηλαδή την πιθανότητα του να είναι η ταξινόμηση επιτυχημένη [1].

Έστω ένα ΝΔ ταξινόμησης το οποίο εκπαιδευμένο παρουσιάζει μια συνάρτηση απεικόνισης S των εισόδων στις εξόδους, της μορφής:

$$S(\vec{x}) = \hat{w} \qquad (3.3.2.1)$$

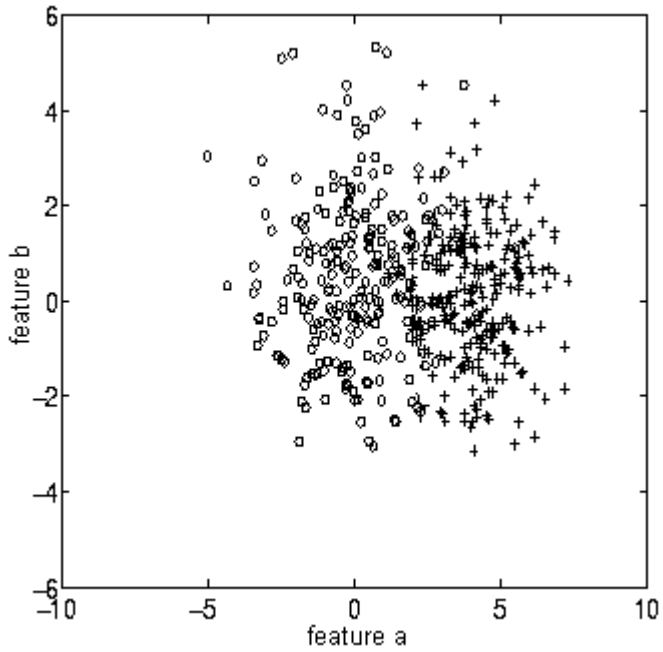
Για κάθε πρότυπο εισόδου $\vec{x}$ η ταξινόμηση που βασίζεται στην S είναι είτε ορθή είτε λανθασμένη, συνεπώς μια συνάρτηση αληθείας μπορεί να οριστεί:

$$C(\vec{x}, w) = \begin{cases} 1\ ,\ S(\vec{x}) = w \\ 0,\ S(\vec{x}) \neq w \end{cases} \qquad (3.3.2.2)$$

όπου w είναι η πραγματική κατηγορία του προτύπου $\vec{x}$. Όταν ταξινομείται ένα άγνωστο πρότυπο η αξιοπιστία για την εκτίμηση $\hat{w}$ της κατηγορίας του προτύπου αντιπροσωπεύει την πιθανότητα του να είναι ορθή η ταξινόμηση. Δοθέντος ενός ταξινομητή το ζητούμενο είναι ο υπολογισμός αυτής της πιθανότητας.

$$q(\vec{x}) = P(C(\vec{x}, w) = 1 \mid \vec{x})$$
$$= P(S(\vec{x}) = w \mid \vec{x})$$
$$= P(\hat{w} = w \mid \vec{x}, S) \qquad (3.3.2.3)$$

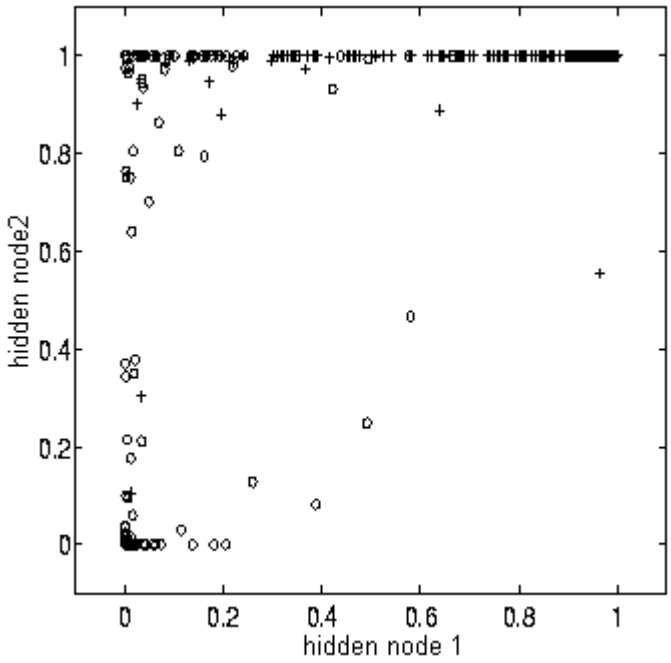
Ένα νευρωνικό δίκτυο απεικονίζει το υπερεπίπεδο των εισόδων στο υπερεπίπεδο των κρυμμένων στρωμάτων και στην συνέχεια στο υπερεπίπεδο των εξόδων. Κατά συνέπεια η πιθανότητα $q(\vec{x})$ μπορεί να εκτιμηθεί στο χώρο των εισόδων, στον χώρο των κρυμμένων νευρώνων όπως και στον χώρο των εξόδων.



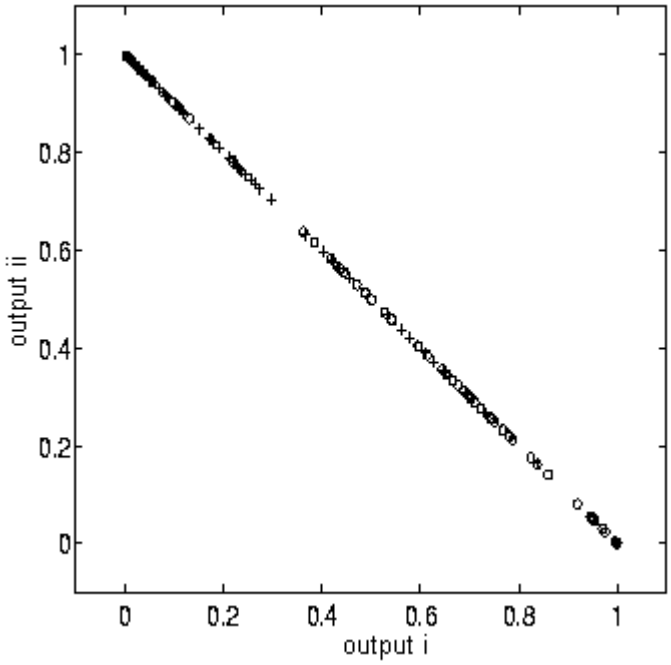
Σχήμα 3.3.2.1: Χώρος των εισόδων για ένα πρόβλημα ταξινόμησης 2 κατηγοριών.

Έστω ένα νευρωνικό δίκτυο 2 εισόδων, με 2 νευρόνια στο κρυμμένο στρώμα και 2 εξόδους κάθε μια εκ των οποίων σχετίζεται με μια κατηγορία. Το ΝΔ εκπαιδεύεται βάσει των προτύπων που εμφανίζονται στο σχήμα (3.3.2.1). Εάν κάποιος παρατηρήσει την τοπολογική διάταξη των 2 κατηγοριών στον χώρο των κρυμμένων στρωμάτων, μπορεί να παρατηρήσει ότι αυτές οι 2 κατηγορίες είναι πιο εύκολα διαχωρίσιμες. Συνεπώς ο υπολογισμός της $q(\vec{x})$ σε αυτό το στρώμα του νευρωνικού δικτύου συγκεντρώνει αρκετά πλεονεκτήματα.

Αξίζει επίσης να σημειωθεί το γεγονός ότι τα ΝΔ κάνοντας μια συνεχή απεικόνιση μεταξύ των χώρων των εισόδων, των κρυμμένων νευρώνων και των εξόδων παρέχουν την δυνατότητα υπολογισμού της πιθανότητας $q(\vec{x})$ σε οποιονδήποτε από αυτούς τους χώρους.



Σχήμα 3.3.2.2: Χώρος του κρυμμένου στρώματος.



Σχήμα 3.3.2.3: Χώρος των εξόδων για το πρόβλημα ταξινόμησης 2 κατηγοριών.

Γενικά υπάρχουν διάφορες μέθοδοι για την μέτρηση της τιμής της αξιοπιστίας και κατά συνέπεια για τον υπολογισμό της πιθανότητας $q(\vec{x})$.

1] Έξοδοι του Δικτύου (Network Outputs)
2] k-οστός Πλησιέστερος Γείτονας (k Nearest Neighbor)
3] Λογιστική Μέθοδος (Logistic Method)
4] Μέθοδος του Νικητή (Winner Method)

Έξοδοι του Δικτύου: Η μέθοδος αυτή αποτελεί τον κλασσικό τρόπο προσδιορισμού της αξιοπιστίας ενός δικτύου ταξινόμησης. Κάθε έξοδος του ΝΔ μπορεί να

μεταφραστεί ως η posteriori πιθανότητα εκτίμησης της κατηγορίας ενός προτύπου. Αυτό οδηγεί στον ακόλουθο κανόνα υπολογισμού της αξιοπιστίας:

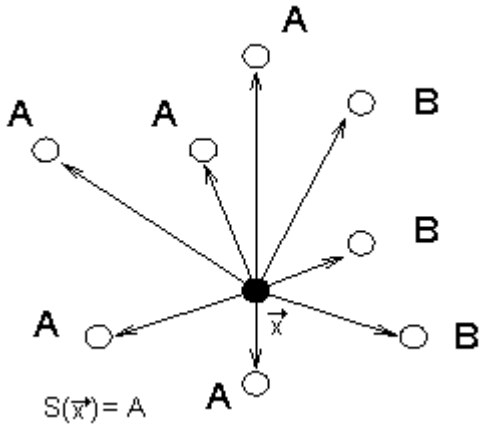
$$\hat{q}_{netout}(\vec{x}) = \frac{o_i(\vec{x})}{\sum_{i=1}^{N} o_i(\vec{x})} \qquad (3.3.2.4)$$

όπου $o_i(\vec{x})$ είναι η τιμή του κόμβου εξόδου i που αντιπροσωπεύει την κατηγορία w_i θεωρώντας βέβαια μια έξοδο ανά κατηγορία. Το άθροισμα χρησιμοποιείται για να κανονικοποιηθεί το μέτρο αξιοπιστίας, ενώ N είναι ο αριθμός των κατηγοριών. Σε γενικές γραμμές δεν είναι γνωστό πότε ένα ΝΔ είναι επαρκώς εκπαιδευμένο, αυτό σημαίνει ότι οι posteriori πιθανότητες των προτύπων ενδεχόμενα να είναι ανεπαρκώς εκτιμημένες. Επιπρόσθετα οι έξοδοι των ΝΔ δεν μεταφράζονται πάντα σε πιθανότητες. Το γεγονός αυτό σημαίνει ότι χρειάζεται μια επεξεργασία των τιμών των εξόδων ώστε να αντιπροσωπεύουν πιθανότητες ή μπορεί να τροποποιηθεί ο αλγόριθμος εκπαίδευσης ή μπορεί να προσαρμοστεί κατάλληλα το κριτήριο σφάλματος. Η τελευταία επιλογή που ενδείκνυται να γίνει είναι η τροποποίηση του αλγορίθμου εκπαίδευσης αφού ο ταξινομητής θεωρείται δεδομένος και ο ακριβής τρόπος εκπαίδευσης του εν γένει άγνωστος. Η καλύτερη επιλογή είναι να λαμβάνονται υπόψη οι τιμές των εξόδων αφού είναι αδύνατο να ανατεθεί μια κλάση στο κόμβους του κρυμμένου στρώματος ή του στρώματος των εισόδων.

<u>k-οστός Πλησιέστερος Γείτονας:</u> Θεωρώντας έναν δεδομένο χώρο με ένα καθορισμένο τρόπο εκτίμησης των αποστάσεων (για παράδειγμα την Ευκλείδεια απόσταση) η εκτιμήτρια πυκνότητας της μεθόδου k Nearest Neighbor (k-NN) μπορεί να χρησιμοποιηθεί ώστε να βρεθεί η αξιοπιστία της ταξινόμησης του υπό εξέταση προτύπου.

$$\hat{q}_{knn}(\vec{x}) = \frac{N_{S(\vec{x})=\vec{w}}}{k} \qquad (3.3.2.5)$$

όπου $N_{S(\vec{x})=\vec{w}}$ είναι ο αριθμός των δειγμάτων μεταξύ των k πλησιέστερων γειτόνων τα οποία ανήκουν στην υπό εξέταση κατηγορία. Οι τιμές των γειτονικών προτύπων είναι γνωστές (από την φάση της εκπαίδευσης) και το πρότυπο $\vec{x}$ κατηγοριοποιείται από τον ταξινομητή S. Όσο περισσότεροι γείτονες συμφωνούν περί της κατηγορίας του $\vec{x}$ τόσο μεγαλύτερη εμπιστοσύνη υπάρχει για την ορθή ταξινόμηση του προτύπου. Παραστατικά η λειτουργία της μεθόδου φαίνεται στο ακόλουθο σχήμα.



Σχήμα 3.3.2.4: Η μέθοδος k-NN για τον υπολογισμό της αξιοπιστίας.

Έστω ότι επιθυμείται να βρεθεί η αξιοπιστία της ταξινόμησης του μαύρου προτύπου, επιπλέον έχει γίνεται η θεώρηση ότι τα γειτονικά πρότυπα είναι γνωστό ότι ανήκουν στις κατηγορίες Α ή Β. Χρησιμοποιώντας την 8-NN προσέγγιση η αξιοπιστία που υπολογίζεται είναι 5/8 αφού 5 από τους 8 γείτονες ανήκουν στην κατηγορία Α. Το πρόβλημα επιλογής του k ωστόσο παραμένει, μια ευριστική τιμή είναι $k = \sqrt{N_L}$ όπου N_L είναι ο αριθμός των προτύπων στο σύνολο εκπαίδευσης. Ένα σημαντικό μειονέκτημα της μεθόδου είναι η υπολογιστική ισχύς που απαιτεί στην περίπτωση που ο αριθμός των προτύπων είναι μεγάλος.

Λογιστική Μέθοδος: Η λογιστική μέθοδος [2] είναι σχετικά απλή ενώ οι παράμετροι που χρησιμοποιεί αρκεί να υπολογιστούν μόνο μια φορά. Κατά την λογιστική μέθοδο οι υπό συνθήκη πιθανότητες των κατηγοριών μοντελοποιούνται χρησιμοποιώντας την σιγμοειδή συνάρτηση. Για ένα πρόβλημα δυο κατηγοριών η πιθανότητα το πρότυπο $\vec{x}$ να ταξινομείται στην σωστή κατηγορία δίδεται από τον τύπο:

$$\hat{q}_{\log istic}(\vec{x}) = \frac{1}{1+\exp(\vec{\beta}\vec{x} + \beta_0)} \qquad (3.3.2.6)$$

Οι επιμέρους παράμετροι που πρέπει να υπολογιστούν είναι οι $\vec{\beta}$ και β_0. Ο συνολικός αριθμός των παραμέτρων που πρέπει να υπολογιστούν ισούται με την διάσταση του $\vec{\beta}$ συν 1. Αυτές μπορούν να υπολογιστούν χρησιμοποιώντας ένα σύνολο εκπαίδευσης και τεχνική βελτιστοποίησης (για παράδειγμα τον αλγόριθμο του Newton). Όπως αναφέρθηκε προηγούμενα ο τύπος (3.3.2.6) ισχύει για την περίπτωση των δυο κατηγοριών αν και επεκτείνεται σε προβλήματα με περισσότερες κατηγορίες. Αυτό γίνεται θεωρώντας μια κλάση ως την κατηγορία βάση g και στην συνέχεια υπολογίζοντας όλες τις παραμέτρους βάσει αυτή της κατηγορίας.

$$\hat{q}_{\log istic}(\vec{x}) = (1 + \sum_{i=1}^{Ng-1} \exp(-\vec{a}_{gi}\vec{x}^{\,i}))^{-1} \qquad (3.3.2.7)$$

όπου τα $\vec{x}^{\,i} = (\vec{x},1)$ και α_{gi} αντιπροσωπεύουν τα $(\vec{\beta},\beta_0)$ για την κλάση i ως προς την κλάση βάση g, ενώ το N_g αντιστοιχεί στον αριθμό των κατηγοριών.

Μέθοδος του Νικητή: Στις συνήθεις περιπτώσεις ταξινομητών οι κόμβοι εξόδου είναι ίσοι με τις κλάσεις ταξινόμησης. Από θεωρητικής απόψεως εάν ένα πρότυπο ανήκει στην k-οστή κατηγορία τότε ο k-οστός νευρώνας του στρώματος εξόδου θα έχει τιμή ίση με 1 ενώ όλες οι άλλες έξοδοι θα λαμβάνουν τιμή 0 (ιδανικό διάνυσμα εξόδου). Στην πράξη οι τιμές που περιέχει το διάνυσμα εξόδου διαφέρουν από αυτές του ιδανικού διανύσματος εξόδου. Εξαιτίας αυτού η κατηγορία κάθε προτύπου υπολογίζεται χρησιμοποιώντας κάποιο κανόνα. Ο απλούστερος κανόνας είναι ο Winner Takes All σύμφωνα με τον οποίο κάθε πρότυπο εισόδου αντιστοιχίζεται στην κλάση της οποίας ο νευρώνας εξόδου έχει την υψηλότερη τιμή.

Το ΝΔ ταξινομητής χρησιμοποιεί τις τιμές των συναπτικών βαρών για να καθορίσει τα υπερεπίπεδα που ορίζουν τις περιοχές απόφασης των κατηγοριών. Στο στάδιο της εκπαίδευσης οι τιμές των βαρών μεταβάλλονται, όταν εμφανίζεται κάθε νέο πρότυπο στις εισόδους του ταξινομητή, έτσι ώστε το διάνυσμα εξόδου να προσεγγίζει το ιδανικό. Συνακόλουθα τα πρότυπα του συνόλου ελέγχου διαφέρουν αρκετά από τα πρότυπα του συνόλου εκπαίδευσης (τα οποία όρισαν την μορφή των υπερεπιπέδων απόφασης) και έτσι είναι πιθανό να πέσουν έξω από κάθε υπερεπίπεδο [3]. Σε αυτή την περίπτωση τα νευρόνια εξόδου λαμβάνουν εξαιρετικά μικρές τιμές. Ένας αποδοτικός ορισμός της παραμέτρου ψ_a (σχέση (3.3.1.1)) είναι:

$\psi_a = O_{win}$ (3.3.2.8)

όπου O_{win} είναι η τιμή της εξόδου του νευρώνα νικητή. Κατά αυτό τον τρόπο όσο πλησιέστερα στο 0 είναι η τιμή O_{win} τόσο μικρότερη είναι η αξιοπιστία της εν λόγω ταξινόμησης.

Δείγματα του τύπου (b) όπου περισσότερες από μια περιοχές απόφασης επικαλύπτονται δημιουργούν διανύσματα εξόδου με δυο ή περισσότερες τιμές στοιχείων να είναι παρεμφερείς. Έτσι για ένα δεδομένο O_{win} όσο μεγαλύτερη είναι η διαφορά του από το O_{win2} (δηλαδή την έξοδο του νευρώνα με την μεγαλύτερη τιμή μετά τον νικητή νευρώνα) τόσο μεγαλύτερη εμπιστοσύνη μπορεί να έχει ένας στο αποτέλεσμα της ταξινόμησης του συγκεκριμένου προτύπου. Συνεπώς ένας αποδοτικός ορισμός της παραμέτρου ψ_b (σχέση (3.3.1.1)) είναι:

$$\psi_b = O_{win} - O_{win2} \qquad (3.3.2.9)$$

Συγκεφαλαιώνοντας μια συνολική παράμετρος αξιοπιστίας είναι δυνατόν να υπολογιστεί βάσει της σχέσης (3.3.1.1):

$$\begin{aligned} \psi &= min(\psi_a, \psi_b) = min(O_{win}, O_{win} - O_{win2}) \\ &= O_{win} - O_{win2} \\ &= \psi_b \end{aligned} \qquad (3.3.2.10)$$

3.3.3 Τροποποιημένος Εκτιμητής Αξιοπιστίας

Λαμβάνοντας υπόψη το νευρο-ασαφές σύστημα ταξινόμησης [§ 4] αλλά και την αρχιτεκτονική του συστήματος [§ 7] που χρησιμοποιήθηκαν στην παρούσα διπλωματική εργασία είναι απαραίτητη η χρήση ενός εκτιμητή αξιοπιστίας ο οποίος πρέπει να συμβαδίζει με ορισμένες προδιαγραφές.

I. Οι έξοδοι του ταξινομητή να μην χρειάζεται να είναι posteriori πιθανότητες εκτίμησης της κατηγορίας.
II. Το μέτρο αξιοπιστίας να είναι ανεξάρτητο του αριθμού των εξόδων του νευρο-ασαφούς συστήματος.
III. Να μην χρειάζεται καμία τροποποίηση του αλγορίθμου εκπαίδευσης.
IV. Ο εκτιμητής αξιοπιστίας ενδείκνυται να εφαρμόζεται στο στρώμα εξόδων αφού είναι αδύνατο να ανατεθεί μια κλάση στο κόμβους του κρυμμένου στρώματος ή του στρώματος των εισόδων.
V. Η μόνη χρησιμοποιούμενη πληροφορία πρέπει να είναι η παρούσα κατάσταση του νευρο-ασαφούς συστήματος και δεν πρέπει να υπάρχει καμία απαίτηση για περισσότερες πληροφορίες (για παράδειγμα γνώση των προτύπων του συνόλου εκπαίδευσης).
VI. Ο υπολογισμός τυχόν άλλων παραμέτρων πρέπει να είναι μικρής έκτασης και να μην χρειάζεται εκ νέου περιόδους εκπαίδευσης ούτε βέβαια την αποθήκευση νέων παραμέτρων.
VII. Η υπολογιστική ισχύς που απαιτείται για τον αναλυτικό υπολογισμό του μέτρου αξιοπιστίας οφείλει να είναι αμελητέα.
VIII. Η τιμή του μέτρου εμπιστοσύνης πρέπει να είναι κατά το δυνατόν αντιπροσωπευτική της επίδοσης της υπό εξέταση ταξινόμησης.

Η πρώτη προδιαγραφή δεν είναι δεσμευτική αφού υπάρχει δυνατότητα να προσαρτηθεί μια εκπαιδευμένη σιγμοειδής συνάρτηση στις εξόδους του SuPFuNIS [1]. Εντούτοις μια τέτοια λύση παρεμβαίνει τις προδιαγραφές V, VI, VII γεγονός που την καθιστά ανεφάρμοστη στην πράξη.

Οι περισσότερες μέθοδοι που αναφέρονται προηγούμενα [§ 3.3.2] αποτυγχάνουν να ικανοποιήσουν αρκετές από αυτές τις προδιαγραφές. Η μέθοδος network outputs προφανώς δεν ικανοποιεί τις προδιαγραφές I και III ενώ δεν

ικανοποιεί και την ΙΙ. Αυτό συμβαίνει στην περίπτωση που ο αριθμός των εξόδων είναι μεγάλος, τότε η κανονικοποίηση που γίνεται θα δώσει πολύ μικρή τιμή παρότι η τιμή του $o_i(\vec{x})$ μπορεί να είναι υψηλή.

Η μέθοδος k Nearest Neighbor δεν ικανοποιεί ην προδιαγραφή V αφού χρειάζεται να είναι γνωστό το σύνολο των προτύπων εκπαίδευσης για να εφαρμοστεί. Το γεγονός όμως που την κάνει απαγορευτική είναι ότι δεν ανταποκρίνεται στην προδιαγραφή VII.

Η logistic μέθοδος βάσει του τρόπου που έχει οριστεί αλλά και σύμφωνα με τους υπολογισμούς που απαιτεί για να υλοποιηθεί έρχεται σε σύγκρουση με τις προδιαγραφές VI, VII. Αυτό είναι έκδηλο από την στιγμή που για να λειτουργήσει χρειάζεται τον υπολογισμό των παραμέτρων $\vec{x}^i = (\vec{x},1)$ και α_{gi} με χρήση κάποιου αλγορίθμου βελτιστοποίησης.

Τέλος εξετάζοντας συστηματικά την winner μέθοδο κανείς μπορεί να παρατηρήσει ότι ικανοποιεί σε απόλυτο σχεδόν βαθμό το σύνολο των προδιαγραφών που σκιαγραφήθηκαν, εκτός ίσως της τελευταίας προδιαγραφής. Η μέθοδος του νικητή υστερεί σε ένα συγκεκριμένο σημείο. Υπολογίζοντας το μέτρο αξιοπιστίας από την σχέση (3.3.2.10) δεν λαμβάνει υπόψη την τιμή εξόδου O_{win} του νευρώνα νικητή. Προφανώς όσο μεγαλύτερη τιμή λαμβάνει το O_{win} τόσο μεγαλύτερη εμπιστοσύνη υπάρχει για την αντίστοιχη ταξινόμηση. Από την άλλη πλευρά υπάρχει το ενδεχόμενο η τιμή ψ για δυο διαφορετικά ζεύγη εξόδων να είναι η ίδια ($[O'_{win},O'_{win2}]$ και $[O_{win},O_{win,}]$) παρότι ισχύει $O_{win} > O'_{win}$. Αυτή η παρατήρηση οδηγεί στο συμπέρασμα ότι η ταξινόμηση με εξόδους τις O_{win} και O_{win2} θα έπρεπε να εμφανίζει υψηλότερη τιμή αξιοπιστίας κάτι που δεν το επιτυγχάνει η μέθοδος winner. Μια τροποποίηση που μπορεί να γίνει και η οποία επιλύει τα προηγούμενα προβλήματα παρουσιάζεται στην συνέχεια:

$$\psi = (O_{win} - O_{win2})O_{win} \qquad (3.3.3.1)$$

Χρησιμοποιώντας αυτόν τον τύπο ως εκτιμητή αξιοπιστίας για τις κατηγοριοποιήσεις που επιτελεί ο ταξινομητής ικανοποιούνται πλήρως οι οχτώ προδιαγραφές ενώ παράλληλα αντιμετωπίζονται οι επιμέρους αστοχίες της μεθόδου του νικητή.

Κεφάλαιο 4

Νευρο-Ασαφές Σύστημα Εξαγωγής Συμπερασμάτων στηριζόμενο στο Γινόμενο Βαθμών Ομοιότητας

4 Νευρο-Ασαφές Σύστημα Εξαγωγής Συμπερασμάτων στηριζόμενο στο Γινόμενο Βαθμών Ομοιότητας

4.1 Εισαγωγή

Το μοντέλο νευρο-ασαφούς συστήματος στο οποίο βασίστηκε η παρούσα υλοποίηση, και παρουσιάστηκε στο [1], είναι γνωστό ως Subsethood-Product Fuzzy Neural Inference System (SuPFuNIS). Έχει την δυνατότητα να χειρίζεται τόσο αριθμητικές όσο και γλωσσικές εισόδους ταυτόχρονα. Οι αριθμητικές είσοδοι ασαφοποιούνται στους κόμβους εισόδου, οι οποίοι συμπεριφέρονται ως εκπαιδευόμενοι ασαφοποιητές χαρακτηριστικών. Γνώση βασισμένη σε κανόνες μπορεί εύκολα να αποτυπωθεί κατευθείαν σε αρχιτεκτονική δικτύου, ενώ οι συνδέσεις στο δίκτυο αντιστοιχούν σε Γκαουσσιανά (Gauss) ασαφή σύνολα. Η καινοτομία του μοντέλου έγκειται:

- στο συνδυασμό ασαφοποιητών που ενημερώνονται βάση των χαρακτηριστικών εισόδου
- στην ταυτόχρονη ενεργοποίηση των κόμβων η οποία διαδίδεται διαμέσου του δικτύου, πράξη η οποία στηρίζεται στα ασαφή υποσύνολα
- στη χρήση του τελεστή γινομένου για τον υπολογισμό του βαθμού ενεργοποίησης ενός κανόνα
- στην χρήση της τροποποιημένης μεθόδου απο-ασαφοποίησης κέντρου βάρους ως μεθόδου παραγωγής αριθμητικών αποτελεσμάτων στους κόμβους εξόδου

Για την εκπαίδευση των κέντρων και των διασπορών των συγκεκριμένων συνδέσεων χρησιμοποιείται η μέθοδος κλίσης, μία μέθοδος επιβλεπόμενης μάθησης. Επίσης προτείνεται μια μέθοδος παραγωγής κανόνων από το εκπαιδευμένο δίκτυο που στηρίζεται στα ασαφή υποσύνολα.

Το SuPFuNIS μπορεί να εφαρμοστεί σε μια πληθώρα εφαρμογών και προβλημάτων, ενώ η απόδοση του, συγκριτικά με ήδη υπάρχοντα μοντέλα, είναι πολύ καλή.

4.2 Το SuPFuNIS σε Αντιδιαστολή με άλλα Νευρο-Ασαφή Μοντέλα

Η ανάπτυξη των νευρωνικών δικτύων έχει μια κοινή συνιστώσα που προέρχεται από την επιθυμία:

- να ενσωματωθεί η προκύπτουσα γνώση από τα δεδομένα στην αρχιτεκτονική του δικτύου ώστε να επιτυγχάνεται ταχύτερη εκπαίδευση
- να σχεδιασθεί ένας κατάλληλος μηχανισμός που να συνθέτει και να συσσωρεύει τα αποτελέσματα, αλλά ταυτόχρονα να μπορεί να χειρίζεται αριθμητικά και γλωσσικά δεδομένα ώστε να προκύπτουν έξοδοι ή να απορρέουν συμπεράσματα
- να ενσωματωθεί ένας μηχανισμός που θα ρυθμίζει βέλτιστα τους κανόνες βάσει της εκπαίδευσης από αριθμητικά δεδομένα

- να αποσπασθεί και να εξηγηθεί η προκύπτουσα γνώση με τη μορφή μιας βάσης γνώσης

Ας εξηγήσουμε περαιτέρω όμως τα παραπάνω:

- *Ενσωμάτωση της γνώσης που προκύπτει από τα δεδομένα*

Τα περισσότερα υβριδικά μοντέλα ενσωματώνουν τη γνώση που προκύπτει είτε από δεδομένα είτε από την εμπειρία με τη μορφή ασαφών κανόνων της μορφής EAN-TOTE, οι οποίοι τελικά απεικονίζονται στη δομή ενός νευρωνικού δικτύου. Αυτή η διαδικασία πραγματοποιείται συχνά υποθέτοντας ότι τόσο τα μέλη της υπόθεσης όσο και τα μέλη του συμπεράσματος των κανόνων της μορφής EAN-TOTE (antecedent and consequent labels of standard fuzzy if–then rules) παρουσιάζονται ως συναπτικά βάρη στο δίκτυο. Όπως προκύπτει από το [2], τα δίκτυα που στηρίζονται στη γνώση απαιτούν σχετικά μικρότερο μέγεθος συνόλου εκπαίδευσης για καλύτερη γενίκευση. Όταν τέτοιου είδους γνώση παράγεται από αριθμητικά δεδομένα, μια λύση είναι να χρησιμοποιηθεί είτε ομαδοποίηση (clustering) είτε διαμερισμός ώστε να προκύψουν οι κανόνες. Χρησιμοποιώντας ομαδοποίηση, τα κέντρα των ασαφών κανόνων αρχικοποιούνται ως διανύσματα κατηγοριών (cluster vectors), που προκύπτουν από το σύνολο των δεδομένων εισόδου. Συνεπώς, ένας αλγόριθμος μάθησης ρυθμίζει λεπτομερειακά αυτούς τους κανόνες βασιζόμενος στο διαθέσιμο σύνολο εκπαίδευσης που αναφέρεται στο πρόβλημα. Τεχνικές διαμέρισης διαιρούν περιοδικά το σύνολο των δεδομένων εισόδου - εξόδου σε μικρότερες περιοχές, βασιζόμενες κάθε φορά σε μια εκτίμηση του τοπικού μέσου τετραγωνικού σφάλματος. Κάθε διαμέριση οδηγεί σε ένα κανόνα της μορφής EAN-TOTE. Σε κάθε μία από αυτές τις τεχνικές, η επιλογή των αριθμών των κανόνων για να επιλυθεί το πρόβλημα είναι περισσότερο ή λιγότερο βασισμένη στην ευριστική προσέγγιση.

- *Σύνθεση και συσσώρευση αποτελεσμάτων*

Το θέμα της σύνθεσης των δεδομένων εισόδου με την ενσωματωμένη βάση κανόνων, εξαρτάται από το αν τα δεδομένα εισόδου είναι αριθμητικά ή γλωσσικά. Με αριθμητικές εισόδους, η συνηθισμένη περίπτωση είναι η χρησιμοποίηση των τιμών που υπολογίζονται από τις συναρτήσεις συμμετοχής των ασαφών συνόλων που απεικονίζουν τα βάρη του δικτύου. Για να γίνει χειρισμός των ασαφών δεδομένων εισόδου, το δοθέν υπερσύνολο αναφοράς κβαντικοποιείται σε προκαθορισμένα ασαφή σύνολα, οπότε μια ασαφής είσοδος είναι πλέον ένα από αυτά τα προκαθορισμένα ασαφή σύνολα.

- *Μάθηση*

Το ζήτημα αυτό αντιμετωπίζεται χρησιμοποιώντας επιβλεπόμενη μάθηση με την μέθοδο κλίσης (ή παραλλαγές αυτής) ή μη επιβλεπόμενη μάθηση ή ενισχυτική μάθηση ή ευριστικές μεθόδους ή γενετικούς αλγόριθμούς.

- *Ερμηνεία κανόνων*

Το τελευταίο ζήτημα της απόσπασης και ερμηνείας των ενημερωμένων ασαφών κανόνων επιτυγχάνεται αναθέτοντας σε κάθε ασαφές βάρος μια γλωσσική τιμή, που επιλέγεται στη βάση σύγκρισης με μια συλλογή καθορισμένων ασαφών συνόλων σύμφωνα με κριτήρια ομοιότητας.

Το νευρο-ασαφές σύστημα που υλοποιείται επιδιώκει να καλύψει ποικίλες απαιτήσεις στους εξής τομείς:

1. να αναπτύξει ένα μηχανισμό που θα μπορεί να χειριστεί αριθμητικές και γλωσσικές εισόδους
2. να δώσει βαρύτητα στον περιορισμό του αριθμού των παραμέτρων που χρησιμοποιεί το μοντέλο για να λύσει ένα συγκεκριμένο πρόβλημα
3. να υπάρχει η δυνατότητα για ενδεχόμενη ενσωμάτωση της γνώσης, που προκύπτει τόσο από τα δεδομένα όσο και από την εμπειρία, στην δημιουργία ενός αρχικού συνόλου κανόνων της μορφής EAN-TOTE
4. να γίνει προσπάθεια το σύστημα να μαθαίνει από την προκύπτουσα γνώση ώστε να ενημερώνει το σύνολο των κανόνων
5. να επιχειρηθεί η ερμηνεία ενός εκπαιδευμένου νευρο-ασαφούς συστήματος

Το νευρο-ασαφές σύστημα που προκύπτει, δηλαδή το SuPFuNIS, προσπαθεί να αντεπεξέλθει σε αυτούς τους τομείς. Το SuPFuNIS χρησιμοποιεί αρχιτεκτονική ενός κανονικού νευρο-ασαφούς δικτύου που ενσωματώνει ασαφείς κανόνες της μορφής EAN-TOTE ως κρυμμένο στρώμα κόμβων. Η ενσωμάτωση γίνεται αντιστοιχίζοντας τα μέλη της υπόθεσης των κανόνων (rule antecedents), με συνδέσεις από το στρώμα εισόδου στο κρυμμένο στρώμα, και τα μέλη του συμπεράσματος των κανόνων (rule consequents), με συνδέσεις από το κρυμμένο στρώμα στο στρώμα εξόδου. Γνώση στη μορφή των κανόνων EAN-TOTE που προκύπτει από ομαδοποιήσεις αριθμητικών δεδομένων χρησιμοποιείται για να αρχικοποιήσει τους ενσωματωμένους κανόνες του δικτύου. Ωστόσο το SuPFuNIS διαφέρει από τα άλλα νευρο-ασαφή δίκτυα σε διάφορα σημεία:

1. Χρησιμοποιεί ένα ασαφοποιητή εισόδου που ενημερώνεται και είναι υπεύθυνος για την ασαφοποίηση των αριθμητικών δεδομένων. Με άλλα λόγια, αριθμητικά δεδομένα ασαφοποιούνται χρησιμοποιώντας διασπορές Gauss ανάλογα με τα χαρακτηριστικά του προβλήματος.
2. Όλες οι πληροφορίες που διαδίδονται από το στρώμα εισόδου είναι ασαφείς. Έτσι το μοντέλο χρησιμοποιεί ένα μηχανισμό σύνθεσης που στηρίζεται στη σύγκριση του *μέτρου αμοιβαιότητας* ασαφών συνόλων για να καθορίσει την ενεργοποίηση που διαδίδεται σε ένα κόμβο κανόνα κατά μήκος μιας ασαφούς σύνδεσης.
3. Το μοντέλο συσσωρεύει τις ενεργοποιήσεις ενός κόμβου χρησιμοποιώντας το *ασαφές εσωτερικό γινόμενο*: ένα γινόμενο αμοιβαίων υποσυνόλων. Αυτό είναι διαφορετικό από την πολύ συνηθισμένη προσέγγιση που χρησιμοποιεί τον τελεστή min (ελαχίστου) των ασαφών συνόλων για να βρεθεί η συνολική ενεργοποίηση.
4. Οι έξοδοι δημιουργούνται χρησιμοποιώντας *ποσοτική απο-ασαφοποίηση* (volume defuzzification) η οποία αποτελεί παραλλαγή της συνηθισμένης τροποποιημένης μεθόδου κέντρου βάρους.

Όπως θα δειχθεί παρακάτω είναι ο συνδυασμός αυτών των στοιχείων που προσδίνουν στο μοντέλο υψηλές επιδόσεις και αυξημένη οικονομία στον αριθμό των χρησιμοποιούμενων παραμέτρων.

Προηγούμενες παραλλαγές του μοντέλου με εφαρμογές στα προβλήματα προσέγγισης συναρτήσεων, εξαγωγής συμπερασμάτων και ταξινόμησης έχουν παρουσιαστεί στα [3], [4] και [5]. Στο [3] ένας συνδυασμός από σταθμισμένες τιμές υποσυνόλων και ένας τροποποιημένος τελεστής ελαχίστου χρησιμοποιείται. Το μοντέλο υιοθετεί μια τριγωνική προσέγγιση αντί για Γκαουσσιανές συναρτήσεις συμμετοχής στους υπολογισμούς μεταξύ των ασαφών συνόλων. Απευθυνόταν σε εφαρμογές προσέγγισης συναρτήσεων και εξαγωγής συμπερασμάτων. Στο [4], το οποίο επεκτείνει το [3] αυξάνοντας τον αριθμό των ελεύθερων παραμέτρων,

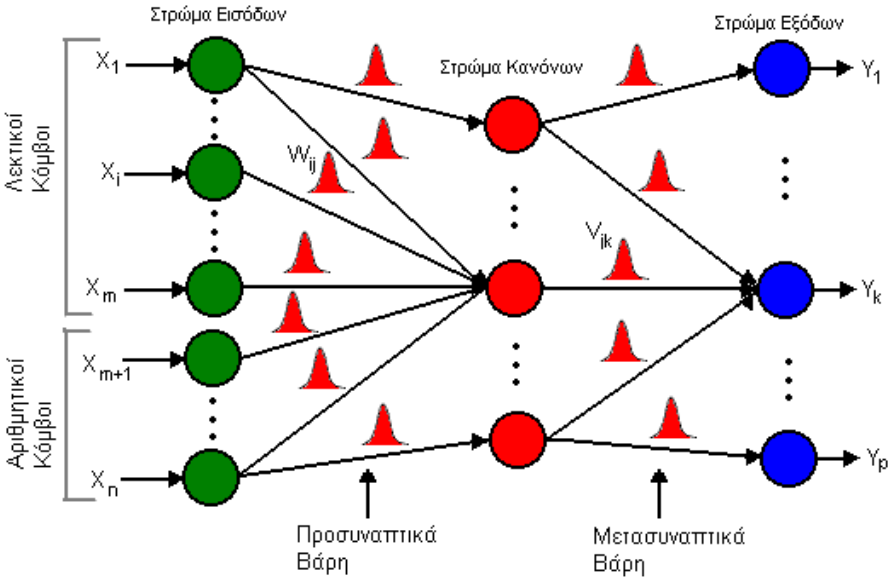
παρουσιάζεται μια απλή ευριστική μέθοδος για να υπολογιστεί ο αριθμός των κανόνων χρησιμοποιώντας ομαδοποίηση. Ένας συνδυασμός του βαθμού ομοιότητας και του τελεστή γινομένου με έναν ασαφοποιητή μη-εκπαιδευόμενο παρουσιάστηκε στο [5]. Το δίκτυο στο [5] χρησιμοποιεί Gauss ασαφή βάρη και προορίζεται για λύση προβλημάτων ταξινόμησης.

4.3 Αρχιτεκτονική και Λειτουργικές Λεπτομέρειες

Το μοντέλο SuPFuNIS ενσωματώνει απευθείας ασαφείς κανόνες της μορφής

ΕΑΝ *x_1 είναι ΧΑΜΗΛΟ και x_2 είναι ΥΨΗΛΟ* ***ΤΟΤΕ*** *y είναι ΜΕΣΑΙΟ*

όπου ΧΑΜΗΛΟ, ΥΨΗΛΟ, ΜΕΣΑΙΟ είναι καθορισμένα ασαφή σύνολα, αντίστοιχα προς τα υπερσύνολα αναφοράς τόσο των εισόδων όσο και των εξόδων. Οι κόμβοι εισόδου αντιπροσωπεύουν μεταβλητές ή χαρακτηριστικά του πεδίου ορισμού ενώ οι κόμβοι εξόδου αντιπροσωπεύουν μεταβλητές ή κατηγορίες του πεδίου τιμών. Κάθε κρυμμένος κόμβος αναπαριστά ένα κανόνα και συνδέσεις κόμβων εισόδων με κρυμμένους κόμβους αντιστοιχούν στα μέλη της υπόθεσης ενός ασαφούς κανόνα. Κάθε σύνδεση από κρυμμένο κόμβο προς κόμβο εξόδου αναπαριστά το μέλος του συμπεράσματος ενός ασαφούς κανόνα. Ασαφή σύνολα αντίστοιχα προς τις γλωσσικές τιμές των ασαφών κανόνων της μορφής EAN-TOTE (όπως ΧΑΜΗΛΟ,ΥΨΗΛΟ ή ΜΕΣΑΙΟ) προσδιορίζονται από τα υπερσύνολα αναφοράς των εισόδων και των εξόδων και απεικονίζονται από Gauss συναρτήσεις συμμετοχής (με παραμέτρους το κέντρο και την διασπορά). Τα προσυναπτικά ασαφή βάρη (antecedent weights) w_{ij} από τον κόμβο εισόδου i στον κόμβο κανόνα j μοντελοποιούνται από το κέντρο w_{ij}^c και τη διασπορά w_{ij}^σ ενός ασαφούς Gauss συνόλου και δηλώνονται ως $w_{ij}=(w_{ij}^c,w_{ij}^\sigma)$. Με παρόμοιο τρόπο, τα μετασυναπτικά ασαφή βάρη (consequent weights) από τον κόμβο κανόνα j στον κόμβο εξόδου k δηλώνονται ως $v_{jk}=(v_{jk}^c,v_{jk}^\sigma)$. Η γνώση που προκύπτει από τα δεδομένα στην μορφή των ασαφών κανόνων EAN-TOTE μετατρέπεται σε αρχιτεκτονική δικτύου όπως φαίνεται στο σχήμα που ακολουθεί.

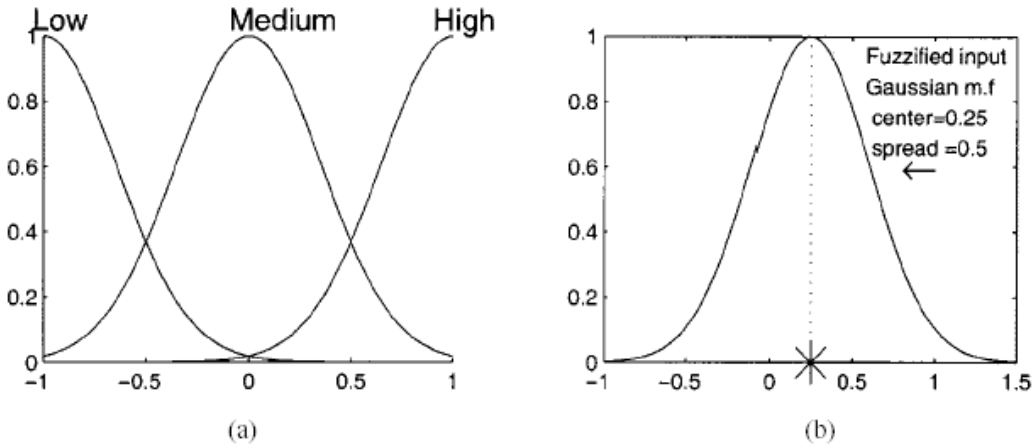


Σχήμα 4.3.1 Αρχιτεκτονική του SuPFuNIS μοντέλου

Το SuPFuNIS μπορεί ταυτόχρονα να δεχθεί αριθμητικές όσο και γλωσσικές εισόδους. Οι αριθμητικές είσοδοι ασαφοποιούνται ώστε οι είσοδοι στο δίκτυο να είναι συνολικά ασαφείς. Στην περίπτωση που τα προσυναπτικά βάρη είναι επίσης ασαφή, απαιτείται η υιοθέτηση μιας μεθόδου για την μετάδοση ενός ασαφούς σήματος διαμέσου ενός ασαφούς βάρους. Στα συμβατικά νευρωνικά δίκτυα οι αριθμητικές είσοδοι "μετασχηματίζονται" (scaled by) από τα βάρη και οι "μετασχηματισμένες" (scaled) τιμές συνθέτονται ως ενεργοποίηση των κόμβων, χρησιμοποιώντας απλή άθροιση. Στο μοντέλο SuPFuNIS η μετάδοση του σήματος διαμέσου των ασαφών βαρών πραγματοποιείται υπολογίζοντας το μέτρο ομοιότητας. Αναλυτικά:

1. Μετάδοση σήματος από τους κόμβους εισόδου

Εφόσον το διάνυσμα των χαρακτηριστικών εισόδου $X=\{x_1,\ldots,x_n\}$ μπορεί να αποτελείται είτε από αριθμητικά είτε από γλωσσικά δεδομένα, υπάρχουν δύο είδη κόμβων στο στρώμα εισόδου. Οι γλωσσικοί κόμβοι δέχονται γλωσσικά δεδομένα που αναπαριστούνται από ένα ασαφές σύνολο με συνάρτηση συμμετοχής την Γκαουσσιανή, η οποία εξαρτάται από το κέντρο x_i^c και τη διασπορά x_i^σ. Αυτές οι γλωσσικές είσοδοι μπορούν να επιλεγούν από προκαθορισμένα ασαφή σύνολα όπως φαίνεται στο Σχήμα 4.3.2 (α), όπου τρία ασαφή Gauss σύνολα έχουν οριστεί στο υπερσύνολο αναφοράς [-1,1].



Σχήμα 4.3.2 (α) Παράδειγμα προκαθορισμένου ασαφούς συνόλου για ασαφείς εισόδους (β) Παράδειγμα ασαφοποίησης αριθμητικών εισόδων από ένα μεταβαλλόμενο ασαφές σύνολο

Έτσι λοιπόν μια γλωσσική είσοδος x_i μπορεί να αναπαρασταθεί με το ζεύγος $x_i=(x_i^c,x_i^\sigma)$. Αυτό είναι επίσης και το σήμα $S(x_i) = x_i$ που μεταδίδεται από τον γλωσσικό κόμβο.

Οι αριθμητικοί κόμβοι είναι εκπαιδευόμενοι ασαφοποιητές συγκεκριμένοι για κάθε χαρακτηριστικό. Δέχονται αριθμητικά δεδομένα και τα ασαφοποιούν χρησιμοποιώντας ασαφή σύνολα Gauss. Οι αριθμητικές είσοδοι ασαφοποιούνται εάν αντιμετωπιστούν ως το κέντρο x_i^c μιας συνάρτησης συμμετοχής Gauss με ρυθμιζόμενη διασπορά x_i^σ. Όπως φαίνεται στο σχήμα 4.3.2 (β) όπου μια αριθμητική τιμή ενός χαρακτηριστικού ίση με 0.25 έχει ασαφοποιηθεί σε συνάρτηση συμμετοχής Gauss με κέντρο 0.25 και διασπορά 0.5. Το σχήμα της συνάρτησης συμμετοχής έχει επιλεχθεί ώστε να ταιριάζει στο σχήμα των Gauss ασαφών βαρών που αντιστοιχούν στους υπολογισμούς που θα παρουσιαστούν στο επόμενο μέρος. Γι' αυτό το σήμα που μεταδίδεται από ένα αριθμητικό κόμβο του στρώματος εισόδου μπορεί να αναπαρασταθεί και από το ζεύγος $S(x_i) = x_i =(x_i^c,x_i^\sigma)$. Αυτά τα διανύσματα από

αριθμητικά ή γλωσσικά δεδομένα μεταδίδονται στους κόμβους του κρυμμένου στρώματος διαμέσου των ασαφών βαρών $w_{ij}=(w_{ij}^{c},w_{ij}^{\sigma})$, δηλαδή διαμέσου των προσυναπτικών βαρών.

2. Μέτρο ομοιότητας

Από τη στιγμή που το σήμα και τα βάρη είναι ασαφή σύνολα, τα οποία αναπαρίστανται από συναρτήσεις συμμετοχής Gauss, χρειάζεται να βρεθεί η καθαρή τιμή του σήματος που μεταδόθηκε. Αυτή μπορεί να είναι ο βαθμός επικάλυψης μεταξύ των δύο ασαφών συνόλων. Αυτός μπορεί να υπολογιστεί από το **μέτρο ομοιότητας** που παρουσιάζεται στη συνέχεια.

Θεωρούμε δύο ασαφή σύνολα A και B που περιγράφονται από συναρτήσεις συμμετοχής Gauss με κέντρα c_1,c_2 και διασπορές σ_1,σ_2 αντίστοιχα.

$$a(x) = e^{-((x-c_1)/\sigma_1)^2}$$
$$b(x) = e^{-((x-c_2)/\sigma_2)^2} \quad (4.3.1)$$

Το μέτρο C(A) ενός ασαφούς συνόλου A ορίζεται ως:

$$C(A) = \int_{-\infty}^{\infty} a(x)\,dx = \int_{-\infty}^{\infty} e^{-((x-c_1)/\sigma_1)^2}\,dx \quad (4.3.2)$$

Οπότε το μέτρο ομοιότητας $\mathcal{E}$(A,B) μετρά το βαθμό στον οποίο το ασαφές σύνολο A ισούται με το ασαφές σύνολο B:

$$\mathcal{E}(A, B) = \text{Degree}(A = B)$$
$$= \text{Degree}(A \subseteq B \text{ and } B \subseteq A)$$

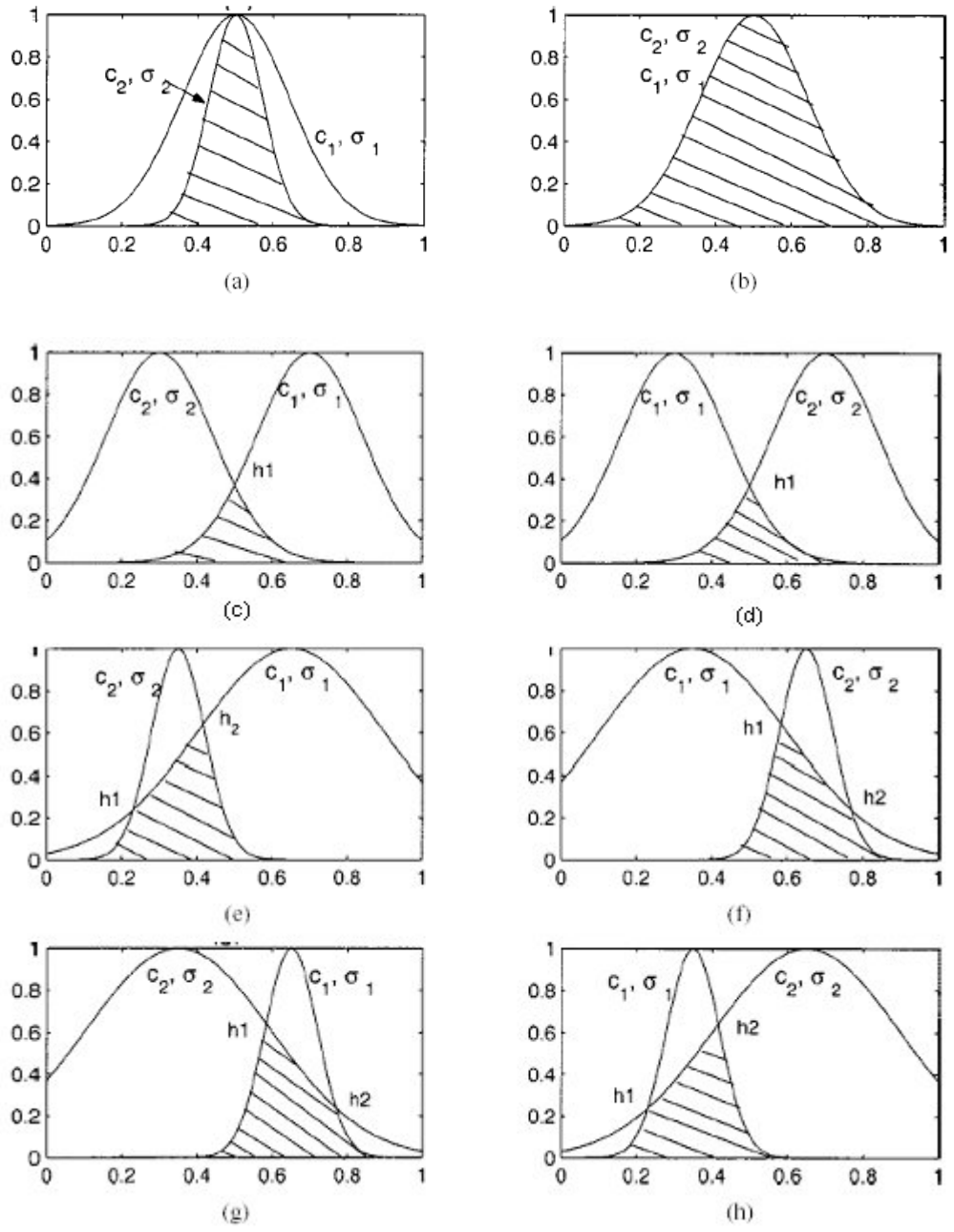
και μπορεί να διατυπωθεί [§ 2.1.6] ως:

$$\mathcal{E}(A, B) = \frac{C(A \cap B)}{C(A \cup B)}$$
$$= \frac{C(A \cap B)}{C(A) + C(B) - C(A \cap B)} \in [0, 1] \quad (4.3.3)$$

Το μέτρο ομοιότητας παίρνει τιμές στο διάστημα [0,1] και εξαρτάται από τις σχετικές τιμές των κέντρων και των διασπορών των ασαφών συνόλων A και B. Τέσσερις διαφορετικές περιπτώσεις επικάλυψης μπορούν να συμβούν:

- Περίπτωση 1: $c_1=c_2$, με τα σ_1,σ_2 να έχουν οποιοσδήποτε τιμές
- Περίπτωση 2: $c_1\neq c_2$,$\sigma_1=\sigma_2$
- Περίπτωση 3: $c_1\neq c_2$,$\sigma_1>\sigma_2$
- Περίπτωση 4: $c_1\neq c_2$,$\sigma_1<\sigma_2$

Αυτές οι τέσσερις περιπτώσεις εξαρτώνται από τις σχετικές τιμές των c_1,c_2,σ_1 και σ_2 όπως φαίνεται στο σχήμα 4.3.3.



Σχήμα 4.3.3 (α) $c_1=c_2$,$\sigma_1>\sigma_2$ (b) $c_1=c_2$,$\sigma_1=\sigma_2$ (c) $c_1>c_2$,$\sigma_1=\sigma_2$ (d) $c_1<c_2$,$\sigma_1=\sigma_2$ (e) $c_1>c_2$,$\sigma_1>\sigma_2$ (f) $c_1<c_2$,$\sigma_1>\sigma_2$, (g) $c_1>c_2$,$\sigma_1<\sigma_2$ (h) $c_1<c_2$,σ1<σ2

Παρατηρούμε ότι στην *Περίπτωση 1* τα δύο ασαφή σύνολα δεν τέμνονται, είτε επειδή το ένα σύνολο ανήκει τελείως στο άλλο είτε επειδή και τα δύο είναι ίσα. Στην *Περίπτωση 2* υπάρχει ακριβώς ένα σημείο τομής, ενώ στις *Περιπτώσεις 3* και *4* υπάρχουν ακριβώς δύο σημεία τομής.

Για να υπολογιστεί το σημείο τομής, θέτουμε α(x)=b(x) έτσι προκύπτουν τα σημεία h_1, h_2:

$$h_1 = \frac{c_1 + \frac{\sigma_1}{\sigma_2} c_2}{1 + \frac{\sigma_1}{\sigma_2}}$$

$$h_2 = \frac{c_1 - \frac{\sigma_1}{\sigma_2} c_2}{1 - \frac{\sigma_1}{\sigma_2}} \quad (4.3.4)$$

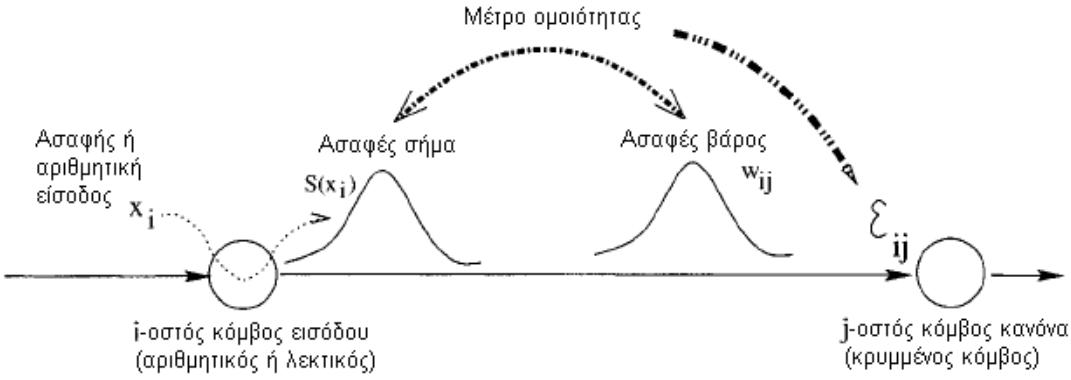
Για την αποτίμηση του $\mathcal{E}(A,B)$ αρκεί να υπολογισθούν οι τιμές του $C(A \cap B)$ σύμφωνα με τη σχέση (4.3.3). Όπως θα δειχθεί παρακάτω το μέτρο ενός ασαφούς συνόλου μπορεί να εκφραστεί σε όρους της γνωστής συνάρτησης σφάλματος:

$$\mathrm{erf}(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-(1/2)t^2}\, dt \quad (4.3.5)$$

η οποία έχει οριακές τιμές $\mathrm{erf}(-\infty)=-1/2$, $\mathrm{erf}(\infty)=1/2$ και $\mathrm{erf}(0)=0$.

3. Μετάδοση σήματος βασισμένη στο μέτρο ομοιότητας

Όπως φαίνεται παραστατικά στο σχήμα, το SuPFuNIS μεταδίδει ένα ασαφές σήμα από ένα κόμβο εισόδου διαμέσου ενός ασαφούς βάρους που αντιπροσωπεύει μια προσυναπτική σύνδεση.



Σχήμα 4.3.4 Μετάδοση ασαφούς σήματος

Το μεταδιδόμενο σήμα ποσοτικοποιείται κατά $\mathcal{E}_{ij}$, τιμή που δηλώνει το μέτρο ομοιότητας ανάμεσα στο ασαφές σήμα $S(x_i)$ και το ασαφές βάρος $(w_{ij}^c, w_{ij}^\sigma)$. Το $\mathcal{E}_{ij}$ υπολογίζεται από τον τύπο (4.3.3).

Συμβολικά, για ένα σήμα $s_i = S(x_i) = (x_i^c, x_i^\sigma)$, το οποίο έχει δημιουργηθεί από ένα αριθμητικό ή γλωσσικό κόμβο εισόδου, και για ένα ασαφές βάρος $w_{ij}=(w_{ij}^c, w_{ij}^\sigma)$ το μέτρο ομοιότητας ορίζεται ως:

$$\mathcal{E}_{ij} = \mathcal{E}(s_i, w_{ij}) = \frac{C(s_i \cap w_{ij})}{C(s_i) + C(w_{ij}) - C(s_i \cap w_{ij})} \quad (4.3.6)$$

Η ακριβής έκφραση του $C(s_i \cap w_{ij})$ δίδεται σε κάθε μία από τις παρακάτω τέσσερις περιπτώσεις:

- Περίπτωση 1: $x_i^c = w_{ij}^c$: Εάν $x_i^\sigma < w_{ij}^\sigma$ το ασαφές σύνολο σήματος s_i καλύπτεται πλήρως από το ασαφές βάρος w_{ij} όπως φαίνεται και στο σχήμα (4.3.3(a)) και το μέτρο $C(s_i \cap w_{ii}) = C(s_i)$:

$$\begin{aligned} \mathcal{C}(s_i \cap w_{ij}) &= \mathcal{C}(s_i) = \int_{-\infty}^{\infty} e^{-((x-x_i^c)/x_i^\sigma)^2}\, dx \\ &= x_i^\sigma \sqrt{\pi}\,[\mathrm{erf}(\infty) - \mathrm{erf}(-\infty)] = x_i^\sigma \sqrt{\pi} \end{aligned} \quad (4.3.7)$$

Παρόμοια $C(s_i \cap w_{ij}) = C(w_{ij})$ εάν $x_i^\sigma > w_{ij}^\sigma$ και $C(s_i \cap w_{ij}) = w_{ij}^\sigma \sqrt{\pi}$.
Εάν $x_i^\sigma = w_{ij}^\sigma$ τα δύο σύνολα είναι ίσα όπως φαίνεται και στο σχήμα 4.3.3(b). Συνοψίζοντας:

$$\mathcal{C}(s_i \cap w_{ij}) = \begin{cases} \mathcal{C}(s_i) = x_i^\sigma \sqrt{\pi} & \text{if } x_i^\sigma < w_{ij}^\sigma \\ \mathcal{C}(w_{ij}) = w_{ij}^\sigma \sqrt{\pi} & \text{if } x_i^\sigma > w_{ij}^\sigma \\ \mathcal{C}(s_i) = \mathcal{C}(w_{ij}) & \\ \quad = x_i^\sigma \sqrt{\pi} = w_{ij}^\sigma \sqrt{\pi} & \text{if } x_i^\sigma = w_{ij}^\sigma \end{cases} \quad (4.3.8)$$

- Περίπτωση 2: $x_i^c \neq w_{ij}^c$, $x_i^\sigma = w_{ij}^\sigma$.Στην περίπτωση αυτή υπάρχει ακριβώς ένα σημείο τομής h_1 όπως φαίνεται στα σχήματα (4.3.3(c)) και (4.3.3(d)). Θεωρώντας $x_i^c < w_{ij}^c$ (4.3.3(c)) το μέτρο $C(s_i \cap w_{ij})$ υπολογίζεται:

$$\begin{aligned} &\mathcal{C}(s_i \cap w_{ij}) \\ &= \int_{-\infty}^{h_1} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\, dx + \int_{h_1}^{\infty} e^{-((x-x_i^c)/x_i^\sigma)^2}\, dx \\ &= w_{ij}^\sigma \sqrt{\pi} \left[\frac{1}{2} + \mathrm{erf}\left(\frac{\sqrt{2}(h_1 - w_{ij}^c)}{w_{ij}^\sigma}\right)\right] \\ &\quad + x_i^\sigma \sqrt{\pi} \left[\frac{1}{2} - \mathrm{erf}\left(\frac{\sqrt{2}(h_1 - x_i^c)}{x_i^\sigma}\right)\right]. \end{aligned} \quad (4.3.9)$$

Θεωρώντας $x_i^c > w_{ij}^c$ (4.3.3(d)) η έκφραση για το μέτρο $C(s_i \cap w_{ij})$:

$$\begin{aligned} &\mathcal{C}(s_i \cap w_{ij}) \\ &= \int_{-\infty}^{h_1} e^{-((x-x_i^c)/x_i^\sigma)^2}\, dx + \int_{h_1}^{\infty} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\, dx \\ &= x_i^\sigma \sqrt{\pi} \left[\frac{1}{2} + \mathrm{erf}\left(\frac{\sqrt{2}(h_1 - x_i^c)}{x_i^\sigma}\right)\right] \\ &\quad + w_{ij}^\sigma \sqrt{\pi} \left[\frac{1}{2} - \mathrm{erf}\left(\frac{\sqrt{2}(h_1 - w_{ij}^c)}{w_{ij}^\sigma}\right)\right]. \end{aligned} \quad (4.3.10)$$

- Περίπτωση 3: $x_i^c \neq w_{ij}^c$, $x_i^\sigma < w_{ij}^\sigma$. Στην περίπτωση αυτή υπάρχουν ακριβώς δύο σημεία τομής h_1 και h_2 τα οποία υπολογίζονται με τη βοήθεια των τύπων (4.3.4) όπως φαίνεται στα σχήματα (4.3.3(e)) και (4.3.3(f)).
Υποθέτοντας $x_i^c > w_{ij}^c$, πράγμα που σημαίνει ότι $h_1 < h_2$ (4.3.3(e)) η έκφραση για το μέτρο $C(s_i \cap w_{ij})$ γίνεται:

$$\begin{aligned}
&\mathcal{C}(s_i \cap w_{ij}) \\
&= \int_{-\infty}^{h_1} e^{-((x-x_i^c)/x_i^\sigma)^2}\,dx + \int_{h_1}^{h_2} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\,dx \\
&\quad + \int_{h_2}^{\infty} e^{-((x-x_i^c)/x_i^\sigma)^2}\,dx \\
&= x_i^\sigma\sqrt{\pi}\left[\frac{1}{2} + \operatorname{erf}\left(\frac{\sqrt{2}(h_1 - x_i^c)}{x_i^\sigma}\right)\right] \\
&\quad + x_i^\sigma\sqrt{\pi}\left[\frac{1}{2} - \operatorname{erf}\left(\frac{\sqrt{2}(h_2 - x_i^c)}{x_i^\sigma}\right)\right] \\
&\quad + w_{ij}^\sigma\sqrt{\pi}\left[\operatorname{erf}\left(\frac{\sqrt{2}(h_2 - w_{ij}^c)}{w_{ij}^\sigma}\right) - \operatorname{erf}\left(\frac{\sqrt{2}(h_1 - w_{ij}^c)}{w_{ij}^\sigma}\right)\right].
\end{aligned} \tag{4.3.11}$$

Εάν $x_i^c < w_{ij}^c$, (4.3.3(f)) η έκφραση για το μέτρο $C(s_i \cap w_{ij})$ είναι ίδια με την (4.3.11). Αρκεί βέβαια να ισχύει ότι $h_1 < h_2$ σε αντίθετη περίπτωση οι τιμές των h_1, h_2 πρέπει να εναλλαχθούν.

- Περίπτωση 4: $x_i^c \neq w_{ij}^c$, $x_i^\sigma > w_{ij}^\sigma$. Στην περίπτωση αυτή (παρόμοια με την προηγούμενη) υπάρχουν ακριβώς δύο σημεία τομής h_1 και h_2 τα οποία υπολογίζονται με τη βοήθεια των τύπων (4.3.4) όπως φαίνεται στα σχήματα (4.3.3(g)) και (4.3.3(h)). Υποθέτοντας $x_i^c < w_{ij}^c$, πράγμα που σημαίνει ότι $h_1 < h_2$ (4.3.3(g)) η έκφραση για το μέτρο $C(s_i \cap w_{ij})$:

$$\begin{aligned}
&\mathcal{C}(s_i \cap w_{ij}) \\
&= \int_{-\infty}^{h_1} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\,dx + \int_{h_1}^{h_2} e^{-((x-x_i^c)/x_i^\sigma)^2}\,dx \\
&\quad + \int_{h_2}^{\infty} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\,dx \\
&= w_{ij}^\sigma\sqrt{\pi}\left[\operatorname{erf}\left(\frac{\sqrt{2}(h_1 - w_{ij}^c)}{w_{ij}^\sigma}\right) + \frac{1}{2}\right] \\
&\quad + w_{ij}^\sigma\sqrt{\pi}\left[\frac{1}{2} - \operatorname{erf}\left(\frac{\sqrt{2}(h_2 - w_{ij}^c)}{w_{ij}^\sigma}\right)\right] \\
&\quad + x_i^\sigma\sqrt{\pi}\left[\operatorname{erf}\left(\frac{\sqrt{2}(h_2 - x_i^c)}{x_i^\sigma}\right) - \operatorname{erf}\left(\frac{\sqrt{2}(h_1 - x_i^c)}{x_i^\sigma}\right)\right]
\end{aligned} \tag{4.3.12}$$

Εάν $x_i^c > w_{ij}^c$, (4.3.3(h)) η έκφραση για το μέτρο $C(s_i \cap w_{ij})$ είναι ίδια με την (4.3.12). Αρκεί βέβαια να ισχύει ότι $h_1 < h_2$ σε αντίθετη περίπτωση οι τιμές των h_1, h_2 πρέπει να εναλλαχθούν.

Οι αντίστοιχες εκφράσεις για το $\mathcal{E}_{ij}(s_i, w_{ij})$ προκύπτουν αν αντικατασταθεί το $C(s_i \cap w_{ij})$ από τους τύπους (4.3.8) έως (4.3.12) στη σχέση (4.3.6)

4. Συσσώρευση των ενεργοποιήσεων στους κόμβους των κανόνων με τη χρήση του ασαφούς εσωτερικού γινομένου

Μετρώντας όλες τις τιμές των μέτρων ομοιότητας $\mathbf{E}_j = \{\mathcal{E}_{1j}, \ldots, \mathcal{E}_{nj}\}$ για ένα κόμβο κανόνα j, στην ουσία αποτιμούμε την συμβατότητα ανάμεσα στο γλωσσικό διάνυσμα σήματος $S=\{s_1,\ldots,s_n\}$ (που μεταδίδεται από το στρώμα εισόδου) και στο διάνυσμα των ασαφών βαρών $W_j=\{w_{1j},\ldots,w_{nj}\}$ που καταλήγει στον κόμβο κανόνα j. Κάθε κόμβος κανόνα προβλέπεται να συσσωρεύει αυτό το διάνυσμα με τέτοιο τρόπο ώστε η ενεργοποίηση που προκύπτει να αντανακλά αυτή τη συμβατότητα. Με άλλα λόγια, η έκταση της πυροδότησης του κανόνα, όπως αυτή παρουσιάζεται μέσα από την ενεργοποίηση του κόμβου κανόνα, μετρά την έκταση με την οποία η σχετική γλωσσική είσοδος S αντιστοιχεί στο προσυναπτικό βάρος του προκειμένου κανόνα. Αντικαθιστούμε τον καθιερωμένο τελεστή ελαχίστου (min) που χρησιμοποιείται συχνά στα ασαφή συστήματα με τον τελεστή γινομένου για να συσσωρευτούν (αθροιστούν) οι ενεργοποιήσεις σε ένα κόμβο κανόνα. Η ενεργοποίηση z_j ενός κόμβου κανόνα j είναι ένα γινόμενο που βασίζεται στο μέτρο ομοιότητας, το οποίο επειδή είναι διαφορίσιμο επιτρέπει στο μοντέλο να χρησιμοποιεί μάθηση με την μέθοδο κλίσης.

Εν συντομία, η ενεργοποίηση του z_j του κόμβου κανόνα j είναι το γινόμενο όλων των μέτρων ομοιότητας, το **ασαφές εσωτερικό γινόμενο**:

$$z_j = \prod_{i=1}^{n} \mathcal{E}_{ij} = \prod_{i=1}^{n} \mathcal{E}(s_i, w_{ij}) \in [0, 1] \qquad (4.3.13)$$

Σημειώνουμε ότι το εσωτερικό γινόμενο στον τύπο (4.3.13) (οπότε και η συνάρτηση ενεργοποίησης κόμβου κανόνα) παρουσιάζει τις εξής ιδιότητες:

- βρίσκεται ανάμεσα στο 0 και στο 1
- είναι γνησίως αύξουσα
- συνεχής
- είναι μη-μοναδιαία

Η χρήση ενός τέτοιου ασαφούς εσωτερικού γινομένου προσδίδει βελτιωμένα χαρακτηριστικά στο SuPFuNIS.

Η συμπεριφορά του τελεστή γινομένου έχει αναλυθεί διεξοδικά στο [5], από όπου προκύπτει ότι ο τελεστής γινομένου δεν παραβλέπει πληροφορίες σχετικά με την διάσταση της εισόδου, όπως ο τελεστής ελαχίστου. Παρέχει μια καλύτερη εκτίμηση της συνεκτικής ισχύς (joint strength) των διαφόρων εισόδων. Επίσης σε ένα μεγάλο εύρος διασπορών, ο τελεστής γινομένου είναι ικανός να διακρίνει ξεκάθαρα ανάμεσα σε εισόδους που είναι παρόμοιες στο διάνυσμα βαρών και σε εισόδους που διαφέρουν από το διάνυσμα βαρών. Με άλλα λόγια, ο τελεστής γινομένου είναι ικανός για καλύτερη διάκριση από ότι ο τελεστής ελαχίστου. Επομένων είναι λογικό να θεωρείται ως σημαντικός συνεισφέρων παράγοντας στην υψηλή επίδοση και εξοικονόμηση πόρων του δικτύου SuPFuNIS.

Η συνάρτηση σήματος για ένα κόμβο κανόνα είναι γραμμική:

$$S(z_j)= z_j \qquad (4.3.14)$$

Αριθμητικές τιμές ενεργοποίησης μεταδίδονται αμετάβλητες στις μετασυναπτικές συνδέσεις.

5. Υπολογισμός του σήματος στρώματος εξόδου

Το σήμα κάθε κόμβου εξόδου καθορίζεται χρησιμοποιώντας την χωρική έκφραση της τροποποιημένης μεθόδου απο-ασαφοποίησης κέντρου βάρους. Ο όρος χωρική χρησιμοποιείται ώστε να συμπεριληφθούν πολυδιάστατες συναρτήσεις. Για δισδιάστατες συναρτήσεις, ο χώρος μειώνεται σε περιοχή.

Εάν η ενεργοποίηση του κόμβου εξόδου είναι y_k και τα V_{jk} δηλώνουν μετασυναπτικά σύνολα χώρων ενώ τα ξ_{jk} είναι οι παράμετροι που κανονικοποιούν τα z_j, τότε η γενική έκφραση απο-ασαφοποίησης είναι:

$$y_k = \frac{\sum_{j=1}^{q} z_j v_{jk}^c V_{jk} \xi_{jk}}{\sum_{j=1}^{q} z_j V_{jk} \xi_{jk}} \qquad (4.3.15)$$

όπου q είναι ο αριθμός των κανόνων κόμβων. Ο χώρος V_{jk} στην συγκεκριμένη περίπτωση είναι απλά η περιοχή των μετασυναπτικών βαρών που αναπαρίστανται από ασαφή σύνολα Gauss. Έτσι ισχύει η ισότητα $V_{jk}=\upsilon_{jk}^{\sigma}\sqrt{\pi}$ **.** Εάν τα βάρη ξ_{jk} θεωρηθούν μοναδιαία τότε:

$$y_k = \frac{\sum_{j=1}^{q} z_j v_{jk}^c v_{jk}^\sigma}{\sum_{j=1}^{q} z_j v_{jk}^\sigma} \qquad (4.3.16)$$

Το σήμα του κόμβου εξόδου k είναι $S(y_k)=y_k$. Με τις αντικαταστάσεις $\beta_{jk}=z_j\upsilon_{jk}^{c}$ και $\beta=\sum_{j=1}^{q}\beta_{jk}$ μπορεί να απλοποιηθεί σε:

$$y_k = \frac{\sum_{j=1}^{q} \beta_{jk} v_{jk}^c}{\beta} = \sum_{j=1}^{q} \hat{\beta}_{jk} v_{jk}^c \qquad (4.3.17)$$

όπου οι συντελεστές $\beta'_{jk}= \beta_{jk} / b$ είναι κανονικοποιημένοι με άθροισμα την μονάδα.

Ο απο-ασαφοποιητής (4.3.16) συνεπώς ουσιαστικά υπολογίζει το κυρτό άθροισμα του συνόλου των μετασυναπτικών κέντρων.

4.4 Επιβλεπόμενη Μάθηση

Το δίκτυο SuPFuNIS εκπαιδεύεται βάσει επιβλεπόμενης μάθησης. Το γεγονός αυτό μεταξύ άλλων εμπλέκει επαναλαμβανόμενη παρουσίαση ενός συνόλου από χαρακτηριστικά εισόδου που έχουν επιλεγεί από το σύνολο εκπαίδευσης. Η έξοδος

του δικτύου συγκρίνεται με την επιθυμητή τιμή για να προκύψει το σφάλμα, τα δε βάρη του νευρωνικού δικτύου αλλάζουν βάσει των αντίστοιχων κριτηρίων ελαχιστοποίησης σφάλματος. Εφόσον το νευρωνικό δίκτυο εκπαιδευτεί στην επιθυμητή τιμή σφάλματος, δοκιμάζεται εφαρμόζοντας ένα νέο σύνολο από χαρακτηριστικά εισόδου που έχουν επιλεγεί από το σύνολο ελέγχου.

1. Επαναληπτική ανανέωση των βαρών

Η μάθηση στο SuPFuNIS επιτυγχάνεται χρησιμοποιώντας την μέθοδο κλίσης (gradient descent method). Το κριτήριο τετραγωνικού σφάλματος χρησιμοποιείται σαν παράμετρος μέτρησης της επίδοσης εκπαίδευσης. Το τετραγωνικό σφάλμα e(t) στην επανάληψη t υπολογίζεται με συγκεκριμένο τρόπο.

$$e(t) = \frac{1}{2} \sum_{k=1}^{p} (d_k(t) - S(y_k(t)))^2 \qquad (4.4.1)$$

όπου $\mathbf{d_k(t)}$ είναι η επιθυμητή τιμή για στον κόμβο εξόδου **k** και το σφάλμα αθροίζεται για όλες τις **p** εξόδους που αφορούν ένα συγκεκριμένο χαρακτηριστικό **X(t)**. Για ταξινόμηση 1-από-K, οι επιθυμητές έξοδοι θα είναι μηδέν ή ένα. Τόσο τα κέντρα w_{ij}^c, υ_{jk}^c όσο και οι διασπορές w_{ij}^σ, υ_{jk}^σ των προσυναπτικών και μετασυναπτικών συνδέσεων και οι διασπορές των κόμβων εισόδου x_i^σ μεταβάλλονται στη βάση των παρακάτω τύπων:

$$\begin{aligned}
w_{ij}^c(t+1) &= w_{ij}^c(t) - \eta \frac{\partial e(t)}{\partial w_{ij}^c(t)} + \alpha \Delta w_{ij}^c(t-1) \\
v_{jk}^c(t+1) &= v_{jk}^c(t) - \eta \frac{\partial e(t)}{\partial v_{jk}^c(t)} + \alpha \Delta v_{jk}^c(t-1) \\
w_{ij}^\sigma(t+1) &= w_{ij}^\sigma(t) - \eta \frac{\partial e(t)}{\partial w_{ij}^\sigma(t)} + \alpha \Delta w_{ij}^\sigma(t-1) \\
v_{jk}^\sigma(t+1) &= v_{jk}^\sigma(t) - \eta \frac{\partial e(t)}{\partial v_{jk}^\sigma(t)} + \alpha \Delta v_{jk}^\sigma(t-1) \\
x_i^\sigma(t+1) &= x_i^\sigma(t) - \eta \frac{\partial e(t)}{\partial x_i^\sigma(t)} + \alpha \Delta x_i^\sigma(t-1)
\end{aligned} \qquad (4.4.2)$$

όπου **η** είναι ο ρυθμός μάθησης και **α** είναι ο όρος ορμής επιπλέον:

$$\begin{aligned}
\Delta w_{ij}^c(t-1) &= w_{ij}^c(t) - w_{ij}^c(t-1) \\
\Delta v_{jk}^c(t-1) &= v_{jk}^c(t) - v_{jk}^c(t-1) \\
\Delta w_{ij}^\sigma(t-1) &= w_{ij}^\sigma(t) - w_{ij}^\sigma(t-1) \\
\Delta v_{jk}^\sigma(t-1) &= v_{jk}^\sigma(t) - v_{jk}^\sigma(t-1) \\
\Delta x_i^\sigma(t-1) &= x_i^\sigma(t) - x_i^\sigma(t-1).
\end{aligned} \qquad (4.4.3)$$

2. Υπολογισμός των μερικών παραγώγων

Οι εκφράσεις των μερικών παραγώγων που απαιτούνται σε αυτές τις ενημερώσεις υπολογίζονται με τον τρόπο που παρουσιάζεται στην συνέχεια. Για την παράγωγο του σφάλματος ως προς τα μετασυναπτικά κέντρα:

$$\frac{\partial e}{\partial v_{jk}^{c}} = \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial v_{jk}^{c}} = -(d_k - y_k)\frac{z_j v_{jk}^{\sigma}}{\sum_{j=1}^{q} z_j v_{jk}^{\sigma}} \quad (4.4.4)$$

και η μερική παράγωγος του σφάλματος ως προς τις μετασυναπτικές διασπορές:

$$\begin{aligned}\frac{\partial e}{\partial v_{jk}^{\sigma}} &= \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial v_{jk}^{\sigma}} \\ &= -(d_k - y_k)\frac{z_j v_{jk}^{c}\sum_{j=1}^{q} z_j v_{jk}^{\sigma} - z_j \sum_{j=1}^{q} z_j v_{jk}^{c} v_{jk}^{\sigma}}{\left(\sum_{j=1}^{q} z_j v_{jk}^{\sigma}\right)^2}\end{aligned} \quad (4.4.5)$$

Οι παράγωγοι του σφάλματος ως προς τα προσυναπτικά κέντρα και τις προσυναπτικές διασπορές εμπεριέχουν μερικές παραγώγους υποσυνόλων με σειρές και είναι πιο περίπλοκες να υπολογιστούν. Ειδικά, οι σειρές των μερικών παραγώγων σφάλματος ως προς τα προσυναπτικά κέντρα και τις προσυναπτικές διασπορές είναι:

$$\begin{aligned}\frac{\partial e}{\partial w_{ij}^{c}} &= \sum_{k=1}^{p} \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \mathcal{E}_{ij}}\frac{\partial \mathcal{E}_{ij}}{\partial w_{ij}^{c}} \\ &= \sum_{k=1}^{p} -(d_k - y_k)\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \mathcal{E}_{ij}}\frac{\partial \mathcal{E}_{ij}}{\partial w_{ij}^{c}}\end{aligned} \quad (4.4.6)$$

και:

$$\begin{aligned}\frac{\partial e}{\partial w_{ij}^{\sigma}} &= \sum_{k=1}^{p} \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \mathcal{E}_{ij}}\frac{\partial \mathcal{E}_{ij}}{\partial w_{ij}^{\sigma}} \\ &= \sum_{k=1}^{p} -(d_k - y_k)\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \mathcal{E}_{ij}}\frac{\partial \mathcal{E}_{ij}}{\partial w_{ij}^{\sigma}}\end{aligned} \quad (4.4.7)$$

ακόμη η σειρά της παραγώγου του σφάλματος ως προς τις διασπορές των χαρακτηριστικών εισόδου προκύπτει να είναι:

$$\begin{aligned}\frac{\partial e}{\partial x_i^{\sigma}} &= \sum_{j=1}^{q}\sum_{k=1}^{p} \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \mathcal{E}_{ij}}\frac{\partial \mathcal{E}_{ij}}{\partial x_i^{\sigma}} \\ &= \sum_{j=1}^{q}\sum_{k=1}^{p} -(d_k - y_k)\frac{\partial y_k}{\partial z_j}\frac{\partial z_j}{\partial \mathcal{E}_{ij}}\frac{\partial \mathcal{E}_{ij}}{\partial x_i^{\sigma}}\end{aligned} \quad (4.4.8)$$

όπου:

$$\frac{\partial y_k}{\partial z_j} = \frac{v_{jk}^{\sigma} v_{jk}^{c} \sum_{j=1}^{q} z_j v_{jk}^{\sigma} - v_{jk}^{\sigma} \sum_{j=1}^{q} z_j v_{jk}^{c} v_{jk}^{\sigma}}{\left(\sum_{j=1}^{q} z_j v_{jk}^{\sigma} \right)^2}$$

$$= v_{jk}^{\sigma} \left[\frac{v_{jk}^{c} \sum_{j=1}^{q} z_j v_{jk}^{\sigma} - \sum_{j=1}^{q} z_j v_{jk}^{c} v_{jk}^{\sigma}}{\left(\sum_{j=1}^{q} z_j v_{jk}^{\sigma} \right)^2} \right]$$

$$= \frac{v_{jk}^{\sigma} (v_{jk}^{c} - y_k)}{\sum_{j=1}^{q} z_j v_{jk}^{\sigma}} \qquad (4.4.9)$$

και

$$\frac{\partial z_j}{\partial \mathcal{E}_{ij}} = \prod_{\substack{i=1 \\ i \neq i}}^{n} \mathcal{E}_{ij} \qquad (4.4.10)$$

Οι εκφράσεις των μερικών παραγώγων των μέτρων ομοιοτήτων των προσυναπτικών συνδέσεων $\partial\mathcal{E}_{ij}/\partial w_{ij}^{c}$, $\partial\mathcal{E}_{ij}/\partial w_{ij}^{\sigma}$ και $\partial\mathcal{E}_{ij}/\partial x_i^{\sigma}$ προκύπτουν παραγωγίζοντας την (4.3.6) ως προς w_{ij}^{c}, w_{ij}^{σ} και x_i^{σ} οπότε προκύπτει:

$$\frac{\partial \mathcal{E}_{ij}}{\partial w_{ij}^{c}} = \frac{\left(\frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^{c}} \left(\sqrt{\pi}(x_i^{\sigma} + w_{ij}^{\sigma}) - C(s_i \cap w_{ij}) \right) \right) - \left(\frac{-\partial C(s_i \cap w_{ij})}{\partial w_{ij}^{c}} C(s_i \cap w_{ij}) \right)}{\left(\sqrt{\pi}(x_i^{\sigma} + w_{ij}^{\sigma}) - C(s_i \cap w_{ij}) \right)^2}$$

$$\frac{\partial \mathcal{E}_{ij}}{\partial w_{ij}^{\sigma}} = \frac{\left(\frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^{\sigma}} \left(\sqrt{\pi}(x_i^{\sigma} + w_{ij}^{\sigma}) - C(s_i \cap w_{ij}) \right) \right) - \left(\left(\sqrt{\pi} - \frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^{\sigma}} \right) C(s_i \cap w_{ij}) \right)}{\left(\sqrt{\pi}(x_i^{\sigma} + w_{ij}^{\sigma}) - C(s_i \cap w_{ij}) \right)^2}$$

$$\frac{\partial \mathcal{E}_{ij}}{\partial x_i^{\sigma}} = \frac{\left(\frac{\partial C(s_i \cap w_{ij})}{\partial x_i^{\sigma}} \left(\sqrt{\pi}(w_{ij}^{\sigma} + x_i^{\sigma}) - C(s_i \cap w_{ij}) \right) \right) - \left(\left(\sqrt{\pi} - \frac{\partial C(s_i \cap w_{ij})}{\partial x_i^{\sigma}} \right) C(s_i \cap w_{ij}) \right)}{\left(\sqrt{\pi}(w_{ij}^{\sigma} + x_i^{\sigma}) - C(s_i \cap w_{ij}) \right)^2}$$

(4.4.11)

Στις προηγούμενες σχέσεις οι μερικές παράγωγοι: $\partial C(s_i \cap w_{ij})/\partial w_{ij}^{c}$, $\partial C(s_i \cap w_{ij})/\partial w_{ij}^{\sigma}$ και $\partial C(s_i \cap w_{ij})/\partial x_i^{\sigma}$ εξαρτώνται από τη φύση της επικάλυψης του ασαφούς συνόλου του χαρακτηριστικού εισόδου με το ασαφές σύνολο του βάρους, δηλαδή από τις τιμές των w_{ij}^{c}, x_i^{c} ,w_{ij}^{σ} και x_i^{σ}. Έτσι προκύπτουν σχέσεις συγκεκριμένες για κάθε περίπτωση:

Περίπτωση 1: $x_i^{c} = w_{ij}^{c}$:

Όπως είναι φανερό από την (4.3.8) το $C(s_i \cap w_{ij})$ δεν εξαρτάται από το w_{ij}^{c}:

$$\frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^{c}} = 0 \qquad (4.4.12)$$

Παραγωγίζοντας ως προς w_{ij}^{σ}:

$$\frac{\partial \mathcal{C}(s_i \cap w_{ij})}{\partial w_{ij}^{\sigma}} = \begin{cases} \sqrt{\pi} & \text{if } w_{ij}^{c} = x_i^{c} \text{ and } w_{ij}^{\sigma} \leq x_i^{\sigma} \\ 0 & \text{if } w_{ij}^{c} = x_i^{c} \text{ and } w_{ij}^{\sigma} > x_i^{\sigma} \end{cases} \quad (4.4.13)$$

και ως προς x_i^{σ}:

$$\frac{\partial \mathcal{C}(s_i \cap w_{ij})}{\partial x_i^{\sigma}} = \begin{cases} 0 & \text{if } w_{ij}^{c} = x_i^{c} \text{ and } w_{ij}^{\sigma} < x_i^{\sigma} \\ \sqrt{\pi} & \text{if } w_{ij}^{c} = x_i^{c} \text{ and } w_{ij}^{\sigma} \geq x_i^{\sigma} \end{cases} \quad (4.4.14)$$

Περίπτωση 2: $x_i^{c} \neq w_{ij}^{c}$, $w_{ij}^{\sigma} = x_i^{\sigma}$:

Εάν $x_i^{c} < w_{ij}^{c}$, οι $\partial C(s_i \cap w_{ij})/\partial w_{ij}^{c}$, $\partial C(s_i \cap w_{ij})/\partial w_{ij}^{\sigma}$ και $\partial C(s_i \cap w_{ij})/\partial x_i^{\sigma}$ προκύπτουν παραγωγίζοντας την (4.3.9) και προκύπτει:

$$\begin{aligned} \frac{\partial \mathcal{C}(s_i \cap w_{ij})}{\partial w_{ij}^{c}} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^{c}} e^{-((x-w_{ij}^{c})/w_{ij}^{\sigma})^2}\, dx \\ &\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial w_{ij}^{c}} e^{-((x-x_i^{c})/x_i^{\sigma})^2}\, dx \\ &= \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^{c}} e^{-((x-w_{ij}^{c})/w_{ij}^{\sigma})^2}\, dx \\ &= -e^{-((h_1-w_{ij}^{c})/w_{ij}^{\sigma})^2} \end{aligned} \quad (4.4.15)$$

$$\begin{aligned} \frac{\partial \mathcal{C}(s_i \cap w_{ij})}{\partial w_{ij}^{\sigma}} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^{\sigma}} e^{-((x-w_{ij}^{c})/w_{ij}^{\sigma})^2}\, dx \\ &\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial w_{ij}^{\sigma}} e^{-((x-x_i^{c})/x_i^{\sigma})^2}\, dx \\ &= -\frac{h_1 - w_{ij}^{c}}{w_{ij}^{\sigma}} e^{-((h_1-w_{ij}^{c})/w_{ij}^{\sigma})^2} \\ &\quad + \sqrt{\pi}\left[\frac{1}{2} + \operatorname{erf}\left(\frac{\sqrt{2}(h_1 - w_{ij}^{c})}{w_{ij}^{\sigma}}\right)\right] \end{aligned} \quad (4.4.16)$$

$$\begin{aligned} \frac{\partial \mathcal{C}(s_i \cap w_{ij})}{\partial x_i^{\sigma}} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial x_i^{\sigma}} e^{-((x-w_{ij}^{c})/w_{ij}^{\sigma})^2}\, dx \\ &\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial x_i^{\sigma}} e^{-((x-x_i^{c})/x_i^{\sigma})^2}\, dx \\ &= \frac{h_1 - x_i^{c}}{x_i^{\sigma}} e^{-((h_1-x_i^{c})/x_i^{\sigma})^2} \\ &\quad + \sqrt{\pi}\left[\frac{1}{2} - \operatorname{erf}\left(\frac{\sqrt{2}(h_1 - x_i^{c})}{x_i^{\sigma}}\right)\right] \end{aligned} \quad (4.4.17)$$

Εάν $x_i^c > w_{ij}^c$, οι $\partial C(s_i \cap w_{ij})/\partial w_{ij}^c$, $\partial C(s_i \cap w_{ij})/\partial w_{ij}^\sigma$ και $\partial C(s_i \cap w_{ij})/\partial x_i^\sigma$ προκύπτουν παραγωγίζοντας την (4.3.10) έτσι προκύπτει:

$$\begin{aligned}\frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^c} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^c} e^{-((x-x_i^c)/x_i^\sigma)^2}\, dx \\ &\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial w_{ij}^c} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\, dx \\ &= e^{-((h_1-w_{ij}^c)/w_{ij}^\sigma)^2}\end{aligned} \tag{4.4.18}$$

$$\begin{aligned}\frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^\sigma} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^\sigma} e^{-((x-x_i^c)/x_i^\sigma)^2}\, dx \\ &\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial w_{ij}^\sigma} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\, dx \\ &= \frac{h_1 - w_{ij}^c}{w_{ij}^\sigma} e^{-((h_1-w_{ij}^c)/w_{ij}^\sigma)^2} \\ &\quad + \sqrt{\pi}\left[\frac{1}{2} - \mathrm{erf}\left(\frac{\sqrt{2}(h_1 - w_{ij}^c)}{w_{ij}^\sigma}\right)\right]\end{aligned} \tag{4.4.19}$$

$$\begin{aligned}\frac{\partial C(s_i \cap w_{ij})}{\partial x_i^\sigma} &= \int_{-\infty}^{h_1} \frac{\partial}{\partial x_i^\sigma} e^{-((x-x_i^c)/x_i^\sigma)^2}\, dx \\ &\quad + \int_{h_1}^{\infty} \frac{\partial}{\partial x_i^\sigma} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\, dx \\ &= -\frac{h_1 - x_i^c}{x_i^\sigma} e^{-((h_1-x_i^c)/x_i^\sigma)^2} \\ &\quad + \sqrt{\pi}\left[\frac{1}{2} + \mathrm{erf}\left(\frac{\sqrt{2}(h_1 - x_i^c)}{x_i^\sigma.}\right)\right]\end{aligned} \tag{4.4.20}$$

Περίπτωση 3: $x_i^c \neq w_{ij}^c$, $w_{ij}^\sigma > x_i^\sigma$:

Εάν $x_i^c > w_{ij}^c$, οι $\partial C(s_i \cap w_{ij})/\partial w_{ij}^c$, $\partial C(s_i \cap w_{ij})/\partial w_{ij}^\sigma$ και $\partial C(s_i \cap w_{ij})/\partial x_i^\sigma$ προκύπτουν παραγωγίζοντας την (4.3.11) και προκύπτουν:

$$\begin{aligned}&\frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^c} \\ &= \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^c} e^{-((x-x_i^c)/x_i^\sigma)^2}\, dx \\ &\quad + \int_{h_1}^{h_2} \frac{\partial}{\partial w_{ij}^c} e^{-((x-w_{ij}^c)/w_{ij}^\sigma)^2}\, dx \\ &\quad + \int_{h_2}^{\infty} \frac{\partial}{\partial w_{ij}^c} e^{-((x-x_i^c)/x_i^\sigma)^2}\, dx \\ &= -e^{-((h_2-w_{ij}^c)/w_{ij}^\sigma)^2} + e^{-((h_1-w_{ij}^c)/w_{ij}^\sigma)^2}\end{aligned} \tag{4.4.21}$$

$$\frac{\partial \mathcal{C}(s_i \cap w_{ij})}{\partial w_{ij}^{\sigma}} = \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^{\sigma}} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx + \int_{h_1}^{h_2} \frac{\partial}{\partial w_{ij}^{\sigma}} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx + \int_{h_2}^{\infty} \frac{\partial}{\partial w_{ij}^{\sigma}} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx$$

$$= \frac{h_1 - w_{ij}^c}{w_{ij}^{\sigma}} e^{-((h_1-w_{ij}^c)/w_{ij}^{\sigma})^2} - \frac{h_2 - w_{ij}^c}{w_{ij}^{\sigma}} \cdot e^{-((h_2-w_{ij}^c)/w_{ij}^{\sigma})^2} + \sqrt{\pi}\left(-\mathrm{erf}\left(\frac{\sqrt{2}(h_1 - w_{ij}^c)}{w_{ij}^{\sigma}}\right) + \mathrm{erf}\left(\frac{\sqrt{2}(h_2 - w_{ij}^c)}{w_{ij}^{\sigma}}\right)\right) \quad (4.4.22)$$

$$\frac{\partial \mathcal{C}(s_i \cap w_{ij})}{\partial x_i^{\sigma}} = \int_{-\infty}^{h_1} \frac{\partial}{\partial x_i^{\sigma}} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx + \int_{h_1}^{h_2} \frac{\partial}{\partial x_i^{\sigma}} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx + \int_{h_2}^{\infty} \frac{\partial}{\partial x_i^{\sigma}} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx$$

$$= -\frac{h_1 - x_i^c}{x_i^{\sigma}} e^{-((h_1-x_i^c)/x_i^{\sigma})^2} + \frac{h_2 - x_i^c}{x_i^{\sigma}} e^{-((h_2-x_i^c)/x_i^{\sigma})^2} + \sqrt{\pi}\left(1 + \mathrm{erf}\left(\frac{\sqrt{2}(h_1 - x_i^c)}{x_i^{\sigma}}\right) - \mathrm{erf}\left(\frac{\sqrt{2}(h_2 - x_i^c)}{x_i^{\sigma}}\right)\right) \quad (4.4.23)$$

Ομοίως αν $x_i^c < w_{ij}^c$:

$$\frac{\partial \mathcal{C}(s_i \cap w_{ij})}{\partial w_{ij}^{c}} = \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^{c}} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx + \int_{h_1}^{h_2} \frac{\partial}{\partial w_{ij}^{c}} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx + \int_{h_2}^{\infty} \frac{\partial}{\partial w_{ij}^{c}} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx$$

$$= -e^{-((h_2-w_{ij}^c)/w_{ij}^{\sigma})^2} + e^{-((h_1-w_{ij}^c)/w_{ij}^{\sigma})^2}. \quad (4.4.24)$$

Παρατηρούμε ότι είτε $x_i^c > w_{ij}^c$ είτε $x_i^c < w_{ij}^c$ οι παραστάσεις του $\partial C(s_i \cap w_{ij}) / \partial w_{ij}^c$ που προκύπτουν είναι ίδιες, εφόσον βέβαια ισχύει $h_1 < h_2$. Παρόμοιες

παρατηρήσεις προκύπτουν για τις παραστάσεις των $\partial C(s_i \cap w_{ij})/\partial w_{ij}^{\sigma}$ και $\partial C(s_i \cap w_{ij})/\partial x_i^{\sigma}$.

Περίπτωση 4: $x_i^c \neq w_{ij}^c$, $w_{ij}^{\sigma} < x_i^{\sigma}$:

Εάν $x_i^c < w_{ij}^c$, οι $\partial C(s_i \cap w_{ij})/\partial w_{ij}^c$, $\partial C(s_i \cap w_{ij})/\partial w_{ij}^{\sigma}$ και $\partial C(s_i \cap w_{ij})/\partial x_i^{\sigma}$ προκύπτουν παραγωγίζοντας την (4.3.12) έτσι προκύπτουν:

$$\begin{aligned}
&\frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^c} \\
&= \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^c} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx \\
&\quad + \int_{h_1}^{h_2} \frac{\partial}{\partial w_{ij}^c} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx \\
&\quad + \int_{h_2}^{\infty} \frac{\partial}{\partial w_{ij}^c} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx \\
&= -e^{-((h_1-w_{ij}^c)/w_{ij}^{\sigma})^2} + e^{-((h_2-w_{ij}^c)/w_{ij}^{\sigma}))^2}
\end{aligned} \tag{4.4.25}$$

$$\begin{aligned}
&\frac{\partial C(s_i \cap w_{ij})}{\partial w_{ij}^{\sigma}} \\
&= \int_{-\infty}^{h_1} \frac{\partial}{\partial w_{ij}^{\sigma}} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx \\
&\quad + \int_{h_1}^{h_2} \frac{\partial}{\partial w_{ij}^{\sigma}} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx \\
&\quad + \int_{h_2}^{\infty} \frac{\partial}{\partial w_{ij}^{\sigma}} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx \\
&= -\frac{h_1-w_{ij}^c}{w_{ij}^{\sigma}} e^{-((h_1-w_{ij}^c)/w_{ij}^{\sigma})^2} + \frac{h_2-w_{ij}^c}{w_{ij}^{\sigma}} e^{-((h_2-w_{ij}^c)/w_{ij}^{\sigma})^2} \\
&\quad + \sqrt{\pi}\left(1 + \operatorname{erf}\left(\frac{\sqrt{2}(h_1-w_{ij}^c)}{w_{ij}^{\sigma}}\right)\right. \\
&\qquad \left. - \operatorname{erf}\left(\frac{\sqrt{2}(h_2-w_{ij}^c)}{w_{ij}^{\sigma}}\right)\right)
\end{aligned} \tag{4.4.26}$$

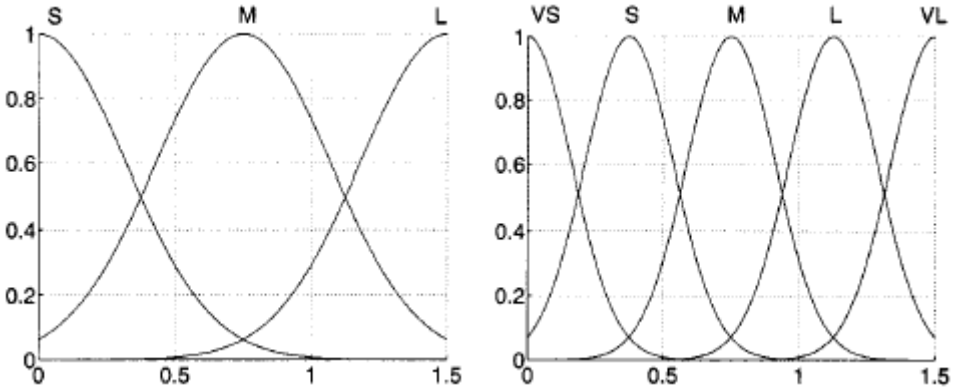
$$\begin{aligned}
&\frac{\partial C(s_i \cap w_{ij})}{\partial x_i^{\sigma}} \\
&= \int_{-\infty}^{h_1} \frac{\partial}{\partial x_i^{\sigma}} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx \\
&\quad + \int_{h_1}^{h_2} \frac{\partial}{\partial x_i^{\sigma}} e^{-((x-x_i^c)/x_i^{\sigma})^2}\, dx \\
&\quad + \int_{h_2}^{\infty} \frac{\partial}{\partial x_i^{\sigma}} e^{-((x-w_{ij}^c)/w_{ij}^{\sigma})^2}\, dx \\
&= \frac{h_1-x_i^c}{x_i^{\sigma}} e^{-((h_1-x_i^c)/x_i^{\sigma})^2} - \frac{h_2-x_i^c}{x_i^{\sigma}} e^{-((h_2-x_i^c)/x_i^{\sigma})^2} \\
&\quad + \sqrt{\pi}\left(-\operatorname{erf}\left(\frac{\sqrt{2}(h_1-x_i^c)}{x_i^{\sigma}}\right) + \operatorname{erf}\left(\frac{\sqrt{2}(h_2-x_i^c)}{x_i^{\sigma}}\right)\right)
\end{aligned} \tag{4.4.27}$$

Εάν $x_i^c > w_{ij}^c$ και εφόσον $h_1 < h_2$, οι παραστάσεις των $\partial C(s_i \cap w_{ij})/\partial w_{ij}^c$, $\partial C(s_i \cap w_{ij})/\partial w_{ij}^{\sigma}$ και $\partial C(s_i \cap w_{ij})/\partial x_i^{\sigma}$ είναι ίδιες με τις (4.4.25)-(4.4.27) αντίστοιχα.

4.5 Εξάγοντας Κανόνες από ένα Εκπαιδευμένο Δίκτυο

Σε αυτό το κεφάλαιο, εστιάζουμε στην ευκολία με την οποία ερμηνεύεται η γνώση που βρίσκεται ενσωματωμένη σε ένα σύστημα SuPFuNIS.

Για να ερμηνεύσουμε το σύνολο των κανόνων, θεωρούμε ασαφείς διαμερίσεις (fuzzy interpretation sets) τέτοιες που να παρέχουν αναλυτικές γλωσσικές αναπαραστάσεις των διαστημάτων που αντιστοιχούν στα υπερσύνολα αναφοράς. Για παράδειγμα, τιμές μιας γλωσσικής μεταβλητής μπορούν να αναπαρασταθούν ως συμμετρικές συναρτήσεις συμμετοχής Gauss με όμοιες διασπορές και ισαπέχοντα κέντρα. Υπόδειγμα ασαφών διαμερίσεων με τρεις και πέντε γλωσσικές τιμές (κλάσεις), στο ίδιο υπερσύνολο αναφοράς, παρουσιάζονται στο παρακάτω σχήμα.



Σχήμα 4.5.1 Ασαφείς διαμερίσεις συνόλων (fuzzy interpretation sets)

VS→VERY SMALL (ΠΟΛΥ ΜΙΚΡΟ)
S→SMALL (ΜΙΚΡΟ)
M→MEDIUM (ΜΕΣΑΙΟ)
L→LARGE (ΜΕΓΑΛΟ)
VL→VERY LARGE (ΠΟΛΥ ΜΕΓΑΛΟ)

Για να διατυπωθεί ένας εκπαιδευμένος κανόνας με γλωσσικές τιμές σύμφωνα με μια επιλεγμένη ασαφή διαμέριση, μετράται η ασαφής τομή ανάμεσα:

- σε κάθε προσυναπτικό μέρος ενός κανόνα και σε κάθε ασαφές σύνολο της ασαφούς διαμέρισης

και ανάμεσα

- σε κάθε μετασυναπτικό μέρος ενός κανόνα και σε κάθε ασαφές σύνολο της ασαφούς διαμέρισης.

Στην συνέχεια το προσυναπτικό ή μετασυναπτικό μέρος ενός κανόνα συσχετίζεται με την ασαφή διαμέριση στην οποία παρουσιάζεται η μέγιστη τιμή ασαφούς τομής. Τελικά το προκύπτον σύνολο ενσωματωμένων κανόνων μπορεί να αναπαρασταθεί σε ένα πίνακα όπου θα έχει ως στήλες τις εισόδους και τις εξόδους (κάθε ασαφούς κανόνα) και ως γραμμές τους ίδιους τους κανόνες. Σε αυτόν τον πίνακα μπορεί να παρατηρηθούν ομοιότητες ανάμεσα σε διάφορους κανόνες, πράγμα που σημαίνει ότι το σύστημα δύναται να αναπαρασταθεί με μικρότερο αριθμό κανόνων.

Σε περίπτωση που παρατηρηθούν κανόνες που για ίδιες εισόδους παράγουν διαφορετικά αποτελέσματα (διαφορετικές γλωσσικές τιμές), τότε απαιτείται αύξηση στον αριθμό των συνόλων της ασαφούς διαμέρισης που χρησιμοποιήθηκε κατά την φάση της αρχικής ερμηνείας. Το ελάχιστο επίπεδο ασαφούς διαμέρισης των υπερσυνόλων αναφοράς στο οποίο δεν θα παρατηρούνται τέτοια φαινόμενα είναι το

κατάλληλο επίπεδο για να ερμηνευθεί η ενσωματωμένη γνώση του SuPFuNIS. Έτσι η ερμηνεία των κανόνων διευκολύνεται με άμεσο τρόπο υιοθετώντας ασαφείς τομές σε συνδυασμό με ασαφείς διαμερίσεις συγκεκριμένου αριθμού γλωσσικών τιμών.

4.6 Εμπλουτίζοντας το Σύστημα SuPFuNIS με Εμπειρική Γνώση

Στο τμήμα αυτό, έχει σημασία να αναδείξουμε ότι το μοντέλο SuPFuNIS είναι κατάλληλο για περιπτώσεις όπου ένα μικρό σύνολο εκπαίδευσης από αριθμητικά δεδομένα εμπλουτίζεται με εμπειρική γνώση. Έτσι μέσα από πειραματικές μετρήσεις [1] επιβεβαιώνεται η ικανότητα του μοντέλου να βελτιώνεται μέσω της παράλληλης χρήσης αριθμητικών και γλωσσικών κανόνων, οι οποίοι προκύπτουν από συμπεράσματα εμπειρογνωμόνων και ειδικών. Μάλιστα αξίζει να σημειωθεί ότι οι αυξημένες επιδόσεις του SuPFuNIS επιβεβαιώνονται ακόμη και στην περίπτωση που το μέγεθος των συνόλων εκπαίδευσης είναι μικρό.

4.7 Προσαρμοζόμενος Νευρο-Ασαφής Ταξινομητής

Η καινοτομία αυτού του μοντέλου ταξινόμησης [1] βρίσκεται σε μια ευέλικτη τεχνική αρχικοποίησης η οποία πρώτα διαμερίζει το χώρο δεδομένων χρησιμοποιώντας κατανομές Gauss και στην συνέχεια συγχωνεύει τις επιμέρους ομάδες (clusters) ώστε τελικά να παραχθεί μια αποτελεσματική διαμέριση.

4.7.1 Εισαγωγή

Τα ασαφή νευρωνικά δίκτυα έχουν την ικανότητα να χειρίζονται και να προσαρμόζονται τόσο σε αριθμητικά όσο και σε γλωσσικά περιβάλλοντα. Επίσης μπορούν να χειριστούν ασαφή χαρακτηριστικά και μπορούν να προσαρμόζονται, διαμέσου εκπαίδευσης, σε ποικίλα προβλήματα. Έτσι σε ένα υπολογιστικό περιβάλλον που λαμβάνει υπόψη το περιεχόμενο, ένα προσαρμοζόμενο νευρο-ασαφές μοντέλο θα έπρεπε να αυτο-αξιολογείται. Παράλληλα θα έπρεπε να ενσωματώνει (στην αρχιτεκτονική του) γνώση προερχόμενη από τα δεδομένα εκπαίδευσης. Συγκεκριμένα, ο συνήθης τρόπος για να ενσωματωθεί γνώση καθοδηγούμενη από τα δεδομένα (ή εμπειρική γνώση) σε ένα νευρο-ασαφές μοντέλο, είναι να χρησιμοποιηθεί μια μέθοδος ομαδοποίησης ή διαμέρισης. Στην μέθοδο ομαδοποίησης, τα κέντρα των ασαφών κανόνων αρχικοποιούνται σαν διανύσματα ομάδων που προκύπτουν από το σύνολο δεδομένων εισόδου. Ένας αλγόριθμος μάθησης χρησιμοποιείται μετά ώστε να βελτιώσει τους κανόνες που βασίζονται στο διαθέσιμο σύνολο εκπαίδευσης. Η μέθοδος διαμέρισης διαχωρίζει το σύνολο εισόδων-εξόδων σε λεπτομερέστερες περιοχές. Κάθε διαμέριση θεωρείται ότι αναπαριστά ένα κανόνα ΕΑΝ-ΤΟΤΕ.

Αμφότερες οι παραπάνω προσεγγίσεις παρουσιάζουν ανεπαρκή προσαρμοστικότητα. Ο αριθμός των κανόνων που περιγράφει το φυσικό, υπό εξέταση, φαινόμενο υπολογίζεται ευριστικά και δεν αλλάζει στη διάρκεια της εκπαίδευσης.

Η παρούσα προσέγγιση περιλαμβάνει έναν έξυπνο ταξινομητή, ο οποίος λαμβάνει υπόψη το περιεχόμενο και ο οποίος παρέχει αυτο-αξιολόγηση επαναπροσδιορίζοντας τους "αδύναμους" κανόνες όποτε αυτό κρίνεται απαραίτητο.

Το προτεινόμενο σύστημα εκπαιδεύεται χρησιμοποιώντας μια καινοτόμα διαδικασία αρχικοποίησης. Αυτή η μέθοδος συνδυάζει ταυτόχρονα μια μέθοδο διαμέρισης και ένα αλγόριθμο ομαδοποίησης ικανά να εξαγάγουν ασαφείς κανόνες από ένα καλά ορισμένο σύνολο δεδομένων με μη-ευριστικό τρόπο.

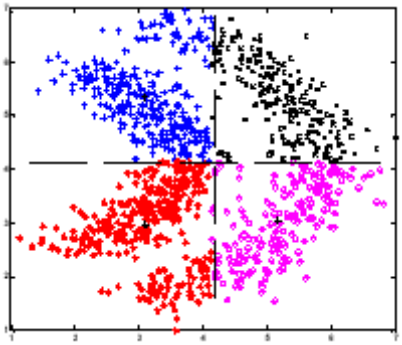
4.7.2 Η Μέθοδος Αρχικοποίησης στο SuPFuNIS

Στο μοντέλο SuPFuNIS, ο αριθμός των ομάδων καθορίζει τον αριθμό των κανόνων και η ομαδοποίηση επιτυγχάνεται στο σύνολο εισόδου-εξόδου. Έτσι τα κέντρα και τα όρια των επιμέρους ομάδων μπορούν να χρησιμοποιηθούν ως τιμές των κέντρων και των διασπορών των προσυναπτικών και μετασυναπτικών ασαφών βαρών. Η χρησιμοποιούμενη τεχνική ομαδοποίησης είναι ο αλγόριθμός Fuzzy C Means, ο οποίος χρησιμοποιείται για να ομαδοποιήσει το δοθέν σύνολο δεδομένων, σε συνδυασμό με τον δείκτη αξιολόγησης Xie-Beni, που χρησιμοποιείται για να επιλέξει την καλύτερη ομάδα.

4.7.3 Εξαγωγή Γνώσης από Αριθμητικά Δεδομένα

Μία από τις μεθόδους για να εξαγάγεις γνώση από το σύνολο εκπαίδευσης είναι να ομαδοποιηθούν τα δεδομένα χρησιμοποιώντας κάποια τεχνική ομαδοποίησης. Η αρχικοποίηση που βασίζεται σε ομάδες θεωρείται ότι βελτιώνει το ρυθμό μάθησης όπως και την επίδοση του συστήματος.

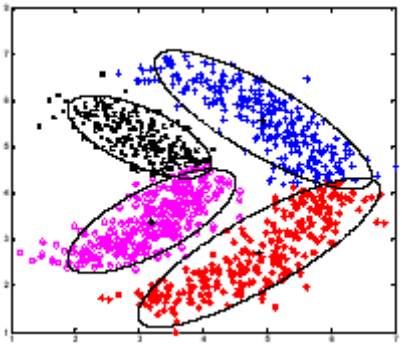
Είναι προφανές όμως, ότι διαφορετικοί αλγόριθμοι ακόμα και πολλαπλές εκτελέσεις του ίδιου αλγορίθμου παράγουν διαφορετικά αποτελέσματα διαμέρισης, λόγω τυχαίας ή παραμετροποιημένης αρχικοποίησης. Για παράδειγμα, το SuPFuNIS υιοθετεί τον αλγόριθμο FCM, για να διαμερίσει ένα σύνολο δεδομένων, μιας και είναι απλός και γρήγορος. Ωστόσο, η επίδοση του εξαρτάται αφενός μεν από τα αρχικά κέντρα των ομάδων και αφετέρου δε από τον προκαθορισμένο αριθμό των ομάδων. Επομένως για να βρεθεί ο βέλτιστος αριθμός ομάδων είναι απαραίτητο να εκτελεστεί ο FCM αρκετές φορές, κάθε φορά χρησιμοποιώντας ένα διαφορετικό αριθμό ομάδων.



Σχήμα 4.7.3.1 Διαμερισμός τεχνητών δεδομένων, χρησιμοποιώντας FCM με Xie-Beni δείκτη για τέσσερις ομάδες (προκαθορισμένες)

Για να ξεπεραστούν αυτοί οι περιορισμοί, αναπτύχθηκε ένας αλγόριθμος πολυομαδοποίησης και συγχώνευσης. Μια παραλλαγή αυτού έχει παρουσιαστεί προηγούμενα [2], έτσι προτείνεται μια τροποποίηση της μεθόδου, η οποία

συμπεριλαμβάνει χρήση Gauss κατανομών όπως και μια νέα προσέγγιση για να προσδιορισθούν κριτήρια συγχώνευσης σε ελλειψοειδείς ομάδες.



Σχήμα 4.7.3.2 Διαμερισμός των ιδίων δεδομένων χρησιμοποιώντας Multi-Clustering Gaussian Fusion, η οποία οδηγεί σε 4 κανόνες

Σύμφωνα με την τεχνική αυτή, τα δεδομένα θεωρείται πως έχουν δημιουργηθεί από διαφορετικά παραμετροποιημένες κατανομές Gauss. Έχοντας αυτό κατά νου, θεωρείται ότι η εύρεση του καταλληλότερου αριθμού από ελλειψοειδείς καμπύλες στο σύνολο δεδομένων μπορεί να βελτιώσει την ισχύ του συστήματος ταξινόμησης. Η προτεινόμενη διαμέριση περιλαμβάνει δύο κύριες φάσεις: την διαμέριση και την συγχώνευση.

4.7.3.1 Η Διαδικασία Διαμέρισης

Στην διαδικασία διαμέρισης, ένας βασικός αλγόριθμος ομαδοποίησης χρησιμοποιείται για ένα ορισμένο αριθμό επαναλήψεων, με σκοπό να επιτευχθεί μια διακριτή διαμέριση N σημείων σε ένα προκαθορισμένο σύνολο C ομάδων. Στην υλοποίηση αυτή, γίνεται χρήση του αλγορίθμου των k-μέσων (k-means) ως βασικού αλγορίθμου ομαδοποίησης. Ο βασικός αλγόριθμος ομαδοποίησης διαμερίζει το σύνολο δεδομένων με διαφορετικό τρόπο σε κάθε επανάληψη, δημιουργώντας έτσι πρόβλημα απόφασης για το ποια ομάδα σε κάποια εκτέλεση αντιστοιχεί σε ποια ομάδα σε κάποια άλλη εκτέλεση.

Ο προτεινόμενος αλγόριθμος ξεπερνά αυτό το πρόβλημα χρησιμοποιώντας την ομοιότητα ανάμεσα σε ομάδες που δημιουργούνται κατά την διάρκεια διαδοχικών εκτελέσεων. Αυτό επιτυγχάνεται καθορίζοντας το ποσοστό των σημείων μιας ομάδας στην t-οστή εκτέλεση που ανήκουν στις ομάδες που δημιουργήθηκαν κατά την (t-1)-οστή εκτέλεση. Με αυτό τον τρόπο κάθε ομάδα της παρούσας εκτέλεσης ανατίθεται σε μια της προηγούμενης εκτέλεσης, οδηγώντας έτσι σε μια διαδικασία επαναρίθμησης των ομάδων.

Μετά την επαναρίθμηση, εάν ένα χαρακτηριστικό $\vec{x}$ έχει ανατεθεί στην ομάδα q, τότε μία θετική ψήφος δίνεται στην ομάδα q και μια αρνητική σε όλες τις υπόλοιπες. Αυτή η διαδικασία καθορίζει μια διαδικασία ψηφοφορίας, κατά την διάρκεια της οποίας ένας πίνακας ψήφων (Voting Table), διαστάσεων NxC, ανανεώνεται, έτσι ώστε το VT(i,j) να δηλώνει το βαθμό συμμετοχής του χαρακτηριστικού $\vec{x}_i$ στην ομάδα j, με i = 1,...,N και j =1,...,C. Χρησιμοποιώντας τον πίνακα VT και τη σχέση ανάμεσα στα πρότυπα μιας ομάδας και στα πρότυπα των υπολοίπων ομάδων, ένας πίνακας NRT, με διαστάσεις CxC, μπορεί να

κατασκευαστεί. Κάθε στοιχείο του NRT(i,j) αντιπροσωπεύει τη σχέση γειτνίασης ανάμεσα στις ομάδες i και j.

Για κάθε ομάδα, μια διαφορετική κατανομή Gauss (ελλειψοειδής) ανατίθεται και ο προσδιορισμός των Gauss παραμέτρων βασίζεται στον αλγόριθμο προσδόκιμης ελαχιστοποίησης (Expectation-Minimization). Έτσι σύμφωνα με τα βήματα εκτέλεσης του ΕΜ, η περαιτέρω ρύθμιση των παραμέτρων συνεχίζεται μέχρι την σύγκλιση.

4.7.3.2 Η Διαδικασία Συγχώνευσης

Αφού οι παράμετροι κάθε Gauss κατανομής καθοριστούν, η διαδικασία συγχώνευσης λαμβάνει χώρα βρίσκοντας τον βέλτιστο αριθμό των ελλειψοειδών στο σύνολο δεδομένων. Από την αποσυνέλιξη του πίνακα αυτοσυσχέτισης K_j της j-οστής ελλειψοειδούς, υπολογίζουμε (για κάθε ελλειψοειδή) την γωνία $\varphi_{kji} \in (-\pi/2, \pi/2)$ ανάμεσα στο ιδιοδιάνυσμα του k μείζονος άξονα και της i-οστής διάστασης.

Δοθείσας επίσης της σχέσης γειτνίασης (πίνακας NRT) μεταξύ ελλειψοειδών, η διαδικασία συγχώνευσης εκκινεί με τον προκαθορισμένο αριθμό ελλειψοειδών C και συγχωνεύει αυτές που πληρούν τις επόμενες προϋποθέσεις:

1. αμφότερα τα ελλειψοειδή πρέπει να είναι γείτονες
2. τα δύο διανύσματα γωνιών (φ_{1i},φ_{2i}), όπου i = 1,...,(d-1), ανάμεσα σε δύο ελλειψοειδή πρέπει να ικανοποιούν ένα από τα παρακάτω κριτήρια για κάθε i (d είναι η διάσταση του προβλήματος):
   Α) εάν τα φ_{1i} και φ_{2i} έχουν ίδιο πρόσημο τότε $|\varphi_{1i}-\varphi_{2i}| < 70^0$
   ή
   Β) εάν τα φ_{1i} και φ_{2i} έχουν διαφορετικό πρόσημο τότε
   $180^0 - [|(\varphi_{1i})|+|(\varphi_{2i})|] < 20^o$ ή $[|(\varphi_{1i})|+|(\varphi_{2i})|] < 20^o$

Το επόμενο βήμα είναι η συγχώνευση αυτών των ελλειψοειδών σε ένα και ο επαναπροσδιορισμός του πίνακα ψήφων, προσθέτοντας τις ψήφους του δεύτερου ελλειψοειδούς στο πρώτο. Η διαδικασία συγχώνευσης τρέχει επαναληπτικά μέχρις ότου ο αλγόριθμος να αποτύχει να βρει ένα νέο ζευγάρι ελλειψοειδών να συγχωνεύσει.

4.8 Τροποποίηση του Αλγορίθμου Εκπαίδευσης

Το νευρο-ασαφές σύστημα SuPFuNIS εκπαιδεύεται βάσει του αλγορίθμου επιβλεπόμενης μάθησης backpropagation κάνοντας χρήση παράλληλα της μεθόδου κλίσης. Τόσο ο συγκεκριμένος αλγόριθμος όσο και η αντίστοιχη μέθοδος έχουν ενδελεχώς δοκιμαστεί και επαληθευτεί σε μεγάλο αριθμό εφαρμογών που αφορούν στους τομείς των νευρωνικών και νευρο-ασαφών δικτύων. Αυτό έχει ως αποτέλεσμα οι επιδόσεις και η αποδοτικότητα τους να θεωρούνται δεδομένες πράγμα που ισχύει σε μεγάλο βαθμό στην περίπτωση του SuPFuNIS.

Ωστόσο το SuPFuNIS διαθέτει μια συγκεκριμένη αρχιτεκτονική, βασικό χαρακτηριστικό της οποίας αποτελεί η αναπαράσταση κάθε εμπλεκόμενου ασαφούς συνόλου από συναρτήσεις συμμετοχής Gauss.

$$gauss(x) = e^{-\left(\frac{x-c}{\sigma}\right)^2}$$

Οι βασικές παράμετροι κάθε μιας συνάρτησης Gauss είναι δυο: το κέντρο c και η διασπορά σ. Κατ' ουσία η λειτουργία του αλγόριθμου επιβλεπόμενης εκπαίδευσης συνίσταται στην τροποποίηση του κέντρου και της διασποράς κάθε κόμβου (του συνολικού νευρο-ασαφούς δικτύου) με στόχο την βέλτιστη προσέγγιση της επιθυμητής λειτουργίας. Αυτό γίνεται, όπως περιγράφηκε εκτενώς σε προηγούμενο κεφάλαιο, βάσει των σχέσεων:

$$w_{ij}^{c}(t+1) = w_{ij}^{c}(t) - \eta \frac{\partial e(t)}{\partial w_{ij}^{c}(t)} + \alpha \Delta w_{ij}^{c}(t-1)$$

$$v_{jk}^{c}(t+1) = v_{jk}^{c}(t) - \eta \frac{\partial e(t)}{\partial v_{jk}^{c}(t)} + \alpha \Delta v_{jk}^{c}(t-1)$$

$$w_{ij}^{\sigma}(t+1) = w_{ij}^{\sigma}(t) - \eta \frac{\partial e(t)}{\partial w_{ij}^{\sigma}(t)} + \alpha \Delta w_{ij}^{\sigma}(t-1)$$

$$v_{jk}^{\sigma}(t+1) = v_{jk}^{\sigma}(t) - \eta \frac{\partial e(t)}{\partial v_{jk}^{\sigma}(t)} + \alpha \Delta v_{jk}^{\sigma}(t-1)$$

$$x_{i}^{\sigma}(t+1) = x_{i}^{\sigma}(t) - \eta \frac{\partial e(t)}{\partial x_{i}^{\sigma}(t)} + \alpha \Delta x_{i}^{\sigma}(t-1) \qquad (4.8.1)$$

Ακολουθώντας την αλυσίδα υπολογισμών των μερικών παραγώγων $\frac{\partial e}{\partial w^{\sigma}{}_{ij}}$, $\frac{\partial e}{\partial v^{\sigma}{}_{ij}}$, $\frac{\partial e}{\partial x^{\sigma}{}_{i}}$ όπως αυτή περιγράφεται από τους τύπους (4.4.4) → (4.4.27) και λαμβάνοντας υπόψη ότι η τιμή του η μπορεί να πάρει οποιαδήποτε τιμή, γίνεται εμφανές ότι οι τιμές των διασπορών μπορούν να λάβουν αρνητικές ή / και μηδενικές τιμές ακολουθώντας αυτές τις σχέσεις ανανέωσης. Ακόμη και στην περίπτωση που το η επιλέγεται για αυτό το λόγο μικρό, δεν υπάρχει καμία εξασφάλιση ότι ο όρος της μάθησης είναι μικρότερος από το άθροισμα όρου ορμής και προηγούμενης τιμής της διασποράς.

Η ύπαρξη αρνητικών τιμών για την διασπορά μπορεί να μην επηρεάζει την μορφή των συναρτήσεων συμμετοχής ωστόσο δημιουργεί μια σειρά αστοχιών (λανθασμένων υπολογισμών) που ενδέχεται να οδηγήσουν σε κατάρρευση της αρχιτεκτονικής με την εμφάνιση άπειρων τιμών στα διανύσματα των εισόδων, των συναπτικών βαρών ή των εξόδων. Αξίζει να σημειωθεί ότι ορισμένοι από τους λανθασμένους αυτούς υπολογισμούς οφείλονται στην σιωπηρή παραδοχή ότι η διασπορά κάθε συνάρτησης Gauss είναι ένας αυστηρά θετικός αριθμός, γεγονός όμως που δεν διασφαλίζεται από τον αλγόριθμό μάθησης. Στην συνέχεια παρουσιάζονται ενδεικτικά τρία σημεία στα οποία αρνητική τιμή της διασποράς οδηγεί σε λανθασμένους υπολογισμούς.

- Είναι γνωστό ότι $\int_{-\infty}^{+\infty} e^{-z^2} dz = \sqrt{\pi}$, συνεπώς $\int_{-\infty}^{+\infty} e^{-\frac{(x-c)^2}{\sigma^2}} dx$ = [x = t+c] = $\int_{-\infty}^{+\infty} e^{-\frac{t^2}{\sigma^2}} dt$ =

  εάν σ>0 ⇒ [t = σx] = $\sigma \int_{-\infty}^{+\infty} e^{-x^2} dx$

  εάν σ<0 ⇒ [t = σx] = $\sigma \int_{+\infty}^{-\infty} e^{-x^2} dx = -\sigma \int_{-\infty}^{+\infty} e^{-x^2} dx$

δηλαδή εν γένει $\int_{-\infty}^{+\infty} e^{-\frac{(x-c)^1}{\sigma^2}} dx = |\sigma| \sqrt{x}$ γεγονός που δεν λαμβάνεται υπόψη σε υπολογισμούς όπως ο (4.3.7).

- Κατά τον υπολογισμό του μέτρου ομοιότητας [§ 4.3] η τομή δυο ασαφών συνόλων υπολογίζεται βάσει του κέντρου και της διασποράς τους [σχήμα 4.3.3]. Οι περιπτώσεις που λαμβάνονται υπόψη ισχύουν εφόσον οι διασπορές έχουν θετικές τιμές ειδαλλιώς η υποπερίπτωση $x_i^c \neq w_{ij}^c$, $x_i^\sigma < w_{ij}^\sigma$ θα έπρεπε να μετασχηματιστεί στην υποπερίπτωση $x_i^c \neq w_{ij}^c$, $|x_i^\sigma| < |w_{ij}^\sigma|$ εάν ένας επιθυμούσε να συμπεριλάβει το ενδεχόμενο ύπαρξης αρνητικών τιμών διασποράς.
- Στην φάση της αποασαφοποίησης τα V_{jk} (τα μετασυναπτικά σύνολα χώρων) αναπαριστούνται από απλά εμβαδά και έχουν τιμή $V_{jk} = v_{jk}^\sigma \sqrt{\pi}$. Είναι εύλογο ότι αρνητικό εμβαδό δεν μπορεί να υπάρχει παρόλα αυτά όταν η διασπορά είναι αρνητικός αριθμός τότε το εμβαδόν γίνεται αρνητικό.

Εκτός της περίπτωσης που η διασπορά λαμβάνει αρνητικές τιμές, όπως εξηγήθηκε, η διασπορά μπορεί να μηδενιστεί σε κάποιο βήμα του αλγορίθμου εκπαίδευσης. Σε αυτή την περίπτωση είναι εμφανές ότι το SuPFuNIS καταρρέει υπό την έννοια ότι εμφανίζονται σχέσεις της μορφής $\frac{x}{0}$ και τις οποίες η υπάρχουσα αρχιτεκτονική δεν είναι σε θέση να χειριστεί.

Έτσι, χρειάζεται μια τροποποίηση του αλγορίθμου μάθησης στο επίπεδο των εξισώσεων ανανέωσης (4.8.1). Μία λύση είναι να ελέγχεται η τιμή της διασποράς και όταν αυτή αποκτήσει αρνητική τιμή τότε να λαμβάνεται η απόλυτη τιμή της. Κάτι τέτοιο όμως δεν επιλύει το φαινόμενο εμφάνισης μηδενικών τιμών, ενώ από την άλλη πλευρά αποτελεί βίαιη επέμβαση στον αλγόριθμο μάθησης. Η ύπαρξη μιας αρνητικής ή μηδενικής τιμής ως επόμενο βήμα μιας θετικής τιμής σημαίνει ότι η εν λόγω διασπορά (βάσει του αλγορίθμου) χρειάζεται να μειωθεί, λαμβάνοντας όμως την απόλυτη τιμή της αυτή αυξάνεται, με συνέπεια την μείωση της απόδοσης του αλγορίθμου.

Ωστόσο υπάρχει μια πιο εκλεπτυσμένη αντιμετώπιση η οποία αντιμετωπίζει και τα δυο προηγούμενα προβλήματα. Αρχικά ελέγχεται η τιμή της διασποράς και εφόσον δεν είναι αυστηρά θετική τότε η τιμή της τίθεται σε μια πολύ μικρή θετική τιμή κατωφλίου.

εάν $\sigma \leq 0$ τότε $\sigma = 0.001$ (4.8.2)

Κάτι τέτοιο προφανώς επιλύει το πρόβλημα μηδενισμού της διασποράς και παράλληλα συμβαδίζει με την επιθυμητή μείωση της τιμής της διασποράς όπως αυτή καθορίζεται από τον αλγόριθμο επιβλεπόμενης μάθησης.

Κεφάλαιο 5

Ασύρματα Τοπικά Δίκτυα

5 Ασύρματα Τοπικά Δίκτυα

5.1 Εισαγωγικό Σημείωμα

Υπάρχουν τρεις κύριες περιοχές εφαρμογής των ασυρμάτων κινητών επικοινωνιών, οι οποίες σημειώνουν μεγάλη ανάπτυξη και συνεπώς παρουσιάζουν μεγάλο σχεδιαστικό και ερευνητικό ενδιαφέρον.

- Η ασύρματη πρόσβαση σε μεγάλα δίκτυα για προσωπικές επικοινωνίες χαμηλών απαιτήσεων.
- Τα ασύρματα δίκτυα κινητών επικοινωνιών.
- Τα ασύρματα τοπικά δίκτυα.

Τα *ασύρματα τοπικά δίκτυα* (*Wireless Local Area Networks*) έχουν ως στόχο την παροχή υψηλών ρυθμών μετάδοσης (αρκετά Mbps) σε φορητά τερματικά που μετακινούνται σε περιορισμένες περιοχές, όπως μεγάλα κτίρια, πανεπιστημιουπόλεις, εμπορικά κέντρα ή νοσοκομειακοί χώροι. Σε αυτές τις περιπτώσεις η ανάγκη της μεταφερσιμότητας είναι εντονότερη από την ανάγκη για κινητικότητα. Τα ασύρματα αυτά τοπικά δίκτυα είναι ως επί το πλείστον μεταγωγής πακέτου και μερικές φορές αναφέρονται ως Personal Communication Services δεδομένων. Οι ρυθμοί μετάδοσης στους ασύρματους διαύλους κυμαίνονται από μερικές δεκάδες kbps μέχρι μερικές δεκάδες Mbps. Τα περισσότερα συστήματα λειτουργούν στις περιοχές συχνοτήτων που προορίζονται για βιομηχανικές, επιστημονικές και ιατρικές εφαρμογές (Industrial, Scientific, Medical), κοντά στα 900Mhz και 2.4Ghz. Η χρησιμοποίηση περιοχής ISM δεν απαιτεί άδεια λειτουργίας και επιτρέπει ισχύ εκπομπής μέχρι 1W, εάν τα εκπεμπόμενα σήματα εκπληρούν προκαθορισμένα κριτήρια ως προς το φάσμα. Λόγω των υψηλών ρυθμών μετάδοσης που χρησιμοποιούνται στην ζεύξη και λόγω των περιορισμών ισχύος που υπάρχουν στα φορητά τερματικά, η περιοχή κάλυψης των πομποδεκτών WLAN είναι πολύ περιορισμένη, της τάξεως μερικών μέτρων μέσα σε κτίρια.

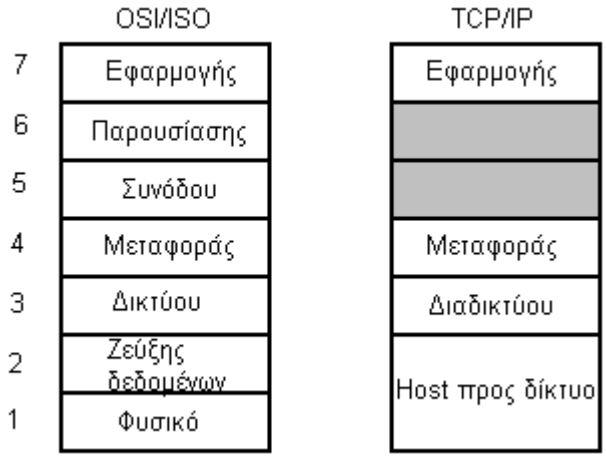
Επειδή η ανάπτυξη των WLAN είναι σχετικά νέα οι ανάγκες και οι απαιτήσεις των χρηστών δεν είναι τόσο σαφείς. Προς το παρόν αυτό έχει ως αποτέλεσμα να μην υπάρχει συγκεκριμένη κατεύθυνση ως προς τις εφαρμογές.

Μια φιλοσοφία της ασύρματης δικτύωσης για μετάδοση δεδομένων βασίζεται σε σταθερούς σταθμούς βάσης, συνδεδεμένους μέσω σταθερού δικτύου και ελεγχόμενους από κεντρικό σημείο. Αυτός ο τύπος WLAN έχει αρχιτεκτονική και δομή ελέγχου εφάμιλλες με εκείνες των κυψελωτών συστημάτων. Φορητά τερματικά μπορούν να έχουν πρόσβαση σε τέτοιο δίκτυο δεδομένων μέσα σε σχετικά καθορισμένη περιοχή που καλύπτεται από τους σταθμούς βάσης (π.χ. μεγάλο κτίριο, περιοχή μιας βιομηχανίας, νοσοκομείο, πανεπιστημιούπολη).

Μια δεύτερη φιλοσοφία δικτύωσης βασίζεται σε φορητά τερματικά με μόνο έναν ή λίγους σταθερούς σταθμούς βάσης που χρησιμεύουν ως πύλες προς εξωτερικά δίκτυα δεδομένων. Τα δίκτυα αυτά δεν έχουν κεντρικό έλεγχο και εφαρμόζουν πρωτόκολλα μεταξύ όλων των οντοτήτων, φορητών και σταθερών, ώστε να σχηματίζεται ένα αυτοοργανούμενο δίκτυο κατανεμημένου ελέγχου. Ένα τέτοιο WLAN δεν υπάρχει μέχρι την στιγμή όπου μια οντότητα εκκινεί της διαδικασία οργάνωσης του. Οι σταθεροί σταθμοί βάσης χρησιμεύουν μόνο ως πύλες προς σταθερά δίκτυα και δεν έχουν ρόλο ειδικού ελεγκτή στο κατανεμημένο δίκτυο.

Μια αναφορά σε οποιασδήποτε μορφής δίκτυα υπολογιστών είναι ελλιπής από την στιγμή που δεν περιλαμβάνει την πλέον διαδεδομένη αρχιτεκτονική δικτύων,

το μοντέλο αναφοράς TCP/IP. Η αρχιτεκτονική που δίδει την δυνατότητα να συνδέονται μαζί πολλαπλά δίκτυα με διαφανή τρόπο είναι *το μοντέλο αναφοράς TCP/IP* (*TCP/IP reference*).



Σχήμα 5.1.1: Το μοντέλο αναφοράς TCP/IP.

Το Στρώμα Host προς Δίκτυο: Το μοντέλο TCP/IP δεν αναφέρει πολλά ως προς το τι ακριβώς συμβαίνει εκεί, εκτός από την υπόδειξη, ότι ο host πρέπει να συνδεθεί με το δίκτυο χρησιμοποιώντας κάποιο πρωτόκολλο, ώστε να μπορεί να στέλνει πακέτα IP σε αυτό. Το πρωτόκολλο αυτό δεν είναι καθορισμένο και ποικίλει από host σε host και από δίκτυο σε δίκτυο.

Το Στρώμα Διαδικτύου: Όλες οι απαιτήσεις οδήγησαν στην επιλογή ενός δικτύου μεταγωγής πακέτου, βασισμένου σε ένα στρώμα διαδικτύου που παρέχει υπηρεσίες χωρίς σύνδεση. Αυτό το στρώμα που αποκαλείται στρώμα διαδικτύου είναι ο ακρογωνιαίος λίθος που συγκρατεί όλη την αρχιτεκτονική. Η δουλεία του είναι να επιτρέπει στους host να εισάγουν πακέτα σε οποιοδήποτε δίκτυο και να δρομολογεί τα πακέτα ανεξάρτητα στον προορισμό τους. Μπορεί να φτάσουν σε διαφορετική σειρά από αυτήν που στάλθηκαν, οπότε είναι δουλειά των ανωτέρων στρωμάτων να τα επαναδιατάξουν, εφόσον είναι επιθυμητή η παραλαβή με την ορθή σειρά.

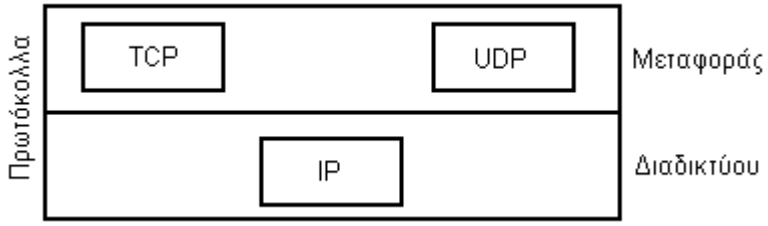
Το στρώμα διαδικτύου καθορίζει μια επίσημη μορφή πακέτου και πρωτοκόλλου που ονομάζεται *πρωτόκολλο διαδικτύου IP* (*Internet Protocol*). Η δουλειά του στρώματος διαδικτύου είναι να παραδίδει τα πακέτα IP στον προορισμό τους. Το κύριο θέμα εδώ είναι η δρομολόγηση πακέτων καθώς επίσης και η αποφυγή συμφόρησης.

Το Στρώμα Μεταφοράς: Το στρώμα αυτό βρίσκεται πάνω από το στρώμα διαδικτύου στην στοίβα πρωτοκόλλων TCP/IP και αποκαλείται συνήθως στρώμα μεταφοράς. Έχει σχεδιαστεί ώστε να παρέχει την δυνατότητα σε ομότιμες οντότητες, που βρίσκονται στους host πηγής και προορισμού, να διεξάγουν μια συζήτηση. Δύο πρωτόκολλα από άκρο σε άκρο έχουν καθοριστεί σε αυτό το σημείο.

Το πρώτο, το *πρωτόκολλο ελέγχου μετάδοσης TCP* (*Transmission Control Protocol*), είναι ένα αξιόπιστο, με σύνδεση, πρωτόκολλο που επιτρέπει σε μια ακολουθία byte που ξεκινά από μια μηχανή να παραδίδεται χωρίς λάθη σε οποιαδήποτε άλλη μηχανή στο διαδίκτυο. Το πρωτόκολλο τεμαχίζει την εισερχόμενη ακολουθία byte σε διακριτά μηνύματα και περνά το καθένα τους στο στρώμα διαδικτύου. Στον προορισμό, η διεργασία λήψης TCP συναρμολογεί την ακολουθία εξόδου από τα λαμβανόμενα μηνύματα. Το TCP χειρίζεται επίσης τον έλεγχο ροής,

ώστε να εμποδίσει έναν γρήγορο πομπό από το να πλημμυρίσει έναν αργό δέκτη με περισσότερα μηνύματα από όσα μπορεί να δεχτεί.

Το δεύτερο πρωτόκολλο στο στρώμα αυτό, το *πρωτόκολλο δεδομενογραφημάτων χρήστη* (*User Datagram Protocol*), είναι ένα μη αξιόπιστο, χωρίς σύνδεση, πρωτόκολλο για εφαρμογές που δεν επιθυμούν τον έλεγχο της ακολουθίας ή της ροής του TCP και επιθυμούν να χρησιμοποιούν το δικό τους.



Σχήμα 5.1.2: Τα βασικά πρωτόκολλα του μοντέλου TCP/IP.

Το Στρώμα Εφαρμογής: Στην κορυφή της στοίβας πρωτοκόλλων TCP/IP βρίσκεται το στρώμα εφαρμογής. Αυτό περιλαμβάνει όλα τα πρωτόκολλα ανωτέρων στρωμάτων που χρησιμοποιούνται στις τεχνολογίες δικτύωσης-διαδικτύωσης.

5.2 Πρότυπο IEEE 802.11

Η υπολογιστική επεξεργασία μέσω ασύρματης πρόσβασης (wireless computing) είναι μια αναπτυσσόμενη τεχνολογία, η οποία επιτρέπει στους χρήστες να συνδέονται σε δίκτυο υπολογιστών χωρίς να δεσμεύονται από την ύπαρξη ενσύρματου δικτύου. Τα ασύρματα τοπικά δίκτυα όπως και τα ενσύρματα ομόλογα τους, έχουν σχεδιαστεί για να παρέχουν υψηλό εύρος ζώνης σε χρήστες που περιφέρονται σε περιορισμένη γεωγραφική περιοχή. Παρά το γεγονός ότι τα WLAN στοχεύουν στο να παρέχουν στους χρήστες την ελευθερία να περιφέρονται οπουδήποτε, χωρίς να χάνουν επαφή με το δίκτυο, οι προσπάθειες για την ανάπτυξη τους κατευθύνονται κυρίως στην χρησιμοποίηση ασυρμάτων στοιχείων ως διεπαφές των WLAN προς τα ενσύρματα δίκτυα κορμού σε δύσκολα περιβάλλοντα δικτύωσης, αντικαθιστώντας τα τελευταία 30-50 m της καλωδίωσης προς το τερματικό σύστημα. Τα WLAN προσφέρουν ένα μέσο για νέες εφαρμογές που εισάγονται, μεταξύ άλλων, με την συνεχώς αυξανόμενη αγορά φορητών υπολογιστών. Για παράδειγμα στα νοσοκομεία οι ιατρικές διαγνώσεις θα μπορούσαν να διευκολυνθούν με την χρησιμοποίηση προσωπικών ψηφιακών βοηθών (Personal Digital Assistants).

Ένα τελείως διαφορετικό πεδίο εφαρμογών των WLAN είναι η εγκατάσταση ad hoc δικτύων (π.χ. σε μία συνεδρίαση ή σε άλλες θέσεις) για τοπική χρήση. Οι σταθμοί που εμπλέκονται μπορούν να αποτελέσουν ένα αυτοδιαχειριζόμενο και δυναμικό δίκτυο, δηλαδή οι φορητοί υπολογιστές μπορεί ελεύθερα να μπαίνουν ή να βγαίνουν από μια σύνοδο. Παρά την αυξημένη πολυπλοκότητα των πρωτοκόλλων, το ad hoc δίκτυο αποτελεί ένα εύκολα εγκαθιστάμενο, και γι' αυτό το λόγο φθηνό, μέσο για τοπική δικτύωση. Με πολύ απλά λόγια μπορεί να λεχθεί, ότι τα συστήματα τοποθετούνται μέσα σε μια περιοχή με λογικά όρια και συγχρονίζονται. Στην συνέχεια, αφού συμφωνηθεί μια διάταξη, φτιάχνεται το δίκτυο. Σύμφωνα με τις κατηγορίες κινητικότητας, αυτό είναι ένα βήμα μπροστά σε σχέση με τις παραδοσιακές εφαρμογές των WLAN όπως αυτές περιγράφηκαν προηγούμενα. Μια

ομάδα από κινητούς χρήστες μπορεί να αποτελέσει ένα αυτόνομο τοπικό δίκτυο, οπουδήποτε, οποιαδήποτε στιγμή, με αμελητέα προσπάθεια.

Στην ιδανική περίπτωση, οι χρήστες των WLAN θα επιθυμούσαν να έχουν τις ίδιες δυνατότητες και την ίδια εξυπηρέτηση, όπως και τα ενσύρματα δίκτυα. Ωστόσο για την εκπλήρωση αυτών των στόχων υπάρχουν περιορισμοί που δεν εμφανίζονται στα ασύρματα τοπικά δίκτυα.

Παρεμβολές και Αξιοπιστία: Οι παρεμβολές στις ασύρματες επικοινωνίες μπορεί να προέλθουν από ταυτόχρονες εκπομπές δυο ή περισσότερων πηγών που μοιράζονται την ίδια ζώνη συχνοτήτων. Όταν υπάρχουν πολλοί σταθμοί που περιμένουν να απελευθερωθεί ο δίαυλος και να αρχίσουν να εκπέμπουν, εμφανίζονται συγκρούσεις. Συγκρούσεις επίσης εμφανίζονται λόγω του προβλήματος του κρυμμένου τερματικού, όπου κάποιος σταθμός πιστεύοντας ότι ο δίαυλος είναι ελεύθερος, αρχίζει την εκπομπή χωρίς να ανιχνεύσει επιτυχώς την ήδη ευρισκόμενη σε εξέλιξη μετάδοση. Παρεμβολές εμφανίζονται και από τις διαλείψεις πολλαπλών διαδρομών, που χαρακτηρίζονται από τυχαίες διακυμάνσεις του πλάτους και της φάσης στην λήψη.

Ασφάλεια Επικοινωνίας: Στα ενσύρματα δίκτυα, το μέσο μετάδοσης μπορεί να παρέχει φυσική ασφάλεια και η πρόσβαση ελέγχεται εύκολα. Στα ασύρματα δίκτυα, η ασφάλεια επικοινωνίας είναι δυσκολότερο να ελεγχθεί, καθότι το μέσο μετάδοσης είναι ανοικτό σε οποιονδήποτε βρίσκεται στην ραδιοκάλυψη ενός πομπού. Η ασφάλεια των δεδομένων στο ασύρματο τμήμα επιτυγχάνεται με απόκρυψη. Ωστόσο η διαδικασία της απόκρυψης αυξάνει το κόστος του συστήματος και μειώνει την επίδοσή του.

Κατανάλωση Ισχύος: Συνήθως οι διάφορες συσκευές που συνδέονται σε ενσύρματο δίκτυο τροφοδοτούνται από το δίκτυο παροχής ηλεκτρικής ισχύος του κτιρίου. Οι ασύρματες συσκευές όμως έχει νόημα να είναι φορητές και / ή κινητές και να τροφοδοτούνται με μπαταρίες. Συνεπώς, πρέπει να σχεδιάζονται ώστε να είναι πολύ αποτελεσματικές, έτσι για παράδειγμα πρέπει να έχουν κατάσταση ηρεμίας με χαμηλή κατανάλωση ισχύος και να έχουν οθόνες χαμηλής ισχύος.

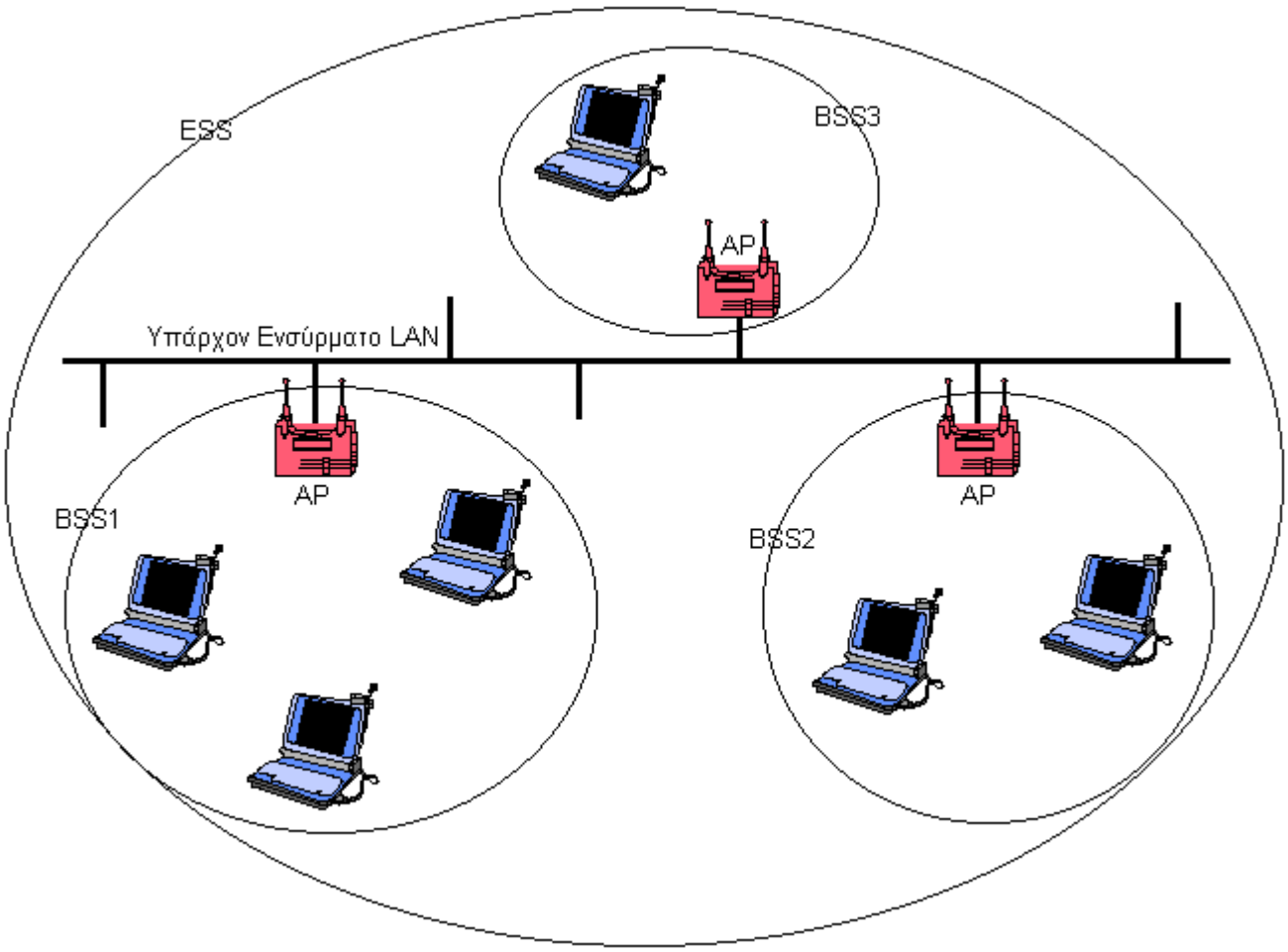
Ασφάλεια Χρηστών: Τα συστήματα πρέπει να σχεδιάζονται κατά τρόπο που να ελαχιστοποιείται η εκπεμπόμενη ισχύς ώστε να μην επιβαρύνεται η υγεία των χρηστών.

Κινητικότητα: Σε αντίθεση με τα ενσύρματα τερματικά, ένα από τα πρωτεύοντα πλεονεκτήματα των ασύρματων τερματικών είναι η ελευθερία κινήσεων. Συνεπώς η σχεδίαση πρέπει να υποστηρίζει τη διαπομπή και τη δρομολόγηση της κίνησης σε κινούμενους χρήστες.

Διέλευση: Η χωρητικότητα των ασυρμάτων τοπικών δικτύων πρέπει στην ιδανική περίπτωση να προσεγγίζει εκείνη των ενσύρματων. Ωστόσο λόγω φυσικών περιορισμών και λόγω περιορισμένου εύρους ζώνης, τα WLAN προς το παρόν ικανοποιούν προδιαγραφές για ρυθμούς μετάδοσης μεταξύ 1-20 Mbps.

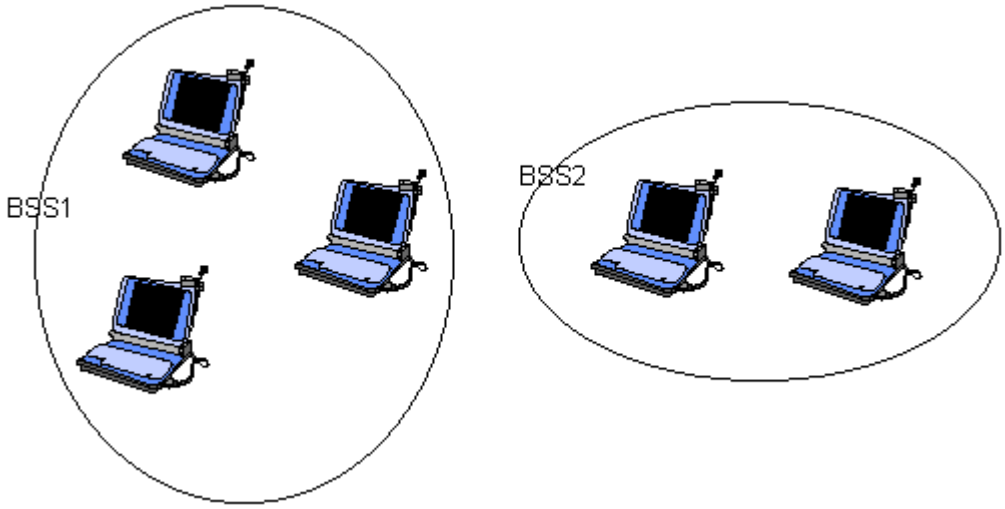
Το πρότυπο 802.11 της IEEE προδιαγράφει ρυθμούς μετάδοσης μέχρι 2 Mbps, χρησιμοποιώντας τεχνολογία απλωμένου φάσματος στην περιοχή συχνοτήτων 2.4 GHz. Για περισσότερες πληροφορίες μπορεί κανείς να ανατρέξει στις αναφορές [1], [2], [3] και [4]. Το πρότυπο διακρίνει δυο τοπολογίες δικτύου: την τοπολογία δικτύου υποδομής και την ad hoc τοπολογία.

- **<u>Δίκτυο Υποδομής:</u>** Στην τοπολογία με δίκτυο υποδομής τα κινητά επικοινωνούν με το δίκτυο κορμού μέσω ενός *σημείου πρόσβασης (Access Point*). Το AP είναι μια γέφυρα που συνδέει το δίκτυο 802.11 με την υποδομή του ενσύρματου δικτύου κορμού. Στην διάταξη αυτή, ένα σύστημα διανομής συνδέει πολλές *ομάδες βασικής εξυπηρέτησης* (*Basic Service Sets*) μέσω σημείων πρόσβασης, ώστε να σχηματιστεί μια ενιαία υποδομή που ονομάζεται *εκτεταμένη ομάδα εξυπηρέτησης* (*Extended Service Set*). Ένα κινητό τερματικό μπορεί να περιφέρεται σε διαφορετικά BSS ενός ESS χωρίς να χάνει την σύνδεση του με το δίκτυο κορμού (διαπομπή).



Σχήμα 5.2.1: Τοπολογία IEEE 802.11 με δίκτυο υποδομής.

- **<u>Ad Hoc:</u>** Στην διάταξη ad hoc τα κινητά τερματικά επικοινωνούν μεταξύ τους σε ανεξάρτητο BSS χωρίς σύνδεση προς το ενσύρματο δίκτυο κορμού. Στην περίπτωση αυτή, μερικές από τις λειτουργίες του AP, που χρειάζονται για τον σχηματισμό και την διατήρηση ενός BSS, παρέχονται από ένα κινητό τερματικό. Επιπρόσθετα με την μέθοδο πολλαπλής πρόσβασης CSMA/CA, που είναι κατάλληλη για ασύγχρονες μεταδόσεις, το πρότυπο IEEE 802.11 παρέχει και έναν μηχανισμό με προτεραιότητες, χωρίς ανταγωνισμό, ελεγχόμενο από ένα σημείο για την υποστήριξη σύγχρονων μεταδόσεων.



Σχήμα 5.2.2: Τοπολογία IEEE 802.11 ad hoc.

Το πρότυπο 802.11 προβλέπει την χρησιμοποίηση ραδιοδιαύλου 2.4 Ghz με απλωμένο φάσμα και αναπηδήσεις συχνότητας (spread spectrum – Frequency Hoping). Στα συστήματα FH, η συχνότητα με την οποία μεταδίδονται τα δεδομένα λαμβάνει διάφορες τιμές που επιλέγονται από μια ομάδα συχνοτήτων (π.χ. 79 συχνότητες για την Ευρωπαϊκή έκδοση του προτύπου 802.11). Έτσι ο πομπός στέλνει δεδομένα σε δοθείσα συχνότητα για ένα σταθερό χρονικό διάστημα και στην συνέχεια μετάγει στην επόμενη συχνότητα για ένα άλλο σταθερό χρονικό διάστημα. Το σχέδιο μεταπηδήσεων είναι γνωστό στον δέκτη, οπότε ο συνθέτης συχνοτήτων στον δέκτη μπορεί να κάνει αναπηδήσεις συγχρονισμού και να ανακτά το αρχικό σήμα δεδομένων.

Τα συστήματα FH που ορίζονται στο φυσικό στρώμα του 802.11, είναι αργά συστήματα όσο αφορά στις αναπηδήσεις συχνότητας, καθότι μεταδίδουν πολλαπλά διαδοχικά σύμβολα στην ίδια συχνότητα. Στα συστήματα FH, γειτονικές ή επικαλυπτόμενες κυψέλες χρησιμοποιούν διαφορετικές ακολουθίες αναπηδήσεων. Για σχέδια αναπηδήσεων με πολλές συχνότητες (π.χ. Ευρωπαϊκό πρότυπο) είναι απίθανο να χρησιμοποιείται η ίδια συχνότητα ταυτόχρονα σε δυο διαδοχικές κυψέλες. Το πρότυπο καθορίζει τρεις διαφορετικές ακολουθίες αναπηδήσεων, κάθε μια από τις οποίες αποτελείται από 26 σχέδια. Τα σχέδια μέσα σε κάθε μια ομάδα έχουν επιλεγεί με τρόπο που να εμφανίζουν καλές ιδιότητες. Για παράδειγμα, οι διαδοχικές συχνότητες σε μια δοθείσα ακολουθία απέχουν στο φάσμα 6 Mhz τουλάχιστον, ώστε να αποφεύγεται παρεμβολή στενής ζώνης.

Συνοψίζοντας παρατηρείται ότι ένα σύστημα FH μπορεί να παρέχει μεγάλο αριθμό διαύλων (δηλαδή σχεδίων αναπήδησης συχνότητας) και συνεπώς μπορεί να χρησιμοποιηθεί σε πυκνά περιβάλλοντα, όπου οι κυψέλες έχουν επικάλυψη με πολλές κυψέλες. Τα συστήματα FH εξαιτίας της κατασκευής τους εμφανίζουν ανθεκτικότητα στην παρεμβολή στενής ζώνης αφού υφίστανται την οποιαδήποτε παρεμβολή μόνο για ένα κλάσμα του χρόνου. Ο τύπος ραδιοσυστήματος FH μεταδίδει σε στάθμη ισχύος 100 mW ή μικρότερη, γεγονός που του επιτρέπει να καλύπτει αποστάσεις μέχρι το πολύ 100 m σε εσωτερικούς χώρους, ανάλογα με τον ρυθμό μετάδοσης δεδομένων και την κατασκευή και διαρρύθμιση των κτιρίων.

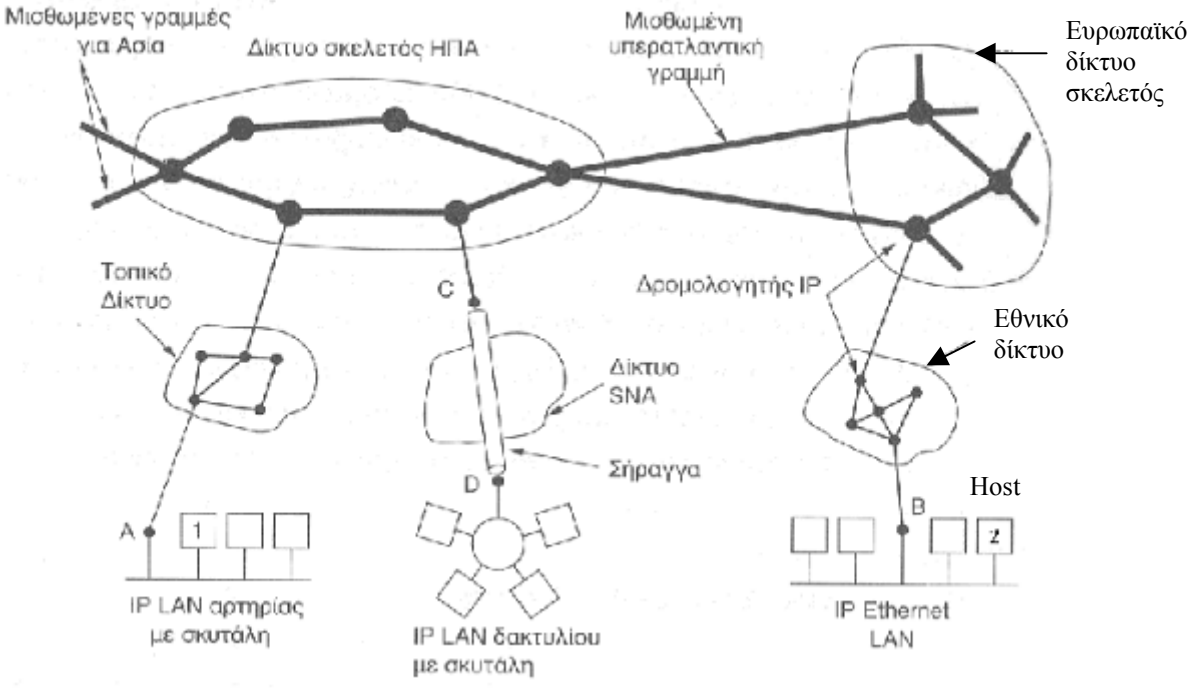
5.3 Το Στρώμα Δικτύου στο Internet

5.3.1 Περιγραφή του Στρώματος Δικτύου

Το στρώμα δικτύου ασχολείται με τη μεταφορά πακέτων από την πηγή στον προορισμό. Η διαδρομή μέχρι τον προορισμό μπορεί να απαιτεί πολλά βήματα σε ενδιάμεσους δρομολογητές που βρίσκονται κατά μήκος της διαδρομής. Το στρώμα δικτύου, συνεπώς, είναι το χαμηλότερο στρώμα που ασχολείται με την από άκρο σε άκρο μετάδοση. Για περισσότερες πληροφορίες [1], [2], [3], [4].

Για να επιτύχει τους σκοπούς του, το στρώμα δικτύου οφείλει να γνωρίζει την τοπολογία του υποδικτύου (δηλαδή το σύνολο των δρομολογητών) και να επιλέγει κατάλληλες διαδρομές μέσα από αυτό. Πρέπει επίσης να φροντίζει να επιλέγει διαδρομές με τρόπο ώστε να αποφεύγει την υπερφόρτωση κάποιων γραμμών και δρομολογητών, ενώ αφήνει άλλους αδρανείς. Τελικά, όταν η πηγή και ο προορισμός βρίσκονται σε διαφορετικά δίκτυα, επαφίεται στο στρώμα δικτύου να χειρισθεί τις διαφορές αυτές και να επιλύσει τα προβλήματα που προκαλούνται από αυτές.

Το στρώμα δικτύου στο internet μπορεί να θεωρηθεί ως συλλογή από διαδίκτυα ή αυτόνομα συστήματα (autonomous systems) που συνδέονται μεταξύ τους. Δεν υπάρχει πραγματική δομή, αλλά υπάρχουν διάφορα δίκτυα σκελετοί. Τα τελευταία είναι κατασκευασμένα από γραμμές υψηλού εύρους ζώνης και ταχείς δρομολογητές. Στα δίκτυα σκελετούς συνδέονται τοπικά τα δίκτυα (Local Area Networks) πολλών πανεπιστημίων, επιχειρήσεων και παροχέων υπηρεσιών internet. Η εσωτερική λειτουργία του internet συνίσταται από ανεξάρτητα πακέτα οργάνωσης χωρίς σύνδεση τα οποία αποκαλούνται δεδομενογραφήματα (datagrams).



Σχήμα 5.3.1.1: Η ιεραρχική οργάνωση του internet.

Ο συνδετικός ιστός του internet είναι το IP. Ένας τρόπος για να σκεφθεί κανείς το στρώμα δικτύου είναι ο ακόλουθος. Το καθήκον του είναι να μάχεται για

την καλύτερη δυνατή μεταφορά datagrams από την πηγή προς τον προορισμό, ασχέτως του αν αυτές οι μηχανές βρίσκονται στο ίδιο δίκτυο ή όχι, ή του εάν υπάρχουν ή όχι άλλα δίκτυα ανάμεσα τους.

Η επικοινωνία στο internet δουλεύει ως εξής. Το στρώμα μεταφοράς παίρνει τους συρμούς δεδομένων και τους τεμαχίζει σε datagrams. Θεωρητικά τα δεδομενογραφήματα μπορεί να φθάσουν τα 64 Kbyte το καθένα, αλλά στην πράξη είναι συνήθως περίπου 1500 byte. Το κάθε datagram μεταδίδεται μέσω του internet, πιθανώς τεμαχιζόμενο σε μικρότερες μονάδες καθώς προχωρεί. Όταν φθάσουν στον προορισμό όλα τα κομμάτια, συναρμολογούνται από το στρώμα δικτύου στο αρχικό δεδομενογράφημα. Μετά το δεδομενογράφημα παραδίδεται στο στρώμα μεταφοράς, που το δίνει στη λαμβάνουσα διαδικασία.

Για το πρωτόκολλο IP λέμε ότι είναι μη αξιόπιστο γιατί δεν εγγυάται την μεταφορά της πληροφορίας που αναλαμβάνει. Κάποιο datagram μπορεί να φθάσει σε λάθος προορισμό, να διπλασιαστεί μέσα στο δίκτυο ή να χαθεί στον δρόμο προς τον προορισμό του. Το πρωτόκολλο IP προσφέρει ένα σύνολο από βασικές λειτουργίες οι οποίες είναι απαραίτητες για την επιτυχή διασύνδεση δικτύων υπολογιστών. Τέτοιες λειτουργίες είναι:

- Λειτουργίες κατακερματισμού των μηνυμάτων και επανασύνδεσης αυτών προκειμένου να περάσουν από υποδίκτυα που υποστηρίζουν διαφορετικού μεγέθους πεδίο δεδομένων στο πλαίσιο τους.
- Λειτουργίες δρομολόγησης των IP datagrams διαμέσου των κόμβων του δικτύου, προκειμένου να φθάσουν στον προορισμό τους.
- Λειτουργίες αναφοράς σφαλμάτων. Κατά την εκτέλεση των δυο παρακάτω λειτουργιών είναι δυνατό να χαθούν δεδομενογραφήματα. Αυτή η τρίτη λειτουργία σκοπό έχει την ενημέρωση του κόμβου πηγή για την σχετική απώλεια.

Το IP datagram αποτελείται από δυο τμήματα την επικεφαλίδα και το κείμενο. Η επικεφαλίδα έχει μήκος από 20 μέχρι 60 byte και περιέχει πλήθος πεδίων. Ίσως τα πιο σημαντικά από αυτά είναι τα πεδία (μεγέθους 4 byte το καθένα) διεύθυνσης πηγής (source address) και διεύθυνσης προορισμού (destination address), τα οποία δείχνουν τον αριθμό δικτύου και τον αριθμό host.

5.3.2 Διευθύνσεις IP

Κάθε host και κάθε δρομολογητής στο Internet έχει μια διεύθυνση IP, που κωδικοποιεί τον αριθμό του δικτύου του καθώς και τον αριθμό του host. Ο συνδυασμός αυτό είναι μοναδικός δεν υπάρχει περίπτωση δυο μηχανήματα να έχουν την ίδια διεύθυνση IP. Όλες οι διευθύνσεις IP έχουν μήκος 32 bit και χρησιμοποιούνται στα πεδία source address και destination address που αναφέρθηκαν προηγούμενα. Οι δομές που χρησιμοποιούνται για τις διευθύνσεις IP φαίνονται στην συνέχεια. Οι μηχανές που είναι συνδεδεμένες σε πολλά δίκτυα έχουν διαφορετική διεύθυνση IP σε κάθε δίκτυο.

Οι δομές για τις κατηγορίες A, B, C, D επιτρέπουν έως και 126 δίκτυα με 16 εκατομμύρια host το καθένα, 16382 δίκτυα με το πολύ 64K host, 2 εκατομμύρια δίκτυα με το πολύ 254 host το καθένα και την πολλαπλή διανομή. Οι διευθύνσεις των δικτύων που είναι αριθμοί 32 bit, είναι συνήθως γραμμένες σε δεκαδικό συμβολισμό με τελείες (dotted decimal notation). Σε αυτή τη μορφή κάθε ένα από τα 4 byte είναι γραμμένο ως δεκαδικός, από το 0 έως το 255. Για παράδειγμα η δεκαεξαδική

διεύθυνση C0290614 γράφεται ως 192.41.6.20. Η κατώτατη διεύθυνση IP είναι η 0.0.0.0 και η ανώτατη 255.255.255.255.

Σχήμα 5.3.2.1: Μορφές διευθύνσεων IP.

Η IP διεύθυνση 0.0.0.0 χρησιμοποιείται από τους host κατά την διάρκεια της εκκίνησής τους. Οι διευθύνσεις IP με αριθμό δικτύου 0 αναφέρονται στο τρέχον δίκτυο. Αυτές οι διευθύνσεις επιτρέπουν στα μηχανήματα να αναφέρονται στο δικό τους δίκτυο χωρίς να γνωρίζουν τον αριθμό του. Η διεύθυνση που αποτελείται μόνο από 1 επιτρέπει την εκπομπή στο τοπικό δίκτυο. Οι διευθύνσεις με κανονικό αριθμό δικτύου και 1 στο πεδίο του αριθμού host επιτρέπουν στις μηχανές να κοινοποιούν πακέτα εκπομπής σε απομακρυσμένα δίκτυα LAN, οπουδήποτε στο internet. Τέλος όλες οι διευθύνσεις της μορφής 127.xxx.yyy.zzz έχουν δεσμευτεί για έλεγχο με βρόχο επιστροφής (loopback). Επεξεργάζονται τοπικά και αντιμετωπίζονται ως εισερχόμενα πακέτα.

5.4 Τα Πρωτόκολλα Μεταφοράς του Internet

Το internet έχει δυο κύρια πρωτόκολλα στο στρώμα μεταφοράς, ένα με σύνδεση και ένα χωρίς σύνδεση. Το πρωτόκολλο με σύνδεση είναι το TCP. Το πρωτόκολλο χωρίς σύνδεση είναι το UDP. Επειδή το UDP είναι βασικά απλώς το IP με την προσθήκη μιας μικρής επικεφαλίδας, το ενδιαφέρον θα εστιαστεί στο TCP. Για περισσότερες πληροφορίες πάνω στο TCP [1], [2], [3], [4].

Το πρωτόκολλο ελέγχου μεταφοράς TCP σχεδιάστηκε ειδικά για να προσφέρει έναν αξιόπιστο συρμό byte από άκρο σε άκρο, μέσω ενός αξιόπιστου διαδικτύου. Ένα διαδίκτυο μπορεί να διαφέρει από ένα απλό δίκτυο, επειδή τα διαφορετικά του μέρη μπορεί να έχουν διαφορετικές τοπολογίες, διαφορετικό εύρος ζώνης, διαφορετικές καθυστερήσεις, διαφορετικά μεγέθη πακέτων και άλλες διαφορετικές παραμέτρους. Το TCP σχεδιάστηκε έτσι ώστε να προσαρμόζεται δυναμικά στις ιδιότητες του διαδικτύου και να είναι ανθεκτικό ως προς πολλά είδη αστοχιών.

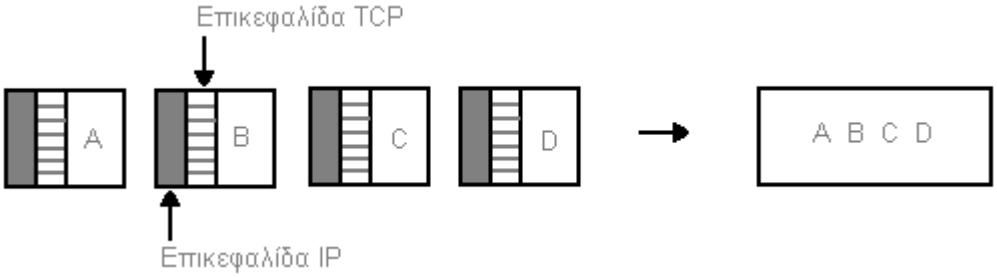
Κάθε μηχανή που υποστηρίζει το TCP έχει μια οντότητα μεταφοράς TCP, η οποία είτε είναι μια διεργασία χρήστη ή τμήμα του πυρήνα που διαχειρίζεται τους συρμούς TCP και αποτελεί τη διεπαφή με το στρώμα IP. Η οντότητα TCP δέχεται συρμούς δεδομένων χρήστη από τοπικές διεργασίες, τους τεμαχίζει, σε κομμάτια που δεν υπερβαίνουν τα 64 Kbyte και στέλνει κάθε κομμάτι ως ένα ξεχωριστό IP datagram. Όταν δεδομενογραφήματα IP που περιέχουν δεδομένα TCP φτάνουν στην μηχανή τα αναλαμβάνει η οντότητα TCP, η οποία ανασυνθέτει τους αρχικούς συρμούς byte.

Το στρώμα IP δεν εξασφαλίζει ότι τα datagrams θα μεταδοθούν σωστά και έτσι επαφίεται στο TCP να διαπιστώσει την εκπνοή του χρόνου και να επαναλάβει την μετάδοση, αν χρειάζεται. Τα datagrams που καταφέρνουν να φτάσουν, είναι πιθανόν να φθάσουν σε λάθος σειρά. Πάλι επαφίεται στο TCP να τα συναρμολογήσει σε μηνύματα με τη σωστή σειρά. Συνοπτικά το TCP πρέπει να προσφέρει την αξιοπιστία που θέλει ο χρήστης και την οποία δεν μπορεί να του προσφέρει το IP.

Για να αποκτηθεί η υπηρεσία TCP, ο αποστολέας και ο παραλήπτης δημιουργούν ακραία σημεία που αποκαλούνται υποδοχές (sockets). Κάθε υποδοχή έχει έναν αριθμό υποδοχής που αποτελείται από την διεύθυνση IP του host και από έναν τοπικό αριθμό 16 bit για κάθε host, αποκαλούμενου θύρα (port). Για να πάρουμε υπηρεσίες TCP, πρέπει να εγκατασταθεί ρητά μια σύνδεση μεταξύ μιας υποδοχής του μηχανήματος αποστολής με μια υποδοχή στο μηχάνημα λήψης.

Μια υποδοχή μπορεί να χρησιμοποιηθεί για πολλές συνδέσεις ταυτόχρονα. Με άλλα λόγια, δυο ή περισσότερες συνδέσεις μπορεί να τερματίζουν στην ίδια υποδοχή. Οι συνδέσεις αναγνωρίζονται από τις ταυτότητες (identifiers) των υποδοχών στα δυο άκρα, δηλαδή [υποδοχή1, υποδοχή2].

Οι θύρες με αριθμό κάτω από 1024 ονομάζονται πασίγνωστες θύρες (well known ports) και δεσμεύονται για τυποποιημένες υπηρεσίες. Όλες οι συνδέσεις TCP είναι πλήρως αμφίδρομες και από σημείο σε σημείο. Πλήρως αμφίδρομες σημαίνει ότι η κίνηση μπορεί να κυκλοφορήσει και προς τις δυο κατευθύνσεις ταυτόχρονα. Από σημείο σε σημείο σημαίνει ότι η σύνδεση έχει ακριβώς δυο ακραία σημεία. Μία σύνδεση TCP είναι ένας συρμός byte και όχι ένας συρμός μηνυμάτων. Τα όρια των μηνυμάτων δεν διατηρούνται από το ένα άκρο στο άλλο. Για παράδειγμα, εάν η διαδικασία αποστολής γράψει τέσσερις φορές 512 byte στον συρμό TCP, αυτά τα δεδομένα μπορούν να αποδοθούν στην λαμβάνουσα διεργασία σαν τέσσερις ομάδες των 512 byte, δυο ομάδες των 1024 byte, μία ομάδα των 2048 byte ή κάποιον άλλο τρόπο. Δεν υπάρχει τρόπος ώστε ο παραλήπτης να βρει τις μονάδες με τις οποίες γράφτηκαν τα δεδομένα.



Σχήμα 5.4.1: Τέσσερα τεμάχια των 512 byte μεταδόθηκαν ως ξεχωριστά IP datagrams, 2048 byte δεδομένων παραδίδονται στην εφαρμογή προορισμό.

Όταν μια εφαρμογή περνάει δεδομένα στο TCP, το TCP μπορεί κατά την κρίση του να τα στείλει αμέσως ή να τα τοποθετήσει σε χώρο προσωρινής

αποθήκευσης (ώστε να συγκεντρώσει και να στείλει μεγαλύτερο όγκο δεδομένων με τη μία). Μερικές φορές, παρόλα αυτά, η εφαρμογή θέλει πραγματικά να αποσταλούν τα δεδομένα αμέσως. Οι εφαρμογές, για να προκαλέσουν τη έξοδο των δεδομένων, μπορούν να χρησιμοποιήσουν τη σημαία PUSH που λέει στο TCP να μην καθυστερήσει την μετάδοση.

Κάθε byte σε μια σύνδεση TCP έχει τον δικό του αύξοντα αριθμό των 32 bit. Για έναν host που μεταδίδει σε πλήρη ταχύτητα σε δίκτυο LAN των 10 Mbps οι αύξοντες αριθμοί θα αναδιπλωθούν σε περίπου μια ώρα, αλλά στην πράξη θα πάρει περισσότερο. Οι αύξοντες αριθμοί χρησιμοποιούνται τόσο για τις επαληθεύσεις όσο και για τον μηχανισμό του παραθύρου, ο οποίος χρησιμοποιεί ξεχωριστά πεδία επικεφαλίδας των 32 bit.

Οι αποστέλλουσες και λαμβάνουσες οντότητες TCP ανταλλάσσουν δεδομένα με την μορφή τεμαχίων. Το τεμάχιο (segment) αποτελείται από μια σταθερή επικεφαλίδα των 20 byte, ακολουθούμενη από μηδέν ή περισσότερα byte δεδομένων. Το λογισμικό του TCP αποφασίζει για το μέγεθος των τεμαχίων. Μπορεί να συσσωρεύσει δεδομένα από πολλές εγγραφές σε ένα τεμάχιο ή να διαχωρίσει δεδομένα μιας εγγραφής σε πολλαπλά τεμάχια. Δυο περιορισμοί υπαγορεύουν το μέγεθος του τεμαχίου. Πρώτον, κάθε τεμάχιο, συμπεριλαμβανομένης της επικεφαλίδας TCP, πρέπει να χωράει σε ένα ωφέλιμο φορτίο IP. Δεύτερον, κάθε δίκτυο έχει μια μέγιστη μονάδα μεταφοράς (Maximum Transfer Unit) και κάθε τεμάχιο πρέπει να χωράει στην MTU. Στην πράξη η MTU είναι εν γένει λίγες χιλιάδες byte και έτσι ορίζει το άνω όριο του μεγέθους τεμαχίου. Εάν ένα τεμάχιο περάσει μέσα από μια σειρά δικτύων χωρίς να διασπαστεί και κατόπιν συναντήσει ένα δίκτυο που έχει μικρότερη MTU από το τεμάχιο, ο δρομολογητής που βρίσκεται στο όριο κομματιάζει το τεμάχιο σε δυο ή περισσότερα τεμάχια.

Ένα τεμάχιο, που είναι πολύ μεγάλο για το δίκτυο το οποίο πρέπει να διασχίσει, κομματιάζεται σε πολλά τεμάχια από τον δρομολογητή. Το κάθε νέο τεμάχιο αποκτά δικές του επικεφαλίδες TCP και IP, κατά συνέπεια ο τεμαχισμός από τους δρομολογητές αυξάνει την συνολική επιβάρυνση (αφού κάθε πρόσθετο τεμάχιο προσθέτει 40 επιπλέον byte πληροφοριών επικεφαλίδας).

Το βασικό πρωτόκολλο που χρησιμοποιείται από τις οντότητες TCP είναι το πρωτόκολλο του ολισθαίνοντος παραθύρου. Όταν ο αποστολέας μεταδίδει ένα τεμάχιο, εκκινεί επίσης έναν χρονομετρητή. Όταν το τεμάχιο φτάσει στον προορισμό του, η λαμβάνουσα οντότητα TCP επιστρέφει ένα τεμάχιο (με δεδομένα εάν υπάρχουν, αλλιώς χωρίς δεδομένα) που μεταφέρει αριθμό επαλήθευσης ίσο με τον αύξοντα αριθμό που περιμένει να δεχτεί. Αν το χρονόμετρο του αποστολέα λήξει πριν λάβει την επαλήθευση, τότε ο αποστολέας μεταδίδει το τεμάχιο ξανά.

Για την αποφυγή καταστάσεων συμφόρησης το πρωτόκολλο TCP προτείνει δυο μηχανισμούς που πλαισιώνουν το πρωτόκολλο του ολισθαίνοντος παραθύρου. Αυτοί οι μηχανισμοί είναι:

- Ο μηχανισμός της πολλαπλασιαστικής μείωσης (multiplicative decrease congestion avoidance) σύμφωνα με τον οποίο σε περίπτωση απώλειας ενός πακέτου το παράθυρο συμφόρησης μειώνεται στο μισό (μέχρι ένα ελάχιστο ίσο με ένα πακέτο). Αφού μειωθεί το παράθυρο στο μισό για τα πακέτα που παραμένουν σε αυτό, αυξάνεται ο χρονομετρητής εκθετικά (exponential backoff).
- Αργού ξεκινήματος (slow start recovery). Όταν η κυκλοφορία ξεκινά σε μια καινούργια σύνδεση, ή αυξάνεται μετά από μια κατάσταση συμφόρησης, το μέγεθος του παραθύρου συμφόρησης τίθεται να είναι ένα πακέτο. Με κάθε άφιξη επιβεβαίωσης (εννοείται χωρίς επαναμετάδοση) αυξάνεται κατά ένα.

Κεφάλαιο 6

Πλατφόρμα Υλοποίησης Λογισμικού Java

6 Πλατφόρμα Υλοποίησης Λογισμικού Java

6.1 Επισκόπηση της Πλατφόρμας Java 2

Η ραγδαία εξάπλωση του Διαδικτύου (Internet) και του Παγκόσμιου Ιστού (World-Wide-Web) δημιούργησαν την ανάγκη νέων τρόπων ανάπτυξης και διανομής του λογισμικού. Οι απαιτήσεις αυτές οδήγησαν στην δημιουργία της γλώσσας προγραμματισμού Java από την εταιρία Sun Microsystems. Η Java σχεδιάστηκε με σκοπό την ανάπτυξη εφαρμογών που τρέχουν σε ετερογενή δικτυακά περιβάλλοντα.

6.1.1 Ιστορία της Γλώσσας Java

Το 1991, σχηματίστηκε από την Sun Microsystems μια ερευνητική ομάδα με την ονομασία “Green Project”, με σκοπό να αναπτύξει λογισμικό για έλεγχο ηλεκτρονικών συσκευών ευρείας κατανάλωσης. Οι ερευνητές ήλπιζαν να αναπτύξουν μια γλώσσα προγραμματισμού που θα λειτουργούσε σε έξυπνες συσκευές στο μέλλον, όπως διαλογικές τηλεοράσεις ή διαλογικές τοστιέρες. Οι ερευνητές της Sun ήθελαν επίσης αυτές οι συσκευές να επικοινωνούν μεταξύ τους, έτσι ώστε η μια συσκευή να μπορεί να δώσει πληροφορίες σε μια άλλη.

Για να προχωρήσουν την έρευνα τους, οι ερευνητές της ομάδας Green ανέπτυξαν μια πρωτότυπη συσκευή με όνομα Star7, ένα μηχανισμό τύπου remote control που μπορούσε να επικοινωνεί με άλλες συσκευές του ίδιου τύπου. Η αρχική ιδέα ήταν να αναπτύξουν το λειτουργικό σύστημα Star7 σε C++, την δημοφιλή γλώσσα προγραμματισμού, που αναπτύχθηκε από τον Bjarne Stroustrup. Αλλά όμως το μέλος της ομάδας James Gosling ενοχλήθηκε από την λειτουργικότητα της C++, οπότε κλείστηκε στο γραφείο του και έγραψε μια νέα γλώσσα προγραμματισμού, που μπορούσε να χειριστεί καλύτερα το Star7. Η γλώσσα ονομάστηκε Oak, ενώ η Sun αργότερα ανακάλυψε ότι το όνομα Oak χρησιμοποιούταν ήδη και αναγκάστηκε να το αλλάξει σε Java.

Επειδή σχεδιάστηκε για οικιακές συσκευές και όχι για PC, η Java έπρεπε να είναι μικρή, αποδοτική και εύκολα μεταφερτή στις πολλές οικιακές συσκευές. Έπρεπε επίσης να είναι αξιόπιστη. Οι άνθρωποι έχουν μάθει να ζουν με τις περιστασιακές καταρρεύσεις των συστημάτων τους. Δεν είναι όμως εύκολο να έχουν τοστιέρα που καίει συνεχώς τα τοστ τους.

Αν και η Java αρχικά δεν εργάστηκε σαν εργαλείο ανάπτυξης διαλογικών οικιακών συσκευών, τα πράγματα που την έκαναν καλή για το Star7 αποδείχτηκαν ότι αποδίδουν εξίσου καλά και για το World Wide Web.

Επίσης η Java μπορεί να χρησιμοποιηθεί σαν μια γλώσσα προγραμματισμού γενικής χρήσης για ανάπτυξη λογισμικού, που μπορεί να εκτελείται σε διαφορετικές πλατφόρμες.

Για να επιδειχθούν οι δυνατότητες της Java και να διασωθεί το ερευνητικό της έργο, δημιουργήθηκε ένας εξεταστής Web το 1994, ο οποίος μπορούσε να εκτελεί μικροεφαρμογές Java. Ο εξεταστής επέδειξε δύο πράγματα για την Java: τι προσέφερε στο World Wide Web και τι είδους προγράμματα μπορούσε να δημιουργήσει. Οι

προγραμματιστές Patrick Naughton και Jonathan Payne χρησιμοποίησαν την Java για να δημιουργήσουν τον εξεταστή, που αρχικά ονομάστηκε WebRunner αλλά μετονομάστηκε σε HotJava.

Αν και, η Java και ο εξεταστής HotJava έτυχαν ιδιαίτερης προσοχής στην κοινότητα του Web, η γλώσσα απογειώθηκε πραγματικά όταν η Netscape έγινε η πρώτη εταιρεία που έδωσε άδεια χρήσης στην γλώσσα τον Αύγουστο του 1995.

6.1.2 Εκδόσεις της Γλώσσας Java

Η Sun έχει εκδώσει τρεις βασικές εκδόσεις της γλώσσας Java:

- Java 1.0 - Ευρύτερα υποστηριζόμενη από εξεταστές Web
- Java 1.1 - Μια έκδοση της άνοιξης του 1997, με βελτιώσεις στην διασύνδεση χρήστη, στον χειρισμό συμβάντων και μεγαλύτερη συνέπεια σε όλη την γλώσσα
- Java 2 - Η νέα έκδοση, που εκδόθηκε σαν έκδοση βήτα τον Δεκέμβριο του 1997 ως Java Development Kit (JDK) 1.2, και τελικά εκδόθηκε τον Δεκέμβριο του 1998 με την ονομασία "Java 2" [1].

6.1.3 Χαρακτηριστικά της Γλώσσας Java

Η πλατφόρμα JAVA 2 υποστηρίζει τα ακόλουθα χαρακτηριστικά:

1.Η όλη διαδικασία προγραμματισμού πραγματοποιείται στην αντικειμενοστραφή γλώσσα JAVA. Συντακτικά, αυτή είναι παρόμοια με την C++, αλλά οι δύο γλώσσες έχουν βασικές διαφορές. Μία από τις μεγαλύτερες διαφορές βρίσκεται στην διαχείριση και στην αναφορά αντικειμένων. Η γλώσσα προγραμματισμού JAVA δεσμεύει και αποδεσμεύει μνήμη αυτόματα καθώς το πρόγραμμα δημιουργεί και καταστρέφει αντικείμενα. Οι προγραμματιστές της C++ πρέπει να δεσμεύουν και να ελευθερώνουν μνήμη με σαφή τρόπο. Δυσάρεστες καταστάσεις που περιλαμβάνουν διαρροές μνήμης και απρόβλεπτες συμπεριφορές παρατηρούνται καθώς τα προγράμματα προσπαθούν να χρησιμοποιήσουν αντικείμενα που δεν χρησιμοποιούνται πλέον. Όπου οι προγραμματιστές σε JAVA δημιουργούν και χρησιμοποιούν, προβλέψιμες και ασφαλείς, άμεσες αναφορές (references) σε αντικείμενα, οι προγραμματιστές σε C++ χρησιμοποιούν δείκτες (pointers), μια πρακτική η οποία εγκυμονεί κινδύνους από διάφορες απόψεις. Οι δείκτες μπορούν να αλλάξουν τιμή άμεσα, ορίζοντας τους να δείχνουν σε μνήμη που δεν τους ανήκει. Οι δείκτες μπορούν να μετατραπούν (cast) σε οποιονδήποτε άλλο τύπο αντικειμένου, διευκολύνοντας έτσι μια εφαρμογή να απαιτεί υπηρεσίες από ένα αντικείμενο, το οποίο δεν παρέχει. Σε κώδικα JAVA, οι αναφορές σε αντικείμενα μπορούν να δείχνουν μόνο σε έγκυρα αντικείμενα και δεν μπορούν να μετατραπούν σε πρωτογενείς (arbitrary) τύπους. Μία ακόμα θεμελιώδης διαφορά ανάμεσα στις δύο γλώσσες βρίσκεται στους μηχανισμούς με τους οποίους δίνουν σε ένα αντικείμενο πολλαπλούς τύπους. Η C++ υποστηρίζει πολλαπλή κληρονομικότητα (multiple inheritance), το οποίο σημαίνει ότι οι κλάσεις μπορούν να έχουν πολλαπλές βασικές κλάσεις. Αυτός ο μηχανισμός οδηγεί σε ασάφεια και σύγχυση. Η γλώσσα JAVA αντικαθιστά αυτόν το μηχανισμό με ένα πιο ξεκάθαρο είδος αφαιρετικής δομής, τις

διεπαφές (interfaces). Μία κλάση JAVA μπορεί να έχει μία μόνο βασική κλάση (parent class) αλλά μπορεί να υλοποιεί πολλαπλές διεπαφές.

2.Η πλατφόρμα *JAVA* χρησιμοποιεί αρχιτεκτονική εικονικής μηχανής (virtual machine), ή (όπως αλλιώς είναι γνωστή) *διερμηνέα JAVA* (**Java Interpreter ή Java runtime**). Αυτό είναι προτέρημα από διάφορες πλευρές:

Η εικονική μηχανή μπορεί να υλοποιηθεί ώστε να λειτουργεί σε διάφορα λειτουργικά συστήματα και πλατφόρμες με σταθερότητα. Επιπροσθέτως η εικονική μηχανή παρέχει αυστηρό έλεγχο σε εκτελούμενα προγράμματα, δίνοντας έτσι την δυνατότητα για ασφαλή εκτέλεση αναξιόπιστου κώδικα. Δηλαδή η εικονική μηχανή παίρνει μεταγλωττισμένα προγράμματα Java από προγράμματα *πηγαίου κώδικα* και μετατρέπει τις εντολές τους σε εντολές που μπορεί να χειριστεί ένα λειτουργικό σύστημα. Το ίδιο μεταγλωττισμένο πρόγραμμα που υπάρχει σε μια μορφή ψευτοκώδικα που καλείται *bytecode* μπορεί να εκτελείται σε οποιαδήποτε άλλη πλατφόρμα και λειτουργικό σύστημα που διαθέτει μια εικονική μηχανή της Java.

Πηγαίος κώδικας (source code) *είναι το σύνολο των προτάσεων προγραμματισμού που εισάγει ένας προγραμματιστής σε ένα κειμενογράφο όταν δημιουργεί ένα πρόγραμμα.*

Bytecode *είναι η έκδοση κώδικα μηχανής της εικονικής μηχανής της Java, δηλαδή οι εντολές που καταλαβαίνει με άμεσο τρόπο.*

Ο πηγαίος κώδικας μεταγλωττίζεται σε bytecode έτσι ώστε να μπορεί να εκτελεστεί από μια εικονική μηχανή Java.

3.Η πλατφόρμα Java περιλαμβάνει μια εκτεταμένη συλλογή από συγκεκριμένες βιβλιοθήκες κλάσεων που ονομάζονται Διεπαφές Προγραμματισμού Εφαρμογών (Application Programming Interfaces σύντομα APIs). Αυτά υποστηρίζουν οτιδήποτε μπορεί μια εφαρμογή να χρειάζεται, από περιβάλλοντα χρήστη μέχρι κρυπτογραφία.

4.Υποστηρίζεται δημιουργία ανεξάρτητων εφαρμογών και applets.

Applets ονομάζονται προγράμματα που περιλαμβάνονται σε σελίδες Γλώσσας Σήμανσης Υπερκειμένου (HyperText Markup Language / HTML) και εκτελούνται από τον εξεταστή (Web browser).

5.Η Java είναι κατανεμημένη (distributed), δηλαδή ένα πρόγραμμα Java είναι δυνατό να εγκατασταθεί και να λειτουργήσει διαμέσου του Διαδικτύου ή ακόμα και από διαφορετικές συσκευές στο Διαδίκτυο.

6.Η πλατφόρμα Java είναι ασφαλής (secure), καθώς στο Διαδίκτυο ελλοχεύουν πολλοί κίνδυνοι για τον χρήστη – παραλήπτη μιας δικτυακής εφαρμογής, ενώ η Java έχει σχεδιαστεί έτσι ώστε να ελαχιστοποιείται η πιθανότητα προσβολής του συστήματος του χρήστη από κάποιο πρόγραμμα γραμμένο για αυτό το σκοπό.

7. Υποστηρίζει ταυτοχρονισμό (multithreading). Η Java υποστηρίζει εγγενώς την χρήση νημάτων (Threads). Προκειμένου να το πετύχει αυτό σε συστήματα με έναν επεξεργαστή, το **Java runtime system (interpreter)** υλοποιεί ένα δικό του

χρονοδρομολογητή (scheduler), ενώ σε συστήματα που υποστηρίζουν πολυεπεξεργασία η δημιουργία νημάτων ανατίθεται στο λειτουργικό σύστημα. Η όλη διαδικασία είναι αόρατη τόσο στον προγραμματιστή όσο και στον χρήστη.

8. Τέλος η Java υποστηρίζει εφαρμογές πολυμέσων τόσο με την ευελιξία της σαν γλώσσα προγραμματισμού όσο και με τις πλούσιες και συνεχώς εμπλουτιζόμενες βιβλιοθήκες [2], [3].

6.1.4 Τομείς της Γλώσσας Java

Η **πλατφόρμα Java** αποτελείται λοιπόν από την **γλώσσα προγραμματισμού Java**, από την **εικονική μηχανή Java** και από τα **Java APIs** (βιβλιοθήκες). Είναι σχεδιασμένη να αναφέρεται σε ένα τεράστιο πλήθος από υπολογιστικά συστήματα, σε οτιδήποτε από έξυπνες κάρτες (smart cards) έως εταιρικούς εξυπηρετητές. Έτσι είναι αναγκαίος ο διαχωρισμός της πλατφόρμας Java σε τρεις τομείς-εκδόσεις:

- Java 2, κανονική έκδοση (Standard Edition /J2SE) η οποία είναι σχεδιασμένη για επιτραπέζιους υπολογιστές (desktop computers). Συνήθως είναι εγκαταστημένη επάνω σε λειτουργικά συστήματα όπως Linux, Solaris ή Microsoft Windows.
- Java 2, εταιρική έκδοση (Enterprise Edition/J2EE) η οποία είναι μια πλήρης πλατφόρμα για εφαρμογές με πολλούς χρήστες. Στηρίζεται στην J2SE και προσθέτει APIs για προγραμματισμό από την πλευρά του εξυπηρετητή.
- Java 2, μικρο-έκδοση (Micro Edition/ J2ME) η οποία είναι ένα σύνολο από τεχνολογίες και πρότυπα ανεπτυγμένα για μικρές συσκευές όπως έξυπνες κάρτες, κινητά τηλέφωνα και υπολογιστές παλάμης (Palmtop). Η J2ME χρησιμοποιεί υποσύνολα των στοιχείων της J2SE όπως μικρότερες εικονικές μηχανές και περιορισμένα APIs.

Πρότυπα των J2SE, J2ME, J2EE αναπτύσσονται υπό την αιγίδα του Συμβουλίου Εξέλιξης της Java (**Java Community Process/JCP**) [4]. Ένα πρότυπο ξεκινά να υπάρχει ως Αίτηση Προτύπου Java (**Java Specification Request/JSR**). Μια ομάδα από ειδικούς που περιλαμβάνει αντιπροσώπους από ενδιαφερόμενες εταιρίες συγκεντρώνεται ώστε να δημιουργήσει το πρότυπο. Μετά το JSR περνά από διάφορα στάδια στο JCP μέχρι να τελειοποιηθεί. Σε κάθε JSR δίνεται ένας κωδικός. Τα πρότυπα της J2ME συνήθως αναφέρονται με τον κωδικό τους [3].

6.2 Επισκόπηση της Java 2 Micro Edition (J2ME)

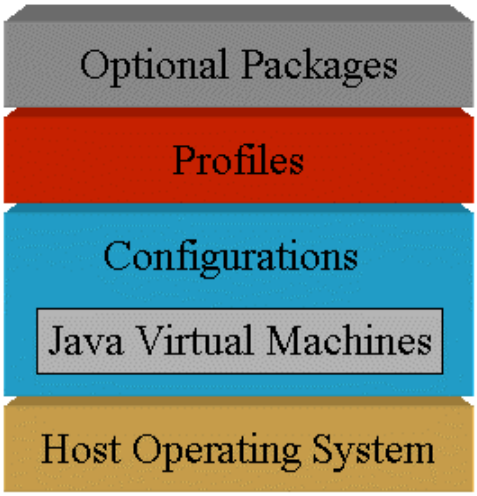
Αντιθέτως προς την J2SE, η J2ME δεν είναι είδος λογισμικού, ούτε ένα μοναδικό πρότυπο. Αυτή η διαφορά μπορεί να προκαλέσει ασάφειες, ακόμα και για προγραμματιστές που ήδη είναι εξοικειωμένοι με την J2SE. Η J2ME είναι μια συλλογή από τεχνολογίες και πρότυπα που είναι σχεδιασμένα για διαφορετικούς τομείς της αγοράς μικρών συσκευών. Επειδή ακριβώς η J2ME αναφέρεται σε μια τέτοια ποικιλία

συσκευών, δεν θα είχε νόημα η προσπάθεια δημιουργίας μιας λύσης που να «χωρά σε όλα» (*one-size-fits-all)*.

Για αυτό λοιπόν η J2ME είναι διαιρεμένη σε **συνθέσεις (*configurations)*** και **προφίλ (*profiles*)**.

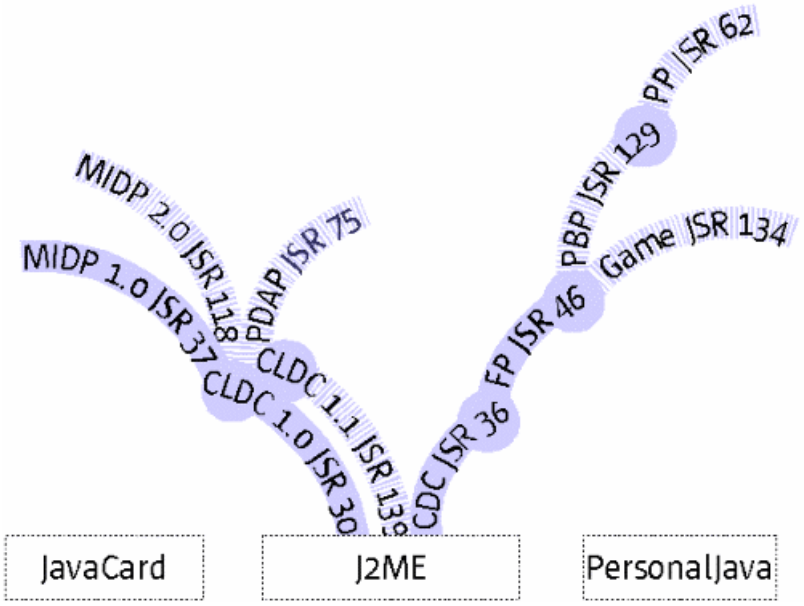
Συνθέσεις (*configurations)* ονομάζονται πρότυπα που περιλαμβάνουν μια εικονική μηχανή (Virtual Machine/VM) για να εκτελείται bytecode και ένα βασικό σύνολο από APIs που μπορούν να χρησιμοποιηθούν σε μια συγκεκριμένη κατηγορία συσκευών. Για παράδειγμα μια σύνθεση μπορεί να έχει σχεδιασθεί για συσκευές με μνήμη τυχαίας προσπέλασης (RAM/ Random Access Memory) λιγότερη από 512 KB. Η εικονική μηχανή είναι είτε πλήρης εικονική μηχανή Java (full Java Virtual Machine/JVM), όπως περιγράφεται στο πρότυπο, είτε ένα υποσύνολο αυτής. Το σύνολο των APIs είναι ανάλογα με την περίσταση υποσύνολο της J2SE.

Τα **Προφίλ (*profiles*)** δημιουργούνται επάνω σε μια σύνθεση αλλά προσθέτουν περισσότερα συγκεκριμένα APIs ώστε να δημιουργηθεί ένα ολοκληρωμένο περιβάλλον για την ανάπτυξη εφαρμογών. Ενώ δηλαδή μία σύνθεση περιγράφει μια εικονική μηχανή και ένα στοιχειώδες σύνολο από APIs, δεν επιτρέπει με εύχρηστο τρόπο την δημιουργία εφαρμογών. Τα προφίλ συνήθως περιλαμβάνουν APIs για τον κύκλο ζωής της εφαρμογής (application life circle), για το περιβάλλον χρήστη (user interface) και για μόνιμη αποθήκευση. Αυτά τα στοιχεία αποτελούν την αρχιτεκτονική της J2ME όπως φαίνεται στο σχήμα 6.2.1 παρακάτω. Τον πυρήνα της J2ME αποτελεί μια σύνθεση, της οποίας η εικονική μηχανή αλληλεπιδρά με το λειτουργικό σύστημα της συσκευής. Ένα ή περισσότερα προφίλ προσθέτουν προγραμματιστικές βιβλιοθήκες που παρέχουν υπηρεσίες όπως Είσοδο/Εξοδο (Input-Output I/O) και γραφικό περιβάλλον χρήστη (Graphical User Interface /GUI). Μια σύνθεση και ένα προφίλ μαζί συνθέτουν ένα περιβάλλον εκτέλεσης (runtime environment). Ένας αριθμός προαιρετικών πακέτων επεκτείνουν την πλατφόρμα J2ME, προσφέροντας συγκεκριμένα APIs για χρήση υπαρχόντων και αναπτυσσόμενων τεχνολογιών (όπως Bluetooth και υπηρεσίες διαδικτύου). Τα προαιρετικά πακέτα προσφέρουν πρόσβαση σε υπηρεσίες που είναι χρήσιμες σε περισσότερες από μία κατηγορίες συσκευών, αλλά όχι σε όλες [5].



Σχήμα 6.2.1: J2ME Αρχιτεκτονική

Η παρούσα κατάσταση από ρυθμίσεις και προφίλ φαίνεται στο σχήμα 6.2.2. Για επεξήγηση των συντομογραφιών βλέπε τον πίνακα 6.2.1 παρακάτω. Κάθε πρότυπο έχει αναπτυχθεί από το JCP (Java Community Process). Στο διάγραμμα 6.6.2 τα ολοκληρωμένα πρότυπα συμβολίζονται με πυκνό χρωματισμό, ενώ τα πρότυπα που δεν έχουν ολοκληρωθεί με αχνό χρωματισμό. Επίσης φαίνεται ο σχετικός κωδικός JSR (Java Specification Request).

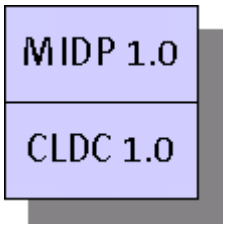


Σχήμα.6.6.2: Δέντρο J2ME

Όπως φαίνεται στο διάγραμμα, το δέντρο J2ME έχει δύο κύριες διακλαδώσεις. Η πρώτη βασίζεται στο **Connected, Limited Device Configuration (CLDC)**. Αυτή η σύνθεση είναι για μικρότερες συσκευές με μη-μόνιμη δικτυακή σύνδεση, όπως κινητά τηλέφωνα και Ψηφιακοί Προσωπικοί Βοηθοί (Personal Digital Assistant). Το **Mobile Information Device Profile (MIDP)**, το οποίο βασίζεται στο CLDC είναι το πρώτο ολοκληρωμένο προφίλ και έτσι το πρώτο ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών σε J2ME. Η άλλη κύρια διακλάδωση του δέντρου J2ME βασίζεται στο **Connected Device Configuration (CDC)**. Αυτή η σύνθεση είναι κατάλληλη για μεγαλύτερες συσκευές (όσον αφορά τη μνήμη και την επεξεργαστική ισχύ) με σταθερές συνδέσεις δικτύου. Ευφυείς οικιακές συσκευές και υψηλών επιδόσεων PDA όπως το i-Paq της Compaq είναι χαρακτηριστικά παραδείγματα αυτής της σύνθεσης. Το **Θεμελιώδες Προφίλ (Foundation Profile)** επεκτείνει το CDC και λειτουργεί ως η βάση για άλλα προφίλ. Παρέχει θεμελιώδη API που προέρχονται από το J2SE, συμπεριλαμβάνοντας κλάσεις και διεπαφές από τα πακέτα *java.lang, java.io, java.security, java.util*.

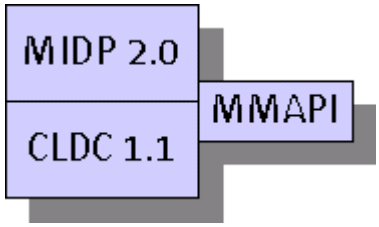
Η πλατφόρμα Java που αναφέρεται στις μικρές συσκευές περιλαμβάνει επίσης τις τεχνολογίες **Java Card,** για μικρές κάρτες (smart cards) και **PersonalJava.** Η PersonalJava ήταν ένα περιβάλλον εφαρμογών βασισμένο στο Java Development Kit 1.1 και είχε μια ανεξάρτητη ύπαρξη πριν δημιουργηθεί η J2ME. Το **Προσωπικό προφίλ (Personal Profile)** συνενώνει την PersonalJava με την J2ME καθώς η επόμενη έκδοση της PersonalJava είναι ένα πρότυπο προφίλ της J2ME βασισμένο στην CDC σύνθεση.

Οι συσκευές υλοποιούν μια πλήρη στοίβα λογισμικού, η οποία περιλαμβάνει μία σύνθεση, ένα προφίλ και προαιρετικά APIs. Για παράδειγμα, σύγχρονα κινητά τηλέφωνα συμβατά με J2ME υλοποιούν την εξής στοίβα



Σχήμα 6.2.3: Παράδειγμα στοίβας J2ME

ενώ τα κινητά τρίτης γενιάς υλοποιούν την εξής στοίβα, η οποία περιλαμβάνει τις βιβλιοθήκες Mobile Media API (MMAPI) και οι οποίες επιτρέπουν την αναπαραγωγή και την συλλογή δεδομένων πολυμέσων ήχου και βίντεο



Σχήμα 6.2.4: Παράδειγμα J2ME στοίβας κινητών τρίτης γενιάς

Ο επόμενος πίνακας περιλαμβάνει προηγούμενες και σύγχρονες ρυθμίσεις, προφίλ και βιβλιοθήκες.

Configurations		
JSR 30	CLDC 1.0	Connected, Limited Device Configuration
JSR 139	CLDC 1.1	Connected, Limited Device Configuration 1.1
JSR 36	CDC	Connected Device Configuration
Profiles		
JSR 37	MIDP 1.0	Mobile Information Device Profile
JSR 118	MIDP 2.0	Mobile Information Device Profile 2.0
JSR 75	PDAP	PDA Profile

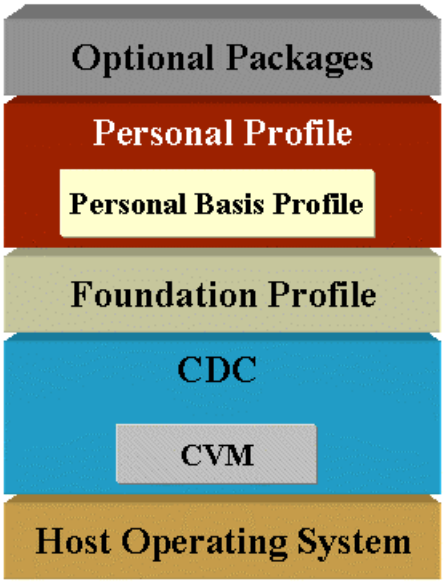
JSR 46	FP	Foundation Profile
JSR 129	PBP	Personal Basis Profile
JSR 62	PP	Personal Profile
JSR 134		Game Profile
APIs		
JSR 66		RMI Optional Package
JSR 80		Java USB API
JSR 82		Java APIs for Bluetooth
JSR 120		Wireless Messaging API
JSR 135	MMAPI	Mobile Media API
JSR 169		JDBC Optional Package for CDC/FP
JSR 179		Location API for J2ME
JSR 180		SIP API for J2ME
JSR 184		Mobile 3D Graphics API for J2ME

Πίνακας 6.2.1: Προηγούμενες και σύγχρονες ρυθμίσεις, προφίλ και βιβλιοθήκες J2ME

Για περισσότερα βλέπε στο [3].

6.3 CDC Configuration

Το σχήμα δείχνει πως ακριβώς υλοποιεί η σύνθεση CDC την αρχιτεκτονική της J2ME.



Σχήμα 6.2.1 J2ME CDC και σχετικές τεχνολογίες

Το CDC configuration είναι μια προσπάθεια προτύπου με κωδικό JSR 36 [4] για να τυποποιηθεί ένα σύνολο από φορητές οικιακές και ενσωματωμένες συσκευές. Παρέχει ένα σύνολο από βασικές βιβλιοθήκες και μια εικονική μηχανή Java κατάλληλη για χρήση από προφίλ σε κάθετες αγορές προϊόντων. Το CDC στοχεύει σε ισχυρές συσκευές που περιοδικά συνδέονται σε ένα δίκτυο όπως διαλογικές τηλεοράσεις, οικιακές συσκευές και συστήματα πλοήγησης αυτοκινήτου.

Το CDC περιέχει μια πλήρη εικονική μηχανή με δυνατότητες παρόμοιες με αυτές των εικονικών μηχανών της J2SE. Η κύρια διαφορά βρίσκεται στην μνήμη και στις δυνατότητες απεικόνισης των συσκευών για τις οποίες προορίζεται.

Πρέπει λοιπόν μια συσκευή για να υποστηρίζει το πρότυπο CDC της J2ME να ικανοποιεί τις εξής προϋποθέσεις:

- επεξεργαστής 32-bit
- διαθέσιμη μνήμη στο περιβάλλον Java τουλάχιστον 2 MB, συμπεριλαμβάνοντας τόσο την μνήμη RAM όσο και μνήμη flash ή ROM
- πλήρη λειτουργικότητα της εικονικής μηχανής Java σύμφωνα με τις προδιαγραφές
- προαιρετική ύπαρξη διεπαφής χρήστη (user interface)

Οι ελάχιστες βιβλιοθήκες API που υποστηρίζει το CDC (που επί της ουσίας είναι τα πακέτα που απαιτούνται από την J2SE v1.3) για να υλοποιηθεί και να εκτελεστεί μια εικονική μηχανή είναι:

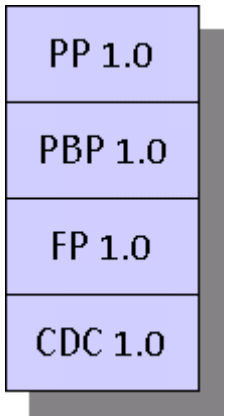
- java.lang : κλάσεις συστήματος εικονικής μηχανής
- java.io : κλάσεις εισόδου -εξόδου
- java.net : UDP (datagrams)
- java.util : κλάσεις γενικής χρήσης Java
- java.text : ελάχιστη υποστήριξη διεθνοποίησης
- java.security : ελάχιστη υποστήριξη ασφάλειας και κρυπτογράφησης για σειριοποίηση αντικειμένων

Η εικονική μηχανή του CDC καθορίζει το δικό της υποσύνολο χαρακτηριστικών εικονικής μηχανής Java και φέρει την ονομασία CVM. Η παρούσα υλοποίηση που είναι διαθέσιμη από την Sun Microsystems εκτελείται σε Linux ή VxWorks. Έχει αναπτυχθεί σχεδόν εξ' ολοκλήρου σε C και περιέχει ένα καλά τεκμηριωμένο στρώμα διασυνδέσεων που διευκολύνει οποιονδήποτε θέλει να σχεδιάσει μια νέα πλατφόρμα.

Τα προφίλ που έχουν αναπτυχθεί σύμφωνα με το CDC είναι τα Foundation Profile, Personal Profile και Personal Basis Profile [5].

6.4 Personal Profile

Το Personal Profile (JSR 62) [4] είναι ένα προφίλ της J2ME, μια προδιαγραφή για ένα τυπικό διερμηνέα Java. Είναι κομμάτι στοίβας λογισμικού που έχει σχεδιασθεί για συσκευές όπως το Sharp Zaurus και το Compaq iPaq, ενώ έχει σχηματιστεί επάνω στα Personal Basis Profile [4] και Foundation Profile [4], δηλαδή σε τελική ανάλυση επάνω στο Connected Device Configuration [4].



Σχήμα 6.4.1 Στοίβα λογισμικού Personal Profile

6.4.1 Foundation Profile

Το Foundation Profile [6] προσθέτει κυρίως J2SE API που έχουν παραληφθεί από το CDC. Συγκεκριμένα προσθέτει τα:

- `java.io`
- `java.lang`
- `java.net`
- `java.security`
- `java.security.cert`
- `java.text`
- `java.util`
- `java.util.jar`
- `java.util.zip`

και κλάσεις από τρία πακέτα που δεν υπάρχουν στο CDC:

- `java.security.acl`
- `java.security.interfaces`

- `ava.security.spec`

6.4.2 Personal Basis Profile

Το Personal Basis Profile είναι ένα υπερσύνολο του Foundation Profile με τρία νέα χαρακτηριστικά:

- το μοντέλο εφαρμογής Xlet που υιοθετήθηκε από τα Java TV API
- ένα υποσύνολο της Αφηρημένης Παραθυρικής Συλλογής Εργαλείων (Abstract Windowing Toolkit /AWT) της J2SE για την δημιουργία διεπαφών χρήστη βασισμένες αποκλειστικά σε ελαφρά (lightweight) συστατικά
- επικοινωνία Inter-Xlet (IXC) που αποτελεί υπερσύνολο του Remote Method Invocation (RMI) API

Συγκεκριμένα προσθέτει τα

- java.awt
- java.awt.color
- java.awt.event
- java.awt.image
- java.beans
- java.rmi
- java.rmi.registry

Σημειώνεται ότι το PBP δεν υποστηρίζει το RMI. Τα java.rmi και java.rmi.registry χρησιμοποιούνται για να υποστηρίξουν το IXC [6].

6.4.3 Ανάλυση του Personal Profile

Το Personal Profile αποτελεί μετεξέλιξη μιας παλαιότερης έκδοσης Java, της PersonalJava, η οποία έχει υλοποιηθεί σύμφωνα με το JDK 1.1.8. Η PersonalJava χρησιμοποιεί εικονική μηχανή όπως και η J2SE. Αλλά μετά την εμφάνιση του JDK 1.1.8, η PersonalJava και το J2SE έχουν εξελιχθεί ξεχωριστά. Ουσιαστικά, η PersonalJava βρίσκεται ανάμεσα σε J2SE και CLDC/MIDP της J2ME.

Η στοίβα του Personal Profile έχει δομηθεί επάνω στο CDC το οποίο ορίζει μια εικονική μηχανή και ένα στοιχειώδες σύνολο API [7].

Το Personal Profile είναι υπερσύνολο του Personal Basis Profile που προσθέτει τα εξής χαρακτηριστικά:

- υποστήριξη applet για δημιουργία εφαρμογών βασισμένων σε εξεταστές
- ένα πιο εκτεταμένο υποσύνολο AWT που συμπεριλαμβάνει και βαριά (heavyweight) συστατικά, γεγονότα (events), όπως επίσης και deprecated (μη-προτεινόμενα) API
- υποστήριξη προχείρου μέσω της υλοποίησης ενός υποσυνόλου του πακέτου java.awt.datatransfer

Επίσης προστίθενται κλάσεις από τρία πακέτα του J2SE 1.3:

- java.applet
- java.awt
- java.awt.datatransfer

Σημειώνουμε ότι τα επόμενα πακέτα υποστηρίζονται τόσο από το PersonalJava όσο και από το Personal Profile [6]:

- java.applet
- java.awt.datatransfer
- java.awt.event
- java.awt.image
- java.io
- java.lang
- java.reflect
- java.net
- java.security
- java.security.cert
- java.security.interfaces
- java.security.spec
- java.text
- java.util
- java.util.jar
- java.util.zip

6.5 Ασύρματη Τεχνολογία Java

Η ασύρματη τεχνολογία Java είναι η ένωση δύο τεράστιων τεχνολογιών: των ασύρματων επικοινωνιών και της πλατφόρμας Java. Η ασύρματη τεχνολογία Java περιλαμβάνει κομμάτια από J2ME, PersonalJava, J2ME και J2EE. Πρέπει να γίνει σαφής ο διαχωρισμός ανάμεσα στις δύο έννοιες. Η *ασύρματη τεχνολογία Java* και η *J2ME* δεν αναφέρονται στο ίδιο αντικείμενο. Η J2ME αναπτύσσεται τόσο σε ασύρματες όσο και σε μη-ασύρματες συσκευές. Για παράδειγμα συσκευές συμβατές με το CDC έχουν συνήθως σταθερή σύνδεση Ethernet. Από την άλλη πλευρά, η ασύρματη τεχνολογία Java δεν περιορίζεται στην J2ME αποκλειστικά. Είναι δυνατή για παράδειγμα η σύνδεση ενός φορητού υπολογιστή ή ενός υπολογιστή παλάμης, οι οποίοι τρέχουν J2SE εφαρμογές δια μέσου 802.11 ασυρμάτου τοπικού δικτύου [3].

6.6 Η πλατφόρμα Java στην Ανάπτυξη Ασύρματων Εφαρμογών

Η πλατφόρμα Java αποτελεί την άριστη επιλογή για ασύρματες εφαρμογές για τους εξής λόγους:

- Η πλατφόρμα Java είναι ασφαλής. Ο κώδικας σε γλώσσα Java εκτελείται πάντα με τους περιορισμούς της εικονικής μηχανής, η οποία παρέχει ένα ασφαλές περιβάλλον για να εκτελείται αποθηκευμένος κώδικας. Μια δυαδική εφαρμογή θα μπορούσε να προκαλέσει το σταμάτημα ή την κατάρρευση μιας συσκευής. Σε αντιδιαστολή, μια εφαρμογή Java στην χειρότερη περίπτωση μπορεί να προκαλέσει την κατάρρευση της εικονικής μηχανής και όχι της συσκευής.
- Η γλώσσα Java ενθαρρύνει τον αποδοτικό προγραμματισμό. Ο συλλέκτης σκουπιδιών προφυλάσσει τους προγραμματιστές από αμέτρητες ώρες προσπάθειας κάλυψης των διαρροών μνήμης. Ομοίως οι μηχανισμοί εξαιρέσεων της γλώσσας Java ενθαρρύνουν τους προγραμματιστές να δημιουργούν ενιαίο και σταθερό κώδικα.
- Η συμβατότητα είναι ένα μεγάλο προτέρημα της τεχνολογίας Java. Ένα εκτελέσιμο αρχείο μπορεί να εκτελεστεί σε διάφορες συσκευές. Μια εφαρμογή κατάλληλη για ένα προφίλ, θα εκτελεστεί σε όλες τις συσκευές συμβατές με αυτό το προφίλ. Αντιλαμβανόμενοι το τεράστιο πλήθος ασύρματων συσκευών, είναι προσόν η μη-ανάγκη διατήρησης μιας πληθώρας από υλοποιήσεις. Ακόμα και εάν μια εφαρμογή Java κάνει χρήση βιβλιοθηκών συγκεκριμένων εταιριών, εφαρμογές γραμμένες σε Java είναι πιο εύκολα τροποποιήσιμες για άλλες συσκευές από ότι εφαρμογές σε C ή C++. Επίσης μεγάλη είναι η ευκολία με την οποία διαμοιράζονται εφαρμογές σε μια συσκευή με την βοήθεια ενός ασύρματου δικτύου, διαδικασία που φέρει την ονομασία Δια Μέσου Αέρος Παροχή (Over-the-air /OTA provisioning). Εκτελέσιμες εφαρμογές μπορούν να μεταφερθούν από ένα εξυπηρετητή σε μια συσκευή χωρίς

όμως ασφάλεια. Επειδή ο κώδικας σε Java εκτελείται στην εικονική μηχανή μπορεί να εκτελεστεί με ασφάλεια [3].

6.7 Βασικές Έννοιες της Γλώσσας Java

Στο σημείο αυτό θα αναφερθούμε σε βασικές έννοιες της γλώσσας προγραμματισμού Java οι οποίες αποτελούν τα στοιχειώδη συστατικά επάνω στα οποία δομείται ένα πρόγραμμα σε Java. Τα συγκεκριμένα στοιχεία επίσης αποτελούν και τα χαρακτηριστικά γνωρίσματα της Java. Όταν λοιπόν συναντάμε αυτά τα χαρακτηριστικά τότε πρέπει να συμπεραίνουμε ότι ο κώδικας έχει υλοποιηθεί σε Java.

6.7.1 Threads

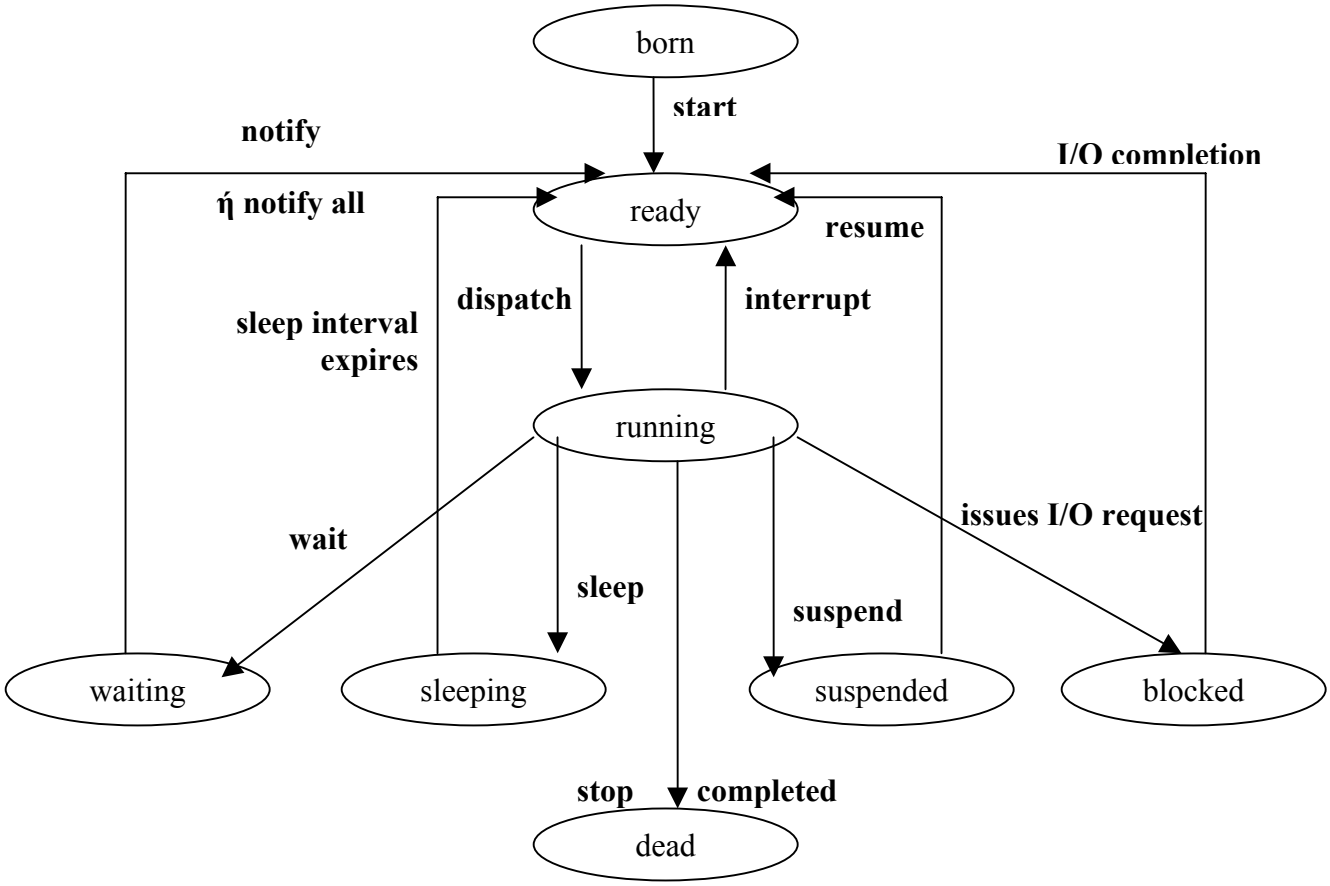
Πολλές γλώσσες προγραμματισμού όπως και η Java διαθέτουν εργαλεία για την υλοποίηση νημάτων threads στα προγράμματα τους. Αυτές οι γλώσσες καλούνται multithreading languages. Στον παραδοσιακό προγραμματισμό ένα πρόγραμμα που εκτελείται ονομάζεται διεργασία (process) και οι εντολές του εκτελούνται σειριακά η μία μετά την άλλη μέχρι το τέλος του. Σε αυτή την περίπτωση μπορούμε να πούμε ότι η process διαθέτει ένα μόνο thread. Στις multithreading γλώσσες προγραμματισμού υπάρχουν διεργασίες οι οποίες εκτελούν κάθε μια ξεχωριστά τον κώδικα του προγράμματος. Είναι δηλαδή σαν να έχουμε πολλά processes που εκτελούνται παράλληλα μέσα σε κάποιο αρχικό process.

6.7.1.1 Υλοποίηση των Threads

Οι ενέργειες που λαμβάνουν χώρα κατά την εκτέλεση ενός Thread ορίζονται στη μέθοδο run ενός Thread. Υπάρχουν δύο τρόποι για τον ορισμό της μεθόδου run ενός Thread.

- Δημιουργία υποκλάσης της κλάσης Thread και υπέρβαση (overriding) της run μεθόδου που υπάρχει στην Thread κλάση.
- Δημιουργία κλάσης που υλοποιεί το Runnable interface που ορίζεται στο πακέτο java.lang. Ένα τέτοιο αντικείμενο θα πρέπει όμως να συσχετιστεί με ένα νήμα περνώντας ως παράμετρος στον κατασκευαστή του νήματος. Εν συνεχεία, η κλήση της μεθόδου run() του νήματος θα ενεργοποιεί την μέθοδο run() του αντικειμένου [2].

Στο παρακάτω σχήμα παρουσιάζονται οι καταστάσεις ενός νήματος.



Σχήμα 6.7.1.1 Καταστάσεις ενός νήματος

6.7.1.2 Διακοπή ενός Thread

Η κλάση Thread συμπεριλαμβάνει την μέθοδο stop() για να σταματά ένα νήμα. Σύμφωνα με το επίσημο specification της Java2 [8] η μέθοδος αυτή αποδοκιμάζεται (deprecated). Αυτή η απόφαση προέκυψε επειδή ακριβώς είναι δομικά επισφαλής. Η διαδικασία σταματήματος ενός νήματος με την Thread.stop() προκαλεί το ξεκλείδωμα όλων των monitors που αυτό είχε κλειδώσει, σαν φυσική συνέπεια της εξαίρεσης ThreadDeath που διαδίδεται προς τα πάνω στην στοίβα που έχει δεσμευθεί στη μνήμη.

Σημειώνουμε ότι επόπτης είναι μια συλλογή κοινών μεταβλητών και διαδικασιών η οποία ικανοποιεί τον περιορισμό ότι κάθε στιγμή μόνο μία εξωτερική διεργασία μπορεί να εκτελεί μία διαδικασία του επόπτη. Η βασική ιδιότητα του επόπτη εξασφαλίζει τον αμοιβαίο αποκλεισμό των διεργασιών χωρίς την ανάγκη ορισμού κρισίμων τμημάτων. Ο επόπτης υλοποιεί επίσης ουρές αναμονής συνδεδεμένες με τις συνθήκες στο εσωτερικό του [9].

Εάν κάποιο από τα αντικείμενα, που προηγούμενα προστατευόταν από κάποιον από αυτούς τους επόπτες, βρεθεί σε ανακόλουθη κατάσταση, τα κατεστραμμένα αντικείμενα γίνονται ορατά σε άλλα νήματα, με ενδεχόμενο να καταλήξουν σε αυθαίρετη συμπεριφορά [10].

Έτσι λοιπόν η διακοπή ενός Thread πρέπει να υλοποιηθεί με διαφορετικό τρόπο. Παρακάτω παρουσιάζουμε τους δύο που χρησιμοποιήθηκαν.

- *Εάν το νήμα είναι επέκταση (extends) της Thread.*
  Τότε ορίζουμε μια μεταβλητή **volatile boolean m_chk** (volatile για να έχουμε synchronized προσπέλαση στη μεταβλητή) και χρησιμοποιούμε την τιμή της (την οποία ελέγχουμε) μέσα στη μέθοδο run σαν κριτήριο εξόδου μιας while loop δηλαδή

```
void run()
{
        while(m_chk==true)
        {
        //εντολές που θέλουμε να εκτελεί το νήμα
        }
}
```

  Δημιουργούμε μια άλλη μέθοδο **public void requestStop** που μεταβάλει την τιμή της volatile μεταβλητής

```
public void requestStop()
{
        m_chk=false;
}
```

  Οπότε για να σταματήσει ένα νήμα αρκεί να καλέσουμε την **requestStop** από το τρεχούμενο νήμα.

- *Εάν το νήμα υλοποιεί (implements) την διεπαφή Runnable.*
  Τότε ορίζουμε μεταβλητή **Thread runner** την οποία αρχικοποιούμε όταν θέλουμε να ξεκινήσουμε το νήμα

```
runner = new Thread(this);
runner.start();
```

  Δηλαδή περνάμε στο runner τον ίδιο του τον εαυτό και το εκκινούμε.
  Η μέθοδος **run** έχει τώρα τη μορφή

```
void run()
{
        Thread thisTread = Thread.currentThread();
        while(runner==thisTread)
        {
                //εδώ εκτελούνται οι εντολές του νήματος
```

```
        }
    }
```

Σημασία έχει η πρώτη γραμμή της μεθόδου όπου δημιουργείται το αντικείμενο thisThread, το οποίο αρχικοποιείται με αναφορά στο τρέχον νήμα. Η while loop εκτελείται όσο η runner ταυτίζεται με το νήμα αυτό.

Για να σταματήσουμε το νήμα καλούμε μια μέθοδο **public void requestStop()** η οποία καλείται από το κύριο νήμα και έχει τη μορφή:

```
void stopClassifing()
  {
    runner = null;
  }
```

6.7.2 Streams

Ο τρόπος με τον οποίο οι εφαρμογές που έχουν γραφτεί στην Java, εισάγουν και εξάγουν δεδομένα, επικοινωνώντας μεταξύ τους με το σύστημα αρχείων (file system) ή με κάποια συσκευή (device) γίνεται με την βοήθεια των Streams. Ένα stream δηλαδή, δεν είναι τίποτε άλλο από ένα γενικό κανάλι στο οποίο υπάρχει ροή δεδομένων.

Η Java διαθέτει αρκετά είδη από streams καθένα από τα οποία χρησιμοποιείται για διαφορετικά είδη δεδομένων. Τα προκαθορισμένα stream εισόδου/ εξόδου InputStream και OutputStream αποτελούν τη βάση για όλα τα υπόλοιπα. Ίσως τα πιο συχνά χρησιμοποιούμενα streams είναι τα DataInputStream και DataOutputStream. Το μεν πρώτο δίνει την δυνατότητα, της ανάγνωσης πρωτογενών τύπων δεδομένων της Java ανεξάρτητα από το είδος της μηχανής (υπολογιστής και λειτουργικό σύστημα) πάνω στο οποίο εκτελείται η εφαρμογή. Αντίστοιχα το δεύτερο χρησιμοποιείται για την εγγραφή πρωτογενών τύπων της Java. Πιο αυστηρά, τα DataInputStream και DataInputStream αναπαριστούν Unicode αλφαριθμητικά (string) σε μορφή που είναι μια ελαφρά τροποποίηση της UTF-8.

Ως άλλα είδη από streams, θα μπορούσαμε να αναφέρουμε τους buffers αρχείων οι οποίοι συχνά χρησιμοποιούνται για να αυξήσουν την ταχύτητα και απόδοση του συστήματος, στην περίπτωση που λαμβάνει χώρα I/O (είσοδος/έξοδος). Τα BufferedInputStream και BufferedOutputStream διαβάζουν και γράφουν τμήματα δεδομένων (το μέγεθος των οποίων μπορεί να καθοριστεί). Όταν γίνεται ανάγνωση από ή εγγραφή σε buffered stream, ο χρήστης στην ουσία ασχολείται με τον buffer και όχι με τα πραγματικά δεδομένα του stream. Θα πρέπει όμως σε τακτά χρονικά διαστήματα να καλείται η μέθοδος flush των buffers, έτσι ώστε να είμαστε σίγουροι ότι όλα τα δεδομένα στον buffer έχουν διαβαστεί από ή γραφτεί στο σύστημα αρχείων (file system). Ακόμη σε περίπτωση που θέλουμε να χειριστούμε απευθείας αρχεία από το τοπικό file system, τα FileInputStream και FileOutputStream χρησιμοποιούνται για το άνοιγμα, την ανάγνωση και την εγγραφή αρχείων [2].

6.7.3 Exceptions

Όταν σε ένα πρόγραμμα Java συμβεί κάποιο λάθος τότε ο κώδικας που θα το ανιχνεύσει μπορεί να εγείρει (throw = πετάω, εγείρω) μία εξαίρεση. Η έγερση εξαίρεσης θα έχει ως αποτέλεσμα τον τερματισμό του νήματος στο οποίο συνέβη το εν λόγω σφάλμα. Ωστόσο τα προγράμματα μπορούν να ορίσουν χειριστές εξαιρέσεων και να φροντίζουν ώστε το πρόγραμμα να ανανήψει από το λάθος. Μερικές από τις εξαιρέσεις προκαλούνται από το runtime system, όπως στην περίπτωση διαίρεσης με το μηδέν.

Για να ορίσουμε ένα χειριστή εξαιρέσεων (exception handler) θα πρέπει να κλείσουμε τον κώδικα που μπορεί να προκαλέσει την εξαίρεση μέσα σε μια εντολή try. Μετά την try θα πρέπει να βάλουμε μία ή περισσότερες εντολές catch. Κάθε catch θα μπορεί να πιάνει μία μόνο κλάση εξαίρεσης. Επίσης σε κάθε catch θα υπάρχει ο κατάλληλος κώδικας για τον χειρισμό της εξαίρεσης. Η πρώτη εντολή catch με παράμετρο που ταιριάζει στην εξαίρεση θα εκτελεστεί. Έπειτα το πρόγραμμα θα συνεχίσει μετά τις εντολές try/catch. Ωστόσο δεν είναι δυνατή η εκτέλεση του προγράμματος να συνεχιστεί από το σημείο όπου συνέβη η εξαίρεση.

Υπάρχουν περιπτώσεις όπου θέλουμε ένα κομμάτι κώδικα να εκτελείται πάντα ανεξάρτητα από το αν συμβεί κάποια εξαίρεση ή όχι. Αυτό επιτυγχάνεται με την εντολή finally. Η finally εκτελείται ακόμα και όταν το try block περιέχει τις εντολές return, break, continue ή throw.

Οι χειριστές εξαιρέσεων μπορούν να φωλιαστούν επιτρέποντας έτσι ο χειρισμός να γίνει σε περισσότερα του ενός σημεία. Αυτό είναι χρήσιμο όταν ο πρώτος exception handler δεν μπορεί να διορθώσει πλήρως το λάθος. Προκειμένου να περαστεί ο χειρισμός της εξαίρεσης στον επόμενο (υψηλότερου επιπέδου) exception handler θα πρέπει να χρησιμοποιήσουμε την throw και για αντικείμενο εξαίρεσης να δώσουμε την εξαίρεση που πιάσαμε. Σημειώνεται ότι έπειτα από το throw η εκτέλεση του τρέχοντος κώδικα χειρισμού διακόπτεται οριστικά [2].

6.7.4 Sockets

6.7.4.1 Το Μοντέλο Client-Server και Βασικοί Ορισμοί

Το ευρύτερα διαδεδομένο μοντέλο ανάπτυξης δικτυακών εφαρμογών, είναι το μοντέλο του πελάτη-εξυπηρετητή (client-server). Ο εξυπηρετητής είναι μια διεργασία, η οποία εκτελείται σε έναν υπολογιστή και αναμένει να συνδεθεί σε αυτήν κάποιο πρόγραμμα (το οποίο καλείται πελάτης) για να του παράσχει υπηρεσίες. Ένα τυπικό σενάριο που ακολουθείται συνήθως είναι το εξής:

- Η διεργασία-εξυπηρετητής αρχίζει να εκτελείται σε κάποιον υπολογιστή. Μετά την αρχικοποίηση της, πέφτει σε λήθαργο, αναμένοντας μία διεργασία-πελάτη να επικοινωνήσει μαζί της και να ζητήσει κάποια υπηρεσία.
- Μία διεργασία-πελάτης αρχίζει να εκτελείται είτε στο ίδιο σύστημα είτε σε κάποιο απομακρυσμένο, το οποίο συνδέεται με τον υπολογιστή στον οποίο εκτελείται ο εξυπηρετητής μέσω δικτύου. Η διεργασία πελάτης στέλνει μια αίτηση μέσω του

δικτύου, στον εξυπηρετητή, ζητώντας του κάποιου είδους υπηρεσία (για παράδειγμα μεταφορά αρχείου, απομακρυσμένη εκτύπωση, ανάγνωση και αποστολή mail).

- Ταυτόχρονα με την εξυπηρέτηση κάποιου πελάτη ο server έχει την δυνατότητα να δέχεται και αιτήσεις άλλων πελατών προς εξυπηρέτηση. Όταν ο εξυπηρετητής τελειώσει με όλους τους πελάτες, τότε ξαναπέφτει σε λήθαργο περιμένοντας για μια καινούργια αίτηση και η διαδικασία ξεκινά από την αρχή.

Ορίζουμε ως **σύνδεση**, τον επικοινωνιακό δίαυλο μεταξύ δύο διεργασιών. Την σύνδεση μπορούμε να την θεωρήσουμε ως μία πεντάδα που περιλαμβάνει τα εξής: {πρωτόκολλο, τοπική-διεύθυνση, τοπική-διεργασία, απομακρυσμένη-διεύθυνση, απομακρυσμένη-διεργασία}.

Το πρωτόκολλο αναφέρεται στο σύνολο των κανόνων που διέπουν την επικοινωνία. Η τοπική-διεύθυνση και απομακρυσμένη-διεύθυνση, προσδιορίζουν την ταυτότητα των υπο-δικτύων και των υπολογιστών, στους οποίους εκτελούνται οι επικοινωνούσες διεργασίες. Η τοπική-διεργασία και η απομακρυσμένη-διεργασία προσδιορίζουν την ταυτότητα των διεργασιών που θα επικοινωνούν, καθώς σε έναν υπολογιστή, μπορούν να εκτελούνται περισσότερες της μιας διεργασίες. Κάθε μία από αυτές τις διεργασίες που εκτελούνται στον ίδιο host και που χρειάζονται επικοινωνία μέσω δικτύου, λαμβάνει ένα 16-bit θετικό ακέραιο αριθμό, ο οποίος αναπαριστά την θύρα (port number) της διεργασίας και κατ' επέκταση της υπηρεσίας.

Σημειώνεται ότι οι αριθμοί θυρών από 1 έως 1023 είναι δεσμευμένοι για υπηρεσίες του συστήματος (κυρίως εξυπηρετητές συστήματος) και συνήθως χρησιμοποιούνται οι αριθμοί πάνω από 5000 για τα προγράμματα των χρηστών (εξυπηρετητές και άλλες εφαρμογές χρηστών). Μπορεί όμως να πάρει την τιμή 0 οπότε το σύστημα διαλέγει ένα ελεύθερο port (>1024) [11].

Ορίζουμε επίσης ως **μισή σύνδεση** (half association) είτε το σύνολο {πρωτόκολλο, τοπική-διεύθυνση, τοπική-διεργασία} είτε το σύνολο {πρωτόκολλο, απομακρυσμένη-διεύθυνση, απομακρυσμένη-διεργασία}. *Η μισή σύνδεση ονομάζεται αλλιώς και* ***socket****.*

Η έκδοση 4.1cBSD του Unix (1982) για τους υπολογιστές VAX από το πανεπιστήμιο του Berkeley εισήγαγε το socket interface σαν μια μέθοδο επικοινωνίας απομακρυσμένων διεργασιών. Είχαν αρχικά υλοποιηθεί στη γλώσσα C του Unix, αλλά η απήχηση που γνώρισαν, επέβαλε την μεταφορά τους τόσο σε άλλα λειτουργικά συστήματα (για παράδειγμα τα Microsoft Windows) όσο και σε άλλες γλώσσες προγραμματισμού (π.χ. Java).

Κάθε πρόγραμμα διαβάζει από και γράφει σε ένα socket με τρόπο παρόμοιο της εγγραφής και ανάγνωσης αρχείων του file system. Υπάρχουν δύο είδη socket:

- Το πρώτο ονομάζεται TCP (Transmission Control Protocol) socket και είναι μια υπηρεσία προσανατολισμένη στην σύνδεση (connection-oriented service). Μπορούμε να το θεωρήσουμε ανάλογο της τηλεφωνικής υπηρεσίας, στην οποία, μετά την εγκαθίδρυση μιας σύνδεσης μεταξύ δύο συνομιλητών, αυτή χρησιμοποιείται μέχρι το πέρας της συζητήσεως τους.
- Το άλλο είδος ονομάζεται UDP (Unreliable Datagram Protocol) socket και είναι μια υπηρεσία χωρίς σύνδεση (connectionless service). Το ανάλογο, σε αυτήν την περίπτωση, είναι το ταχυδρομείο: μπορούμε να στείλουμε πολλά

πακέτα στον ίδιο παραλήπτη αλλά δεν είναι σίγουρο ότι όλα θα ακολουθήσουν την ίδια διαδρομή (νοητή σύνδεση) για να φτάσουν στον προορισμό τους.

Μια ακόμη σημαντική διαφορά, μεταξύ των παραπάνω δύο ειδών, είναι ότι τα TCP sockets εξασφαλίζουν μια αξιόπιστη μεταφορά της πληροφορίας: ότι αποστέλλεται από το ένα άκρο είναι σίγουρο ότι θα φτάσει στο άλλο. Στο UDP socket όμως δεν συμβαίνει αυτό. Είναι στην ευθύνη του αποστολέα να ελέγξει ότι αυτό που έστειλε, το έλαβε τελικά ο παραλήπτης και δεν χάθηκε στον δρόμο. Από την άλλη, η σύνδεση με το TCP socket απαιτεί την ανταλλαγή τριών "πακέτων χειραψίας" (handshake packets) και είναι πιο χρονοβόρα στην αρχικοποίηση της από την αντίστοιχη με UDP datagrams. Οι προηγούμενες δύο διαφορές καθορίζουν τελικά και την χρήση των δύο αυτών ειδών.

Για την αποφυγή σύγχυσης, να σημειώσουμε ότι ειδικά στην Java, ο όρος Socket χρησιμοποιείται για τα TCP sockets στην ονοματολογία των κλάσεων και των μεθόδων, ενώ για την δήλωση των UDP sockets χρησιμοποιείται ο όρος Datagram.

Στον παρακάτω πίνακα βλέπουμε τις διάφορες κλάσεις για τα sockets που περιέχονται στο πακέτο java.net καθώς και το είδος τους.

Μηχανισμός / Κλάση	**Περιγραφή**
Socket	TCP άκρο-πελάτης
ServerSocket	TCP άκρο-εξυπηρετητής
DatagramSocket	UDP άκρο (client& server)
DatagramPacket	UDP πακέτο
InetAdress	Διεύθυνση Internet Protocol (IP)
URL	Uniform Resource Locator
URLConnection	Σύνδεση με αντικείμενο του web

Πίνακας 6.7.4.1.1 Κλάσεις που περιέχονται στο java.net για υλοποίηση socket

6.7.4.2 Ανάπτυξη της Πλευράς του Πελάτη (Client) με TCP Sockets

Μια τυπική αλληλουχία βημάτων που συνήθως ακολουθούνται για το πρόγραμμα του Client είναι η εξής:

socket() - αρχικοποίηση και σύνδεση στον **server**
getInputStream() - σύνδεση του **socket** με τον προκαθορισμένο
και **getOutputStream**() μηχανισμό εισόδου/εξόδου στη Java, δηλαδή τα **streams**.

read() και **write**() - ανάγνωση και εγγραφή στο συνδεδεμένο με το **socket stream**
close() - κλείνει το **socket** και απελευθερώνονται πόροι του συστήματος

Η δημιουργία ενός αντικειμένου της κλάσης **socket** γίνεται πολύ απλά καλώντας έναν από τους κατασκευαστές της κλάσης

Socket s= new Socket(ip_adress,8000);

δηλαδή φτιάξε το αντικείμενο s που είναι **socket** και σύνδεσε το στην **IP** διεύθυνση που καθορίζεται από την μεταβλητή ip_adress, στην θύρα 8000.

Εφόσον η σύνδεση είναι επιτυχής, μπορούμε να λάβουμε τα **streams** εισόδου και εξόδου του socket εισόδου και εξόδου που δημιουργήσαμε για να επιτελέσουμε στην συνέχεια **I/O**. Αυτό γίνεται γιατί ο ενοποιημένος (και μοναδικός) μηχανισμός για **I/O** προγραμμάτων σε Java είναι τα **streams**. Για τον σκοπό αυτό χρησιμοποιούμε τις μεθόδους **getInputStream**() και **getOutputStream**(0 που διαθέτει η κλάση **socket** ως εξής:

DataInputStream sin=new DataInputStream(s.getInputStream());
DataOutputStream sout=new DataOutputStream(s.getOutputStream());

Μετά την αρχικοποίηση και την λήψη των **stream** του **socket** έχουμε είσοδο/έξοδο με τις καθορισμένες μεθόδους.

Όταν τελειώσουμε με το **I/O** πρέπει να απελευθερώνουμε το **socket** που είχαμε ανοίξει γιατί είναι πολύτιμος πόρος του συστήματος και μπορεί να εξαντληθεί.

Σημειώνεται ότι οι εντολές που συσχετίζονται με αρχικοποίηση του socket και των stream αυτού, θα πρέπει να περιέχονται στο σώμα **try** εντολής κι αυτό γιατί υπάρχει περίπτωση να μην είναι δυνατή η σύνδεση οπότε θα έχουμε και έγερση μιας εξαίρεσης **IOException**. Επίσης η μέθοδος **close**() είναι θεμιτό να καλείται σε κάθε περίπτωση για αυτό και την καλούμε μέσα από **finally**.

6.7.4.3 Ανάπτυξη της Πλευράς του Εξυπηρετητή (Server) με TCP Sockets

Μια τυπική TCP εφαρμογή εξυπηρετητή ανοίγει ένα port για την λήψη αιτήσεων για σύνδεση και ύστερα δημιουργεί μία διεργασία-παιδί, ή ένα ξεχωριστό νήμα εκτέλεσης για να εκτελέσει την ζητούμενη υπηρεσία. Το port που ανοίγει ο server ονομάζεται well known γιατί αυτό χρησιμοποιεί ο οποιοσδήποτε πελάτης για να συνδεθεί στον εξυπηρετητή. Επίσης λέμε ότι ο server ακούει το port για καινούργιους πελάτες. Για το λόγο αυτό, το socket ονομάζεται listening socket. Θα πρέπει να δοθεί προσοχή στην επιλογή του αριθμού θύρας της υπηρεσίας, ο οποίος δεν θα πρέπει να είναι ήδη σε χρήση.

Ο εξυπηρετητής μόλις δεχθεί την σύνδεση καινούργιου πελάτη, γεννά μια καινούργια διεργασία-νήμα για την εξυπηρέτηση των αιτήσεων αυτού. Με αυτόν τον τρόπο, καθίσταται δυνατή η παράλληλη εξυπηρέτηση παλιών και η αποδοχή νέων πελατών.

Μια τυπική αλληλουχία βημάτων που συνήθως ακολουθούνται για το πρόγραμμα του Server είναι η εξής:

ServerSocket()	- αρχικοποίηση του listening socket
accept()	- αναμονή και εντοπισμός καινούργιου πελάτη
new Thread	- δημιουργία καινούργιου thread για την εξυπηρέτηση πελάτη
getInputStream() και **getOutputStream**()	- το καινούργιο socket που επιστρέφει από την accept συνδέεται με τον προκαθορισμένο μηχανισμό εισόδου/εξόδου της Java
read()και **write**()	- ανάγνωση και εγγραφή στο stream
close()	- κλείνει το socket και απελευθερώνει τον αντίστοιχο, πόρο του συστήματος

Η κλάση ServerSocket περιέχεται στο πακέτο java.net και παρέχει μια υλοποίηση σύνδεσης TCP socket ανεξάρτητη συστήματος, από την πλευρά του εξυπηρετητή, με βάση το client/server μοντέλο. Το αντικείμενο αυτής της κλάσης συνδέει το πρόγραμμα του εξυπηρετητή με κάποια θύρα του συστήματος, για να μπορεί αυτός στην συνέχεια να ελέγχει για την σύνδεση νέων πελατών. Στον κατασκευαστή του αντικειμένου προσδιορίζεται ο αριθμός θύρας και σε περίπτωση που δεν χρησιμοποιείται ήδη από άλλη υπηρεσία, δημιουργείται το αντικείμενο. Αν όμως η θύρα είναι κατειλημμένη, τότε εγείρεται εξαίρεση. Ο κώδικας είναι ο εξής:

```
ServerSocket servesoc=new SesrverSocket(8000);
```

Αν το αντικείμενο κατασκευαστεί επιτυχώς, τότε ο server είναι έτοιμος για την υποδοχή καινούργιου πελάτη:

```
Socket incoming =serversoc.accept();
```

Η μέθοδος **accept**() μπλοκάρει το νήμα από το οποίο έχει κληθεί, μέχρι να παρατηρηθεί δραστηριότητα στο listening-socket δηλαδή μέχρι να συνδεθεί καινούργιος πελάτης. Όταν συμβεί το τελευταίο, τότε η accept() επιστρέφει ένα καινούργιο Socket (όχι listening-socket) με το οποίο αρχίζει πλέον η επικοινωνία. Αυτό συμβαίνει για να μπορούμε να συνεχίσουμε να ακούμε από το listening-socket και άλλους πελάτες, χωρίς να χρειάζεται να αλλάξουμε τον αριθμό της θύρας. Επίσης δημιουργείται ένα νέο νήμα και αρχίζει η εκτέλεση του. Στην κατασκευή του νέου νήματος παίρνουμε σαν παράμετρο το socket που επιστρέψει η accept(), για να λάβουμε ύστερα τα συνδεδεμένα με αυτό, stream εισόδου/εξόδου.

```
DataInputStream sin=new DataInputStream(s.getInputStream());
DataOutputStream sout=new DataOutputStream(s.getOutputStream());
```

Από εκεί και πέρα, ακολουθεί I/O στο socket (μέσω των stream του) σαν να είχαμε I/O σε αρχείο. Στο τέλος το thread εξυπηρέτησης του πελάτη κλείνει το socket που μας είχε επιστρέψει η accept() (*με τη μέθοδο close()*) και τερματίζει την εκτέλεση του. Το listening socket παραμένει ανοικτό, απελευθερώνοντας όμως πριν τον τερματισμό του, το ίδιο το νήμα. Σημειώνεται επίσης η απαραίτητη χρήση των μπλοκ try για να αποφεύγουμε την περίπτωση της χρησιμοποίησης κάποιου πόρου (στην περίπτωση μας socket), ο οποίος πιθανόν δεν μας έχει διατεθεί από το σύστημα [2].

Κεφάλαιο 7

Το Σύστημα WiDIFuNeCS

7 Το Σύστημα WiDIFuNeCS

7.1 Γενικός Σχεδιασμός

Το αντικείμενο της παρούσας διπλωματικής εργασίας όπως πιθανότατα έχει διαφανεί από τα προηγούμενα κεφάλαια εστιάζεται στα νευρο-ασαφή συστήματα ταξινόμησης. Συνακόλουθα χρησιμοποιώντας ένα συγκεκριμένο νευρο-ασαφές σύστημα το SuPFuNIS, επιχειρείται μια ασύρματη κατανεμημένη λειτουργία αυτού σε πραγματικό περιβάλλον. Με τον όρο κατανεμημένη λειτουργία περιγράφεται μια λειτουργία κατά την οποία τα πρότυπα εισόδου προέρχονται από διαφορετικές συσκευές από αυτήν που εκτελείται ο ταξινομητής. Συνακόλουθα η επιβλεπόμενη εκπαίδευση και η υβριδική επανεκπαίδευση του SuPFuNIS υλοποιούνται επίσης σε διαφορετική συσκευή από αυτήν του ταξινομητή. Με τον όρο ασύρματη, περιγράφεται η λειτουργία κατά την οποία τα δεδομένα εισόδου συλλέγονται με ασύρματο τρόπο ενώ και η αρχικοποίηση / ανανέωση του διανύσματος των βαρών γίνονται επίσης ασύρματα. Τέλος με τον όρο πραγματικό περιβάλλον περιγράφεται μια λειτουργία σε πραγματικές ασύρματες συνθήκες (π.χ. παρεμβολές, απώλειες, θόρυβος) και σε πραγματικό χρόνο αφού η συλλογή των εισόδων και η εξαγωγή της κατηγορίας γίνονται real time.

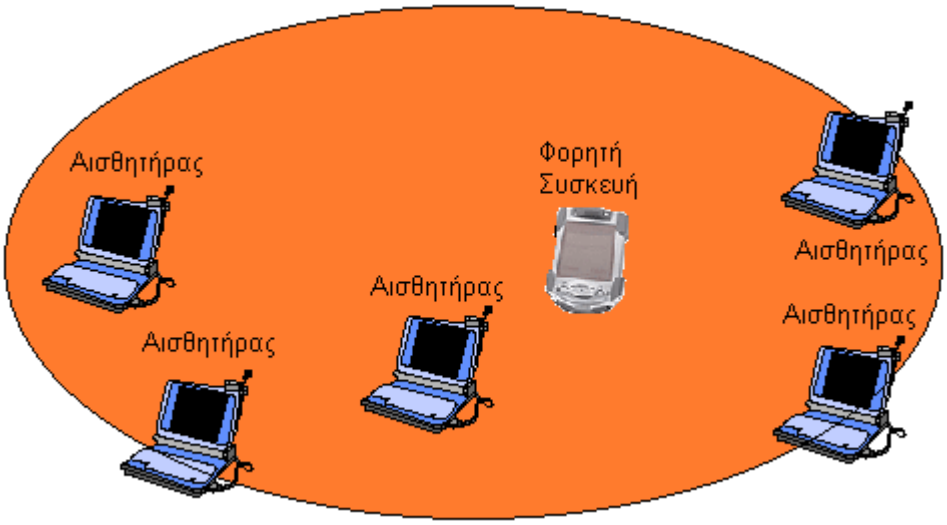
Το πραγματικό πρόβλημα το οποίο αποτέλεσε την πηγή έμπνευσης της προηγούμενης αρχιτεκτονικής και αυτό πάνω στο οποίο έγινε προσπάθεια να αναπτυχθεί η διπλωματική εργασία (τουλάχιστον σε επίπεδο υλοποίησης) είναι το εξής: *Να δημιουργηθεί μία φορητή συσκευή η οποία θα έχει την δυνατότητα να λαμβάνει πληροφορίες από έναν ασθενή με αισθητήρες που θα φέρει επάνω του και θα μπορεί να βγάλει ιατρικής φύσεως συμπεράσματα για την κατάσταση του. Παράλληλα θα πρέπει να υπάρχει η δυνατότητα ανανέωσης-επέκτασης των δυνατοτήτων εξαγωγής συμπερασμάτων από την πλευρά της συσκευής.* Συνολικά τα στοιχεία-πεδία που συνιστούν το περιβάλλον στο οποίο κινείται η υλοποίηση του WiDIFuNeCS είναι τα κάτωθι:

1. **ο αλγόριθμος εξαγωγής συμπερασμάτων και η διαδικασία βελτίωσης-επέκτασης των δυνατοτήτων αυτού**
2. **οι ασύρματες διασυνδέσεις του συνόλου των επιμέρους συσκευών**
3. **οι λειτουργίες των αισθητήρων**
4. **οι λειτουργίες της φορητής συσκευής**
5. **οι λειτουργίες της συσκευής εξυπηρετητή**

<u>Επιμέρους Στοιχεία του WiDIFuNeCS:</u>

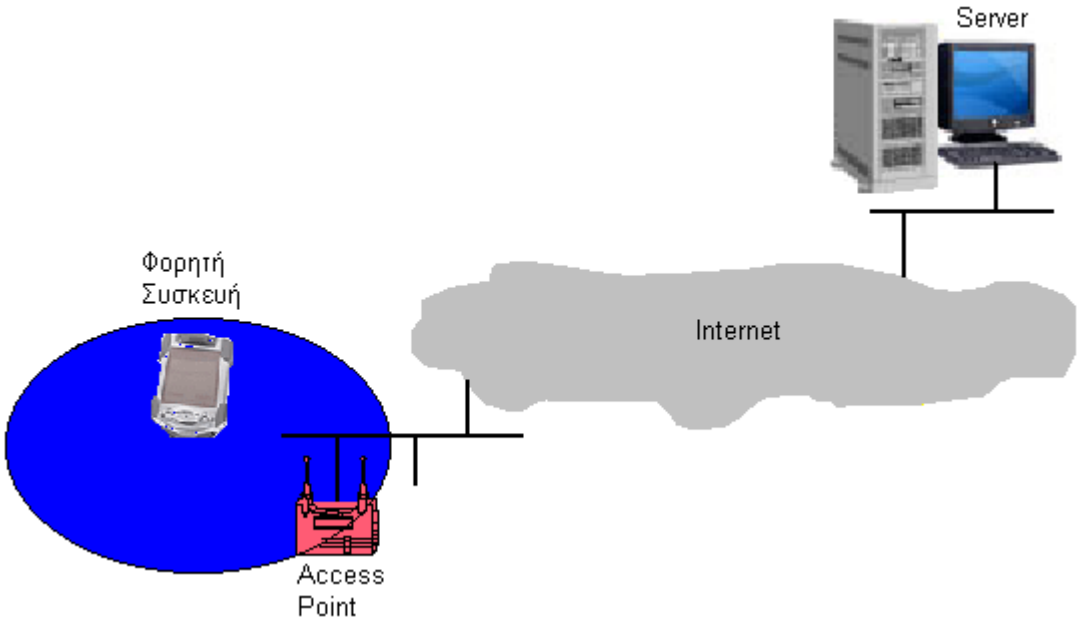
1] Η εξαγωγή συμπερασμάτων γίνεται με την χρήση του νευρο-ασαφούς συστήματος SuPFuNIS [§ 2.3], [§ 4]. Το SuPFuNIS χρησιμοποιείται ως ταξινομητής, δηλαδή ως σύστημα εξαγωγής συμπερασμάτων σχετικών με την κατηγορία στην οποία ανήκει το εκάστοτε πρότυπο εισόδου [§ 3.1]. Υπάρχει επιπλέον η δυνατότητα κατά την εξαγωγή του συμπερασμάτων να χρησιμοποιηθεί πολυεπίπεδος ταξινομητής SuPFuNIS [§ 7.2]. Για την αρχικοποίηση του ταξινομητή, ώστε να είναι σε θέση να κατηγοριοποιεί πρότυπα που ανήκουν σε καινούργια προβλήματα, χρησιμοποιείται ο αλγόριθμος backpropagation με την μέθοδο κλίσης [§ 3.2] όπως αυτός ορίζεται για το SuPFuNIS [§ 4]. Ακόμη για την επανεκπαίδευση-βελτίωση της ικανότητας ταξινόμησης του συστήματος εξαγωγής συμπερασμάτων χρησιμοποιείται ένας πρότυπος υβριδικός αλγόριθμος μάθησης [§ 7.3].

2] Η συλλογή προτύπων εισόδου από τους αισθητήρες για λογαριασμό της φορητής συσκευής γίνεται ασύρματα κάνοντας χρήση ad hoc τοπολογίας, όπως αυτή περιγράφεται στο πρότυπο IEEE 802.11 [§ 5.2]. Η επικοινωνία της φορητής συσκευής με τον server ούτως ώστε να γίνει δυνατή η εκπαίδευση-επανεκπαίδευση του συστήματος ταξινόμησης γίνεται ασύρματα με χρήση του προτύπου IEEE 802.11 για WLAN. Σε αυτή την περίπτωση όμως η τοπολογία διασύνδεσης των συσκευών μπορεί να είναι ad hoc τοπολογία ή τοπολογία δικτύου υποδομής.
3] Αισθητήρας καλείται μια μικρή συσκευή που έχει την δυνατότητα να λαμβάνει σήματα διαφόρων ειδών και να παράγει αριθμητικές τιμές σχετικές με τα σήματα αυτά. Ένα ζήτημα που προκύπτει είναι η ευφυΐα των αισθητήρων, δηλαδή το ποιοτικό και ποσοτικό επίπεδο των πράξεων που μπορεί να πραγματοποιεί ο αισθητήρας τόσο στα δεδομένα εισόδου όσο και στα δεδομένα εξόδου. Η επιλογή του να εξομοιωθούν οι αισθητήρες από κατάλληλα προγράμματα ήταν αναγκαστική από την στιγμή που δεν υπήρχαν πραγματικές υλοποιήσεις τους. Εντούτοις δόθηκε ιδιαίτερη μέριμνα στο να προσομοιώνουν κατά το μέγιστο δυνατόν την λειτουργία θεωρητικά υλοποιήσιμων αισθητήρων. Έτσι η πολυπλοκότητα τους περιορίστηκε στο ελάχιστο ενώ οι μόνες παράμετροι που μπορούν να ρυθμιστούν είναι αυτές που αφορούν στην ασύρματη επικοινωνία καθώς και στην επιθυμητή λειτουργικότητα αναφορικά με την εξομοίωση. Έτσι επιλέγεται οι αισθητήρες απλά να στέλνουν ανά τυχαία καθορισμένα χρονικά διαστήματα τυχαίες τιμές από μια συγκεκριμένη περιοχή τιμών.
4] Η φορητή συσκευή αποτελεί τον κρίκο που προσδίδει συνεκτικότητα στην υλοποίηση WiDIFuNeCS. Περιλαμβάνει μια γραφική διεπαφή που επιτρέπει στον χρήστη να επιλέξει μεταξύ των λειτουργιών της ταξινόμησης και της μάθησης του WiDIFuNeCS. Κατά την διάρκεια της λειτουργίας της ταξινόμησης η φορητή συσκευή εξάγει συμπεράσματα σε πραγματικό χρόνο ενώ παράλληλα συλλέγει διανύσματα εισόδου με σκοπό να τα χρησιμοποιήσει στην φάση της επανεκπαίδευσης του συστήματος. Κατά την διάρκεια της λειτουργίας μάθησης παρέχονται δυο δυνατότητες. Η πρώτη είναι να ζητηθεί από τον server ένα σύνολο διανυσμάτων βαρών που να αντιστοιχούν σε ένα εντελώς καινούργιο πρόβλημα ταξινόμησης (αρχικοποίηση). Ενώ η δεύτερη είναι να αποσταλούν τα διανύσματα εισόδων, που έχουν συλλεγεί, στον server (για περαιτέρω επεξεργασία) και να ανακτηθεί το μέχρι εκείνη την στιγμή βέλτιστο σύνολο βαρών που διαθέτει ο server επί του συγκεκριμένου προβλήματος ταξινόμησης (ανανέωση).
5] Το πρόγραμμα της συσκευής εξυπηρετητή είναι τέτοιο που από την στιγμή που καθοριστεί σε αυτό ένα πρόβλημα ταξινόμησης δεν χρειάζεται να επέμβει εκ νέου ένας διαχειριστής (administrator) για να καθορίσει-τροποποιήσει παραμέτρους ή επιτελούμενες λειτουργίες. Κατά την φάση της εκκίνησης στον εξυπηρετητή ανατίθεται ένα δεδομένο πρόβλημα ταξινόμησης και στην συνέχεια με παραμετροποιημένη επιβλεπόμενη μάθηση καθορίζονται ορισμένα αρχικά σύνολα βαρών για τον πολυεπίπεδο ταξινομητή. Κατά την φάση της κανονικής λειτουργίας ο server δέχεται και επεξεργάζεται αιτήσεις από φορητές συσκευές. Εάν η ληφθείσα αίτηση είναι αίτημα αρχικοποίησης τότε αποστέλλονται στην συσκευή πλήρη διανύσματα βαρών καθώς και ορισμένες πληροφορίες περί του νέου προβλήματος ταξινόμησης που συνδέεται με αυτά τα βάρη. Εάν η αίτηση είναι αίτημα επανεκπαίδευσης τότε αποστέλλονται στην φορητή συσκευή τα μέχρι εκείνη την στιγμή βέλτιστα διανύσματα βαρών. Παράλληλα αφού συλλεχθούν τα πρότυπα εισόδου που έχουν σταλεί από την φορητή συσκευή εκκινείται αυτόματα μια υβριδική διαδικασία επανεκπαίδευσης.



Σχήμα 7.1.1: Αισθητήρες → φορητή συσκευή (ad hoc τοπολογία).

Σχήμα 7.1.2: Φορητή συσκευή ↔ server (ad hoc τοπολογία).



Σχήμα 7.1.3: Φορητή συσκευή ↔ server (τοπολογία δικτύου υποδομής).

7.2 Πολυεπίπεδος Ταξινομητής

Ένα από τα βασικά χαρακτηριστικά που καθορίζει καταλυτικά την συμπεριφορά του SuPFuNIS ως ταξινομητή είναι το πλήθος των κανόνων που χρησιμοποιεί (δηλαδή κατά μια άλλη προσέγγιση ο αριθμός των νευρώνων στο κρυμμένο στρώμα). Αυξάνοντας αυθαίρετα τον αριθμό των κανόνων είναι δυνατόν να επιτευχθεί ορθή ταξινόμηση όλων των προτύπων ακόμη και εάν το σύνολο εκπαίδευσης είναι πολύπλοκο (π.χ. στο υπερπεπίπεδο των εισόδων τα χαρακτηριστικά δεν είναι γραμμικά διαχωρίσιμα). Μια τέτοια επιλογή όμως έχει ως άμεσο αποτέλεσμα την εμφάνιση του φαινομένου της υπερεκπαίδευσης. Σε αυτή την περίπτωση ο ταξινομητής ουσιαστικά έχει πολύ μεγάλο αριθμό παραμέτρων και έχει ουσιαστικά απομνημονεύσει τα πρότυπα εισόδου ελαττώνοντας την γενικευτική του ικανότητα.

Ένας αποτελεσματικός τρόπος εκτίμησης του βέλτιστου αριθμού κανόνων του SuPFuNIS είναι αυτός που περιγράφεται στο [§ 4.7]. Παράλληλα με αυτό τον αριθμό κανόνων μπορεί να χρησιμοποιηθεί ένα δεύτερο σύνολο κανόνων με περισσότερους κανόνες από τον βέλτιστο αριθμό. Κάτι τέτοιο είναι ιδιαίτερα αποτελεσματικό στην περίπτωση ύπαρξης ομάδων χαρακτηριστικών (συνήθως περιορισμένων σε πλήθος) τα οποία ο ταξινομητής αποτυγχάνει να ταξινομήσει επιτυχώς. Ο αριθμός των επιπλέον κανόνων μπορεί να αφιερωθεί στην ορθή ταξινόμηση αυτών των ομάδων και έτσι να αυξηθεί η απόδοση του ταξινομητή.

Μια τέτοια σχεδιαστική επιλογή όμως δεν μπορεί να αποτρέψει την εμφάνιση του φαινομένου της υπερεκπαίδευσης. Για να εξισορροπηθεί το αρνητικό αυτό αποτέλεσμα είναι θεμιτό να χρησιμοποιηθεί ένα τρίτο σύνολο κανόνων με πλήθος μικρότερο του βέλτιστου. Ο ταξινομητής με το εν λόγω πλήθος κανόνων ουσιαστικά είναι σε θέση να απομνημονεύσει την πληροφορία μόνο των βασικών ομάδων προτύπων, γεγονός που του παρέχει την δυνατότητα αυξημένης γενικευτικής ικανότητας συγκριτικά με τον ταξινομητή μεγάλου πλήθους κανόνων.

Συνδυάζοντας τους ταξινομητές με τους τρεις διαφορετικούς αριθμούς χρησιμοποιούμενων κανόνων είναι δυνατόν να κατασκευαστεί ένας πολυεπίπεδος ταξινομητής (συγκεκριμένα 3 επιπέδων). Το ζητούμενο όμως από έναν ταξινομητή οσαδήποτε επίπεδα και εάν διαθέτει είναι η ταξινόμηση κάθε δεδομένου προτύπου εισόδου σε μια κατηγορία και όχι σε τρεις εν γένει διαφορετικές κατηγορίες.

Μια αποτελεσματική και υπολογιστικά εύρωστη επιλογή είναι η χρησιμοποίηση μιας διαδικασίας ψηφοφορίας (voting procedure). Έτσι εφόσον τρεις ή δυο ταξινομητές συμφωνούν περί της κατηγορίας ενός προτύπου εισόδου τότε το συνολικό αποτέλεσμα του πολυεπίπεδου ταξινομητή είναι αυτή η κατηγορία. Στην περίπτωση που όλοι διαφωνούν μπορεί να ακολουθηθεί μια λογική της μορφής “εκεί που ναυαγεί ο κοινοβουλευτισμός κηρύσσεται δικτατορία”. Έτσι εφόσον και οι τρεις διαφωνούν μπορεί να χρησιμοποιηθεί ο τροποποιημένος εκτιμητής αξιοπιστίας [§ 3.3.3], ο ταξινομητής με την μεγαλύτερη τιμή αξιοπιστίας στο συμπέρασμα που εξάγει υπερισχύει των δυο άλλων. Στην περίπτωση αυτή η συνολική έξοδος του πολυεπίπεδου ταξινομητή ταυτίζεται με την έξοδο του ταξινομητή που υπερισχύει.

7.3 Υβριδική Μάθηση

Η ιδιότητα της προσαρμογής των νευρο-ασαφών συστημάτων σε μεταβαλλόμενους χώρους προβλημάτων σχετίζεται με την ικανότητα μάθησης τους. Η μάθηση είναι μια θεμελιακή ικανότητα των νευρο-ασαφών δικτύων η οποία τους παρέχει την ικανότητα να μαθαίνουν από το περιβάλλον τους και να βελτιώνουν τη συμπεριφορά τους με το πέρασμα του χρόνου. Ειδικότερα στα νευρο-ασαφή δίκτυα η μάθηση αναφέρεται στην διεργασία επίτευξης μιας επιθυμητής συμπεριφοράς μέσω ενημέρωσης των τιμών των συναπτικών βαρών. Έτσι ένα τέτοιο σύστημα μαθαίνει για το περιβάλλον του μέσω μιας επαναληπτικής διαδικασίας ανανέωσης των συναπτικών βαρών και κατωφλίων.

Ο αλγόριθμος εκπαίδευσης του SuPFuNIS [§ 3.2], [§ 4] αποτελεί ένα χαρακτηριστικό μοντέλο επιβλεπόμενης μάθησης [§ 2.2.5]. Το κύριο χαρακτηριστικό της επιβλεπόμενης μάθησης είναι η ύπαρξη ενός συνόλου εκπαίδευσης δηλαδή μιας βάσης γνώσης που περιέχει διανύσματα εισόδου καθώς και την κατηγορία που ανήκει κάθε ένα από αυτά. Όταν το νευρο-ασαφές δίκτυο λαμβάνει ένα διάνυσμα εισόδου εκπαίδευσης, παράλληλα λαμβάνει μια επιθυμητή έξοδο η οποία αναπαριστά την βέλτιστη δράση που πρέπει να εμφανίζει. Στην συνέχεια οι παράμετροι του SuPFuNIS ανανεώνονται βάσει του διανύσματος εκπαίδευσης και του διανύσματος σφάλματος (δηλαδή της διαφοράς πραγματικής και επιθυμητής απόκρισης του δικτύου).

Στην περίπτωση που η επιθυμητή έξοδος κάθε προτύπου εκπαίδευσης δεν είναι γνωστή ο παραπάνω αλγόριθμος δεν δύναται να λειτουργήσει. Η τροποποίηση που προτείνεται ως βασικό στόχο έχει την υπέρβαση αυτού του εμποδίου μεταβάλλοντας κατά τον μικρότερο δυνατό βαθμό τον υπόλοιπο αλγόριθμο μάθησης.

Σε πρώτη φάση λαμβάνει χώρα εφαρμογή του αλγορίθμου επιβλεπόμενης μάθησης (χωρίς καμία τροποποίηση) χρησιμοποιώντας ένα σύνολο εκπαίδευσης που περιέχει και τις επιθυμητές εξόδους.

Σε κάθε βήμα του προτεινόμενου αλγορίθμου μάθησης παρουσιάζεται ένα πρότυπο επανεκπαίδευσης στις εισόδους του SuPFuNIS, με αυτό τον τρόπο υπολογίζονται τόσο οι ενεργοποιήσεις z_j των κόμβων κανόνων όσο και τα σήματα εξόδου y_k. Στην συνέχεια ελλείψει επιθυμητής εξόδου εφαρμόζεται η μέθοδος winner takes all στα σήματα εξόδου. Ταυτόχρονα γίνεται η θεώρηση ότι η επιθυμητή κατηγορία ταυτίζεται με την κατηγορία που προκύπτει με την μέθοδο του νικητή, έτσι οι έξοδοι d_k καθορίζονται με χρήση της κωδικοποίησης 1 out of K.

Ακολούθως με δεδομένες τιμές για τα α και η, εκτελείται χωρίς καμία τροποποίηση ο αλγόριθμος backpropagation με την μέθοδο κλίσης εκτός από ένα συγκεκριμένο σημείο: τα κέντρα αλλά και οι διασπορές ενημερώνονται βάσει των σχέσεων (4.4.2) αλλά παράλληλα αποθηκεύονται οι τιμές κέντρων και διασπορών ακριβώς πριν από αυτή την ενημέρωση (ούτως ώστε να υπάρχει η δυνατότητα αναίρεσης αυτής της αλλαγής). Στην συνέχεια υπολογίζεται η ακρίβεια υποκατάστασης (resubstitution accuracy) στο αρχικό σύνολο εκπαίδευσης. Εφόσον η resubstitution accuracy παραμένει σταθερή ή αυξάνει τότε η προαναφερθείσα ανανέωση παραμέτρων διατηρείται και παρουσιάζεται ένα νέο πρότυπο επανεκπαίδευσης στις εισόδους του SuPFuNIS. Σε αντίθετη περίπτωση η ανανέωση των παραμέτρων αναιρείται οι τιμές των α και η μειώνονται κατά μια τάξη μεγέθους και ο αλγόριθμος εκτελείται με τον τρόπο που περιγράφεται στην παρούσα παράγραφο. Τέλος εάν αυτό επαναληφθεί έναν αριθμό φορών (για παράδειγμα 4 ή 5), δεν γίνεται καμία ανανέωση

των βαρών και παρουσιάζεται ένα νέο πρότυπο επανεκπαίδευσης στις εισόδους του SuPFuNIS.

Σε αυτό το σημείο αξίζει να αναφερθεί το γεγονός ότι πηγή έμπνευσης για αυτόν τον αλγόριθμο μάθησης αποτέλεσε η τεχνική bold driver (σχέση (3.2.2.6)). Η μείωση της συνάρτησης σφάλματος αντικαταστάθηκε από την ακρίβεια υποκατάστασης, παρέχοντας την δυνατότητα στα συναπτικά βάρη να μεταβληθούν κατά ένα υπολογίσιμο ποσοστό πράγμα που δεν θα ήταν δυνατό να γίνει εάν ενσωματωνόταν το κριτήριο ΔΕ. Ακόμη ανά πρότυπο (σε πρώτη φάση τουλάχιστον) χρησιμοποιούνται υψηλές τιμές για τα α και η, σε περίπτωση αποτυχίας όμως υπάρχει μείωση τους κατά μια τάξη μεγέθους έτσι ώστε να εξοικονομείται υπολογιστική ισχύς.

Η ιδέα που υπάρχει πίσω από τον προτεινόμενο αλγόριθμο υβριδικής μάθησης έχει δυο βασικούς άξονες. Ο πρώτος είναι ότι δεν πρέπει να επηρεαστεί η τιμή της ακρίβειας υποκατάστασης σε καμία περίπτωση. Ο δεύτερος είναι να παρέχεται η δυνατότητα τροποποίησης των βαρών βάσει της τοπικής εκτίμησης της κατηγορίας που ανήκει το κάθε ένα πρότυπο επανεκπαίδευσης. Σε κάποιο βήμα εάν η τιμή της resubstitution accuracy μειωθεί αυτό σημαίνει ότι η παραπάνω εκτίμηση πιθανότατα είναι λανθασμένη οπότε η μεταβολή του διανύσματος των βαρών αναιρείται και τα α και η μειώνονται με την προσδοκία ότι λιγότερο απότομες αλλαγές ενδεχόμενα να φέρουν το επιθυμητό αποτέλεσμα. Εάν η ακρίβεια υποκατάστασης παραμείνει σταθερή ή αυξηθεί τότε η ανανέωση των βαρών ενδεχόμενα να είναι προς την σωστή κατεύθυνση οπότε διατηρείται.

7.4 SuPFuNIS σε Java

Ο πυρήνας του συστήματος WiDIFuNeCS τουλάχιστον σε ότι σχετίζεται με τον αλγόριθμο εξαγωγής συμπερασμάτων καθώς και σε ότι σχετίζεται με την διαδικασία βελτίωσης-επέκτασης των δυνατοτήτων αυτού βασίζεται στο νευρο-ασαφές σύστημα SuPFuNIS. Το σύνολο του συστήματος υλοποιήθηκε σε γλώσσα προγραμματισμού Java [§ 6]. Σε γενικές γραμμές επιχειρήθηκε οι συναρτήσεις που σχετίζονταν με την λειτουργία του νευρο-ασαφούς συστήματος (είτε αυτή ήταν ταξινόμηση είτε εκπαίδευση-επανεκπαίδευση) να τοποθετηθούν σε διαφορετικές κλάσεις από ότι οι πηγαίοι κώδικες των συναρτήσεων του υπόλοιπου συστήματος. Η αντικειμενοστραφής αυτή αντιμετώπιση αφενός μεν παρέχει την δυνατότητα εννοιολογικού διαχωρισμού των επιμέρους στοιχείων του συστήματος αφετέρου δε συμβάλει στην αρτιότερη λειτουργία του όλου προγράμματος.

Αρχικά η κλάση **Supfunis** παρέχει την βασική λειτουργικότητα που σχετίζεται με τον υπολογισμό της συνάρτησης λάθους, του μέτρου ομοιότητας, του σήματος στο στρώμα εξόδου καθώς και των σφαλμάτων υποκατάστασης. Αξίζει να σημειωθεί το γεγονός ότι οι παραπάνω λειτουργίες ενσωματώνονται σε αντίστοιχες συναρτήσεις που έχουν κατασκευαστεί με τέτοιο τρόπο ώστε να μπορούν να κληθούν ως συναρτήσεις κλάσης (αποφεύγοντας έτσι την δημιουργία αντικειμένων).

Η συνάρτηση **erf(x)** υπολογίζει την τιμή της συνάρτηση λάθους όπως αυτή περιγράφεται στο [Π 1].

$$erf_{JAVA}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-z^2} dz \qquad (7.4.1)$$

όμως η συνάρτηση λάθους όπως ορίζεται για το SuPFuNIS είναι η:

$$erf_{SuPFuNIS}(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-\frac{1}{2}t^2} dt \qquad (7.4.2)$$

επομένως όταν χρειάζεται ο υπολογισμός της συνάρτησης λάθους όπως αυτή περιέχεται στην περιγραφή του SuPFuNIS ενδείκνυται να γίνεται ο ακόλουθος μετασχηματισμός:

$$erf_{SuPFuNIS}(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-\frac{1}{2}t^2} dt \; = [\, t = \sqrt{2}z \,] = \frac{\sqrt{2}}{\sqrt{2\pi}} \int_0^{\frac{x}{\sqrt{2}}} e^{-z^2} dz \; = \frac{1}{2}\frac{2}{\sqrt{\pi}} \int_0^{\frac{x}{\sqrt{2}}} e^{-z^2} dz$$

$$erf_{SuPFuNIS}(x) = \frac{1}{2} erf_{JAVA}(\frac{x}{\sqrt{2}}) \qquad (7.4.3)$$

Η συνάρτηση **mutualSubsethood(xc,xs,wc,ws)** δέχεται τις τιμές του κέντρου και της διασποράς δυο συναρτήσεων συμμετοχής Gauss και υπολογίζει τον βαθμό ομοιότητας τους (σχέση (4.3.3)). Αυτό το επιτυγχάνει καλώντας την **intersection(xc,xs,wc,ws)** μέθοδος η οποία υπολογίζει το μέτρο της τομής δυο ασαφών συνόλων που έχουν Gauss συναρτήσεις συμμετοχής (σχέσεις (4.3.7) → (4.3.12)).

Σε αυτό το σημείο αξίζει να σημειωθεί ότι καθ' όλη την έκταση της υλοποίησης WiDIFuNeCS τα διανύσματα του κέντρου και της διασποράς των παραμέτρων του συστήματος αποθηκεύονται σε πίνακες με συστηματικό τρόπο. Τα κέντρα και οι διασπορές του διανύσματος εισόδου αποθηκεύονται σε διαφορετικούς πίνακες διαστάσεων ίσων με τα επιμέρους χαρακτηριστικά κάθε προτύπου εισόδου. Για παράδειγμα εάν τα χαρακτηριστικά κάθε διανύσματος εισόδου είναι n τότε δημιουργούνται πίνακες της μορφής xc_{nx1}, xs_{nx1}. Τα κέντρα και οι διασπορές των συναπτικών βαρών αποθηκεύονται σε δισδιάστατους πίνακες. Η πρώτη διάσταση ταυτίζεται με τον αριθμό των κόμβων εισόδου ενώ η δεύτερη με τον αριθμό των κανόνων. Έτσι επεκτείνοντας το προηγούμενο παράδειγμα εφόσον ο αριθμός των κανόνων είναι q τότε δημιουργούνται πίνακες της μορφής wc_{nxq}, ws_{nxq}. Ομοίως τα κέντρα και οι διασπορές των κόμβων εξόδου αποθηκεύονται σε δισδιάστατους πίνακες. Η πρώτη διάσταση ταυτίζεται με τον αριθμό των κανόνων ενώ η δεύτερη με τον αριθμό των κόμβων εξόδου. Κατά συνέπεια εφόσον ο αριθμός των κόμβων εξόδου είναι p δημιουργούνται πίνακες της μορφής vc_{qxp}, vs_{qxp}.

Η μέθοδος **testing(xc[],xs[],wc[][],ws[][],vc[][],vs[][])** δέχεται ένα πρότυπο εισόδου καθώς και τις παραμέτρους του νευρο-ασαφούς δικτύου σε μορφή πινάκων και επιστρέφει το υπολογιζόμενο σήμα εξόδου (σχέσεις (4.3.13) → (4.3.17)). Από την άλλη πλευρά η συνάρτηση **getMaxIndex(pin[])** υλοποιεί κατά κάποιο τρόπο την μέθοδο winner takes all επιστρέφοντας τον δείκτη του μεγαλύτερου στοιχείου του πίνακα που δέχεται ως είσοδο.

Η τελευταία μέθοδος της Supfunis κλάσης ουσιαστικά υπολογίζει το πλήθος λανθασμένων ταξινομήσεων που συμβαίνουν σε ένα σύνολο διανυσμάτων εισόδου. Η συνάρτηση **countResubstitutions(data[][],xs[],wc[][],ws[][],vc[][],vs[][])** δέχεται τις παραμέτρους του δικτύου καθώς και έναν πίνακα του οποίου οι γραμμές περιέχουν τις τιμές των χαρακτηριστικών και την κατηγορία κάθε διανύσματος εισόδου, ως αποτέλεσμα επιστρέφει τον αριθμό λανθασμένων ταξινομήσεων.

Ακολούθως η κλάση **Learning** είναι επιφορτισμένη με την υλοποίηση του αλγορίθμου ανάστροφης διάδοσης με την μέθοδο κλίσης. Εξαιτίας του γεγονότος ότι περιλαμβάνει έναν αριθμό μεθόδων που χρειάζεται να επικοινωνούν μέσω κοινών

μεταβλητών αυτές οι μεταβλητές έχουν οριστεί ως μεταβλητές ομότυπου. Έτσι για να γίνει χρήση της λειτουργικότητας της χρειάζεται να δημιουργηθεί ένα αντικείμενο της.

Οι μέθοδοι **pdevc(j,k), pdevs(j,k), pdewc(i,j), pdews(i,j), pdexs(i), pdyz(k,j), pdzE(j,i), pdEwc(i,j), pdEws(i,j), pdExs(i,j)** υπολογίζουν τις μερικές παραγώγους των σχέσεων (4.4.4) → (4.4.11) αντίστοιχα. Η δε συνάρτηση **createPartialDerivatives()** αποθηκεύει στους πίνακες pdcwc, pdcws, pdcxs τις μερικές παραγώγους των σχέσεων (4.4.12) → (4.4.27).

Η μέθοδος **gradientDescent(h,a,d[],xc[],xs[],wc[][],ws[][],vc[][],vs[][],pxs[], pwc[][],pws[][],pvc[][],pvs[][])** συγκεντρώνοντας την λειτουργικότητα των παραπάνω συναρτήσεων υλοποιεί τον αλγόριθμο backpropagation με την μέθοδο κλίσης όπως περιγράφεται μεταξύ άλλων από τις σχέσεις (4.4.2) και (4.4.3). Οι πίνακες που περνούν ως ορίσματα χρησιμοποιούνται στους υπολογισμούς αλλά στην συνέχεια χρησιμοποιούνται για να επιστραφούν οι ανανεωμένες τιμές των παραμέτρων του δικτύου. Αξίζει να σημειωθεί ότι κατά την επιβλεπόμενη εκπαίδευση γίνεται χρήση της τροποποίησης [§ 4.8]. Τα ορίσματα h και a είναι οι τιμές του learning rate και momentum αντίστοιχα. Τέλος εφόσον το διάνυσμα επιθυμητών εξόδων δεν είναι διαθέσιμο, οπότε το όρισμα d[] είναι null, τότε αυτό εκτιμάται [§ 7.3].

Το σύστημα WiDIFuNeCS παρέχει την δυνατότητα να επιλέγεται πολυεπίπεδος ταξινομητής [§ 7.2]. Έτσι κάθε επιτελούμενη διεργασία σε επίπεδο ταξινόμησης ή σε επίπεδο εκπαίδευσης-επανεκπαίδευσης εμπλέκει τρία διαφορετικά σύνολα παραμέτρων συστήματος. Το πρώτο με αναγνωριστικό a αφορά στον βέλτιστο αριθμό κανόνων [§ 4.7]. Το δεύτερο με αναγνωριστικό b αφορά στον υποβέλτιστο αριθμό κανόνων. Ενώ το τρίτο με αναγνωριστικό c σχετίζεται με αριθμό κανόνων μεγαλύτερο του βέλτιστου.

Η κλάση **Classification** περιέχει τον απλό και τον πολυεπίπεδο ταξινομητή, οι οποίοι χρησιμοποιούνται από την φορητή συσκευή [§ 7.6]. Αρχικά η συνάρτηση **getMaxIndex(pin[])** υπολογίζει τους δείκτες των κόμβων εξόδου με τις μεγαλύτερες τιμές. Αυτό γίνεται ούτως ώστε να υπολογίζεται ο τροποποιημένος εκτιμητής αξιοπιστίας (σχέση (3.3.3.1)). Η μέθοδος **isEqual(a[],b[])** συγκρίνει τα δυο πρότυπα εισόδου που δέχεται ως ορίσματα και επιστρέφει true εάν είναι πανομοιότυπα ειδάλλως επιστρέφει false. Ο λόγος που χρησιμοποιείται είναι για να αποφεύγονται ταξινομήσεις διαδοχικών προτύπων που είναι ίδια μιας και η κατηγορία τους είναι ήδη γνωστή.

Η βασική λειτουργία της Classification λαμβάνει χώρα στην μέθοδο **run()**. Επαναληπτικά αυτό που συμβαίνει είναι το εξής: ανακτάται από μια στοίβα το πιο πρόσφατο πρότυπο εισόδου, εάν αυτό ταυτίζεται με το προηγούμενο τότε δεν συμβαίνει τίποτα μιας και η κατηγορία αυτού του προτύπου έχει ήδη υπολογιστεί. Σε αντίθετη περίπτωση από κάθε σύνολο κανόνων υπολογίζεται η κατηγορία του προτύπου καθώς και η αντίστοιχη τιμή του μέτρου αξιοπιστίας [§ 3.3.3]. Στην συνέχεια υπολογίζεται το συνολικό αποτέλεσμα όπως περιγράφεται στο [§ 7.2]. Η μόνη διαφορά στην περίπτωση που δεν έχει επιλεγεί πολυεπίπεδος ταξινομητής είναι ότι οι υπολογισμοί γίνονται για το σύνολο με τον βέλτιστο αριθμό κανόνων ενώ η τελευταία διαδικασία εξαγωγής του τελικού αποτελέσματος παραλείπεται αφού είναι περιττή.

Η κλάση **Initial** υλοποιεί την εκπαίδευση του ταξινομητή σε ένα συγκεκριμένο πρόβλημα για λογαριασμό το server. Με την μέθοδο κατασκευαστή ορίζεται το αναγνωριστικό του συγκεκριμένου προβλήματος ταξινόμησης, ο αριθμός των χαρακτηριστικών κάθε διανύσματος εισόδου, το αρχείο που περιέχει τα πρότυπα εκπαίδευσης, ο βέλτιστος αριθμός κανόνων, ο αριθμός εποχών εκπαίδευσης καθώς και η

επιθυμητή ακρίβεια υποκατάστασης **Initial(name,inputs,file,aRules,epochs,accuracy, arg,btn,btnSwitch)**. Το αρχείο που περιέχει τα πρότυπα εκπαίδευσης πρέπει να περιέχει διατεταγμένα τα χαρακτηριστικά κάθε προτύπου ενώ αμέσως μετά πρέπει να περιέχει την κατηγορία κάθε προτύπου σε αριθμητική μορφή. Στην συνέχεια μπορεί να βρίσκεται το επόμενο πρότυπο εκπαίδευσης, η αντίστοιχη κατηγορία και ούτω καθ' εξής. Τόσο τα χαρακτηριστικά κάθε προτύπου όσο και τα πρότυπα μεταξύ τους μπορεί να είναι χωρισμένα με οποιοδήποτε αριθμό χαρακτήρων κενών, στηλοθετών, νέων γραμμών και επαναφορών.

Η αρχικοποίηση των παραμέτρων του δικτύου γίνεται ακολουθώντας ορισμένους κανόνες. Έτσι κάθε τιμή διασποράς είναι ένας τυχαίος αριθμός στο διάστημα [0.2, 0.9], τα κέντρα των μετασυναπτικών βαρών λαμβάνουν τυχαίες τιμές στο διάστημα [0, 1]. Τα κέντρα των προσυναπτικών βαρών αρχικοποιούνται τυχαία στο διάστημα μεταξύ της ελάχιστης και της μέγιστης τιμής κάθε επιμέρους χαρακτηριστικού, λαμβάνοντας υπόψη βέβαια τα επιμέρους χαρακτηριστικά όλων των προτύπων εκπαίδευσης. Επιπλέον με την **storeWeights()** όλες οι παράμετροι του συστήματος για κάθε σύνολο κανόνων αποθηκεύονται σε αντίστοιχα byte αρχεία. Τα παραπάνω επιτυγχάνονται συνδυάζοντας τις συναρτήσεις **createSpecNfo(), getMaxValue(pin[][],column), getMinValue(pin[][],column), createRandomArray(dim,min,max), createRandomArray(rows,columns,min,max), createRandomArray(rows,columns,min[],max[]), constructNewWeights()**.

Η εκπαίδευση του συστήματος ταξινόμησης γίνεται στην μέθοδο **run()**. Για κάθε επιμέρους σύνολο κανόνων ακολουθείται μια πανομοιότυπη διαδικασία. Αρχικά ρυθμίζονται οι τιμές των συντελεστών εκπαίδευσης και ορμής στην τιμή 0.1. Σε κάθε εποχή αυτές οι δυο τιμές μειώνονται γραμμικά έτσι ώστε στην τελευταία εποχή η τιμή τους να έχει γίνει 0.0001. Κάθε εποχή περιλαμβάνει την παρουσίαση όλων των προτύπων εκπαίδευσης μια φορά ενώ η ανανέωση των παραμέτρων γίνεται (ανά πρότυπο) βάσει του αλγορίθμου backpropagation με την μέθοδο κλίσης. Ο τερματισμός αυτής της διαδικασίας γίνεται όταν συμπληρωθεί ο απαιτούμενος αριθμός εποχών εκπαίδευσης ή όταν επιτευχθεί ικανοποίηση του ορίου της ακρίβειας υποκατάστασης, δηλαδή όταν το ποσοστό των σωστών ταξινομήσεων γίνει μεγαλύτερο είτε ίσο της τιμής resubstitution accuracy. Με το τέλος της παραπάνω διαδικασίας και για τα τρία σύνολα κανόνων αποθηκεύονται εκ νέου όλες οι παράμετροι του συστήματος σε αντίστοιχα byte αρχεία.

Η κλάση **Update** υλοποιεί την επανεκπαίδευση του ταξινομητή σε ένα συγκεκριμένο πρόβλημα για λογαριασμό το server και είναι κατά κάποιο τρόπο συμπληρωματική της Initial. Με την μέθοδο κατασκευαστή **Update()** ανακτώνται όλες οι παράμετροι του συστήματος από τα αντίστοιχα byte αρχεία. Η μέθοδος **run()** ελέγχοντας ορισμένες συνθήκες εκκινεί την επανεκπαίδευση του συστήματος ταξινόμησης αυτόματα εφόσον κρίνει ότι ικανοποιούνται ορισμένα κριτήρια.

Η επανεκπαίδευση του ταξινομητή βάσει της υβριδικής μάθησης [§ 7.3] γίνεται στις μεθόδους **aUpdate()**, **bUpdate()**, **cUpdate()** (για κάθε σύνολο κανόνων). Ο συνολικός αριθμός εποχών ανά σύνολο κανόνων έχει επιλεγεί να είναι 100, ενώ με κάθε πρότυπο επανεκπαίδευσης γίνονται τέσσερις προσπάθειες ανανέωσης των βαρών θέτοντας τις τιμές των α και η ίσες με 0.1, 0.01, 0.001, 0.0001 αντίστοιχα.

7.5 Αισθητήρες

Οι αισθητήρες υλοποιήθηκαν σε λογισμικό αφού η υλοποίηση τους σε υλικό ξεφεύγει από τα πλαίσια της παρούσας διπλωματικής εργασίας. Το υπολογιστικό σύστημα στο οποίο εκτελούνται είναι ένας φορητός υπολογιστής που έχει εγκαταστημένο το JDK1.4 (J2SE). Η ασύρματη επικοινωνία πραγματοποιήθηκε με PCMCIA κάρτα που υποστηρίζει το πρωτόκολλο 802.11 ενώ η διασύνδεση με το PDA έγινε με τοπολογία ad-hoc [§5.2].

7.5.1 Ασύγχρονη Εκπομπή Ενός Χαρακτηριστικού Κάθε Δείγματος

Ο πρώτος τύπος αισθητήρων (τύπος Α) βασίζεται στη λογική ξεχωριστών εφαρμογών ανεξάρτητων μεταξύ τους οι οποίες στέλνουν δεδομένα στην φορητή συσκευή. Ο τύπος αυτός επιτυγχάνει ασύγχρονη επικοινωνία επειδή ακριβώς η αποστολή δεδομένων από έναν αισθητήρα δεν σχετίζεται με την αποστολή δεδομένων από τους άλλους αισθητήρες. Σε κάθε αισθητήρα υπάρχουν οι εξής παράμετροι που μπορεί να καθορίσει ο χρήστης:

1. **η περιοχή τιμών** εντός της οποίας ο αισθητήρας στέλνει δεδομένα. Ο χρήστης καθορίζει το άνω και το κάτω όριο και μετά με χρήση της μεθόδου nextDouble() της κλάσης Random και της έκφρασης r*(b-a)+a επιτυγχάνουμε να προκύπτουν τυχαίοι αριθμοί στο διάστημα [a,b]. Η μέθοδος nextDouble() δημιουργεί τυχαίους αριθμούς κινητής υποδιαστολής 64 bit στο διάστημα [0,1]. Σημειώνουμε ότι το τυχαίο είναι σχετικό αφού στα υπολογιστικά συστήματα το τυχαίο προσομοιώνεται με τη βοήθεια γεννητριών τυχαίων αριθμών (random number generators) [1].

   Απόδειξη του τύπου r*(b-a)+a:

   $0 \le r \le 1$ $\rightarrow$ εφόσον $b > a$
   $0 \le r*(b-a) \le (b-a)$ $\rightarrow$
   $a \le r*(b-a)+a \le (b-a)+a = b$ $\rightarrow$
   $a \le r*(b-a)+a \le b$

   Εάν b=a τότε r=a δηλαδή προκύπτει σταθερή τιμή.
2. **το χρονικό διάστημα καθυστέρησης** (delay time) το οποίο μετριέται σε msec και καθορίζει τον ρυθμό με τον οποίο στέλνει ο αισθητήρας δεδομένα. Ο ρυθμός αυτός περιορίζεται από τα τεχνικά χαρακτηριστικά της πλατφόρμας [Π. 4]. Έχουν διενεργηθεί εκτενή πειράματα με ρυθμούς μεγαλύτερούς από 10 δείγματα /sec, ωστόσο για περιορισμένα χρονικά διαστήματα έχουν επιτευχθεί και υψηλότεροι ρυθμοί. Ο χρόνος καθυστέρησης είναι ένας τυχαίος αριθμός από το 0 έως τον αριθμό που καθορίζει ο χρήστης.

```
try{    double tmp=Integer.parseInt(tim_val)*r.nextDouble();
                sleep((int)tmp);
        } catch(InterruptedException e) {};
```

3. **η IP διεύθυνση** που καθορίζει τον προορισμό στον οποίο στέλνει δεδομένα ο αισθητήρας. Συγκεκριμένα η IP για κάθε συσκευή αισθητήρα προσδιορίζεται από τον χρήστη.
4. **ο αριθμός της θύρας** ο οποίος εξαρτάται από το χαρακτηριστικό το οποίο προσομοιώνει ο αισθητήρας. Στην περίπτωση μας έχουν προκαθοριστεί οι θύρες 8001, 8002, ... που χρησιμοποιούνται για αποστολή των τιμών του πρώτου, του δευτέρου, … χαρακτηριστικού.

Δημιουργείται σύνδεση ανάμεσα στον αισθητήρα και στο PDA με την βοήθεια ενός clientSocket [§ 6.7.4]:

Δηλώσεις

```
Socket socket;
OutputStream os;
InputStream is;
String ran_num;
```

Κώδικας

```
InetAddress ipaddress = InetAddress.getByName(ip_adr);
InetSocketAddress socketaddress =
                new InetSocketAddress( ipaddress , Integer.parseInt(port_num));
socket = new Socket();
try {
        socket.setSoTimeout(1000); //1second
    }
catch(SocketException e)
    {
      e.printStackTrace();
    }
socket.connect(socketaddress);
os = socket.getOutputStream();
is = socket.getInputStream();
os.write(ran_num.getBytes());
os.flush();
ascii = is.read();
```

Δηλαδή ο αισθητήρας προσπαθεί να συνδεθεί στην ipadress και στο port που καθορίζει ο χρήστης, αρχικοποιεί τα os ,is και στέλνει τυχαίους αριθμούς ran_num, byte προς byte (**ran_num.getBytes()**), στο serversocket με το οποίο έχει συνδεθεί. Επισημαίνεται ότι ένας χαρακτήρας αντιστοιχεί σε ένα byte. Εάν η αποστολή είναι επιτυχής τότε ο αισθητήρας λαμβάνει ένα χαρακτήρα '@' ως ένδειξη ορθής αποστολής κάθε αριθμού. Η όλη διαδικασία ανάγεται σε ξεχωριστό νήμα (Client.class) [§ 6.7.1.2]. Ιδιαίτερα σημαντικός είναι ο χρωματισμός του πλαισίου στο οποίο ο χρήστης καθορίζει τον αριθμό της θύρας. Κόκκινο σημαίνει πως ο αισθητήρας βρίσκεται σε ανενεργή κατάσταση, στην οποία ο χρήστης μπορεί να ρυθμίσει τις επιμέρους παραμέτρους του αισθητήρα. Πράσινο σημαίνει ότι έχει επιτευχθεί σύνδεση και το PDA λαμβάνει δεδομένα με ορθό τρόπο τέλος πορτοκαλί

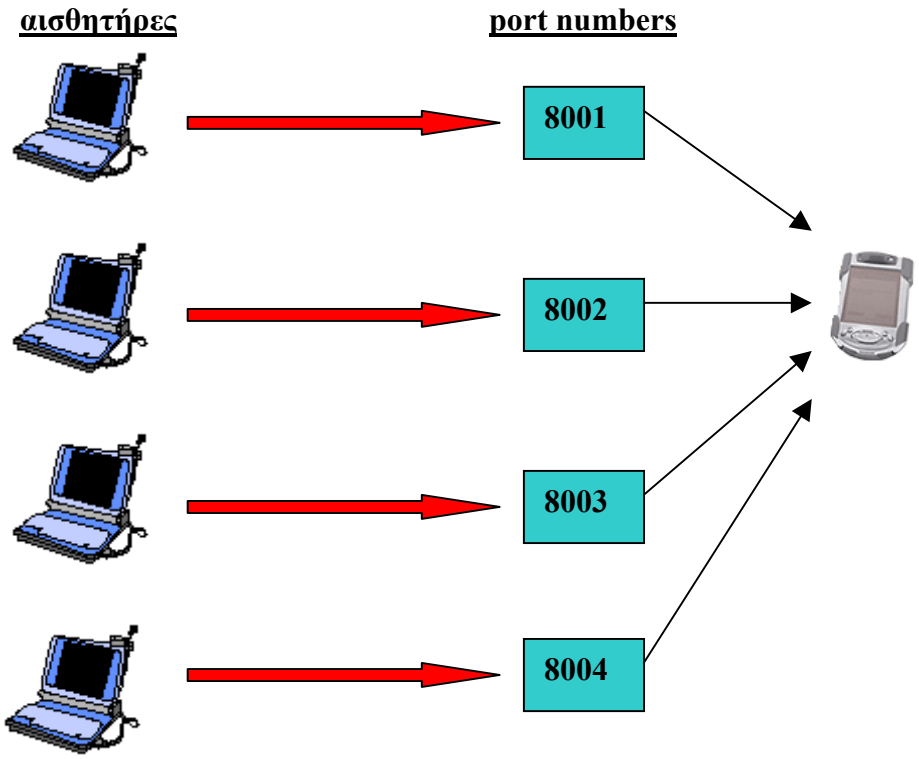
σημαίνει ότι παρότι ο αισθητήρας λειτουργεί, η σύνδεση με το PDA έχει διακοπεί. Ο αισθητήρας σε αυτή την περίπτωση δεν σταματά να λειτουργεί αλλά προσπαθεί να συνδεθεί με το PDA μέχρι να το καταφέρει ή μέχρι ο χρήστης να διακόψει τη λειτουργία του. Τέλος η εντολή **socket.setSoTimeout(1000)** αναφέρεται στην περίπτωση που ενώ το socket συνδέθηκε στο serversocket, τα os, is δεν ανταλλάσσουν δεδομένα οπότε ο αισθητήρας παίρνει την πρωτοβουλία να διακόψει την σύνδεση και να αρχίσει μια καινούργια. Το διάστημα το οποίο περιμένει ο αισθητήρας πριν διακόψει την σύνδεση είναι ο αριθμός μέσα στην παρένθεση (1000msec=1sec). Αυτή η διαδικασία κρίθηκε απαραίτητη ώστε να μην εγκλωβίζεται ο αισθητήρας σε συνδέσεις που ουσιαστικά δεν υφίστανται. Η ίδια λογική ακολουθείται σε όλα τα επιμέρους σημεία της εφαρμογής. Εάν εγερθεί εξαίρεση [§ 6.7.3] τότε εγκαταλείπεται η τρέχουσα σύνδεση και εκκινείται η διαδικασία για την δημιουργία μιας καινούργιας σύνδεσης.

Σημειώνουμε επίσης, ότι οι τιμές μεταδίδονται byte-byte και τελειώνουν όλες με 'X'. Ο 'X' χαρακτήρας σημαίνει το τέλος μετάδοσης μιας τιμής και προστίθεται σε όλες τις τιμές .

ran_num= r + 'X';

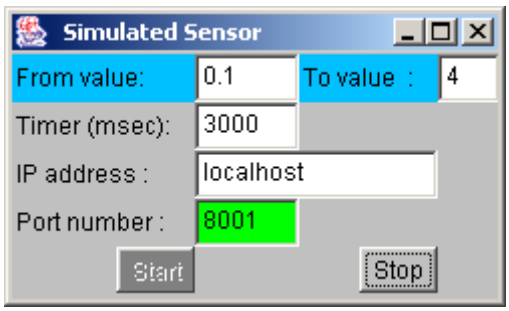
Η σύμβαση ισχύει και για τους δύο τύπους αισθητήρων.

Σχηματικά η λειτουργία των αισθητήρων τύπου Α (simple sensors) παρουσιάζεται στο παρακάτω σχήμα:

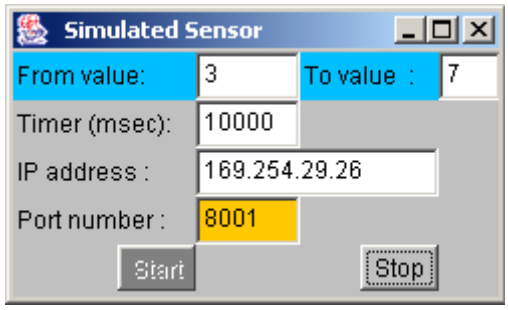


Σχήμα 7.5.1.1 Λειτουργία αισθητήρων τύπου Α για τέσσερις εισόδους

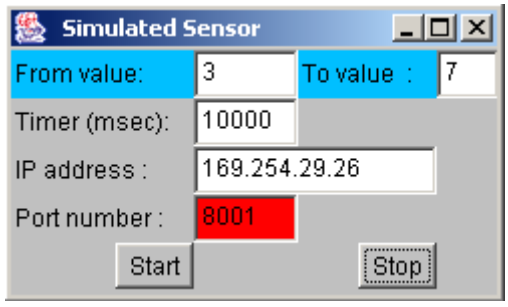
Στα παρακάτω σχήματα φαίνεται το περιβάλλον λειτουργίας των αισθητήρων.



Σχήμα 7.5.1.2 Κανονική λειτουργία αισθητήρα



Σχήμα 7.5.1.3 Κανονική λειτουργία αισθητήρα αλλά το PDA δεν στέλνει δεδομένα



Σχήμα 7.5.1.4 Κατάσταση μη-λειτουργίας αισθητήρα

7.5.2 Εκπομπή Όλων των Χαρακτηριστικών Κάθε Δείγματος

Οι αισθητήρες αυτού του τύπου (τύπος Β) στηρίζονται στη λογική συγχρονισμένης επικοινωνίας ανάμεσα στους αισθητήρες και στο PDA. Σε αυτή την περίπτωση υπάρχει ένας μοναδικός αισθητήρας ο οποίος αναλαμβάνει να στέλνει όλα τα χαρακτηριστικά κάθε δείγματος. Οι τιμές των χαρακτηριστικών προκύπτουν από δεδομένα που διαβάζει ο αισθητήρας από ένα αρχείο. Οι παράμετροι 2, 3 είναι ίδιες με αυτές των αισθητήρων του προηγούμενου τύπου. Οι διαφορές έγκεινται στα:
1. τώρα αντί για πεδίο τιμών έχουμε **όνομα αρχείου** το οποίο αποτελείται από τιμές χαρακτηριστικών οι οποίες είναι αποθηκευμένες σε σειριακή διάταξη ομάδων. Κάθε ομάδα αποτελείται από τόσες τιμές όσες είναι οι είσοδοι του προβλήματος που καλείται να αντιμετωπίσει το SuPFuNIS. Με απλά λόγια, το αρχείο αποτελείται από n-άδες (όπου n τα χαρακτηριστικά κάθε προτύπου) διατεταγμένες η μία μετά την άλλη

4. ο αριθμός θύρας είναι μοναδικός αφού ο αισθητήρας είναι μοναδικός και προσυμφωνημένα είναι ο 8000
Η λειτουργία του αισθητήρα είναι η εξής: μόλις τίθεται σε λειτουργία διαβάζει όλο το αρχείο που του όρισε ο χρήστης για να βρει το πλήθος των δεδομένων και μετά αποθηκεύει τις τιμές σε ένα πίνακα:

Δηλώσεις:

```
FileInputStream fis ;
BufferedInputStream bis ;
DataInputStream dis;
int num length;
```

Κώδικας

```
try
  {
     fis = new FileInputStream(file_name);
     bis = new BufferedInputStream(fis);
     dis = new DataInputStream(bis);
     while(true)
              {
                 dis.readDouble();
                 num_lenght++;
              }
  }
 catch(IOException e)
  {
     e.printStackTrace();
     try
     {
              dis.close();
     }
     catch
     (IOException fdh)
     {
              fdh.printStackTrace();
     }
  }
try
  {
     fis = new FileInputStream(file_name);
     bis = new BufferedInputStream(fis);
     dis = new DataInputStream(bis);
              strb=new String[num_lenght/num][num+1];
     for(int i=0;i<strb.length;i++)
     {
              for(int j=1;j<strb[0].length;j++)
              {
              strb[i][j]=String.valueOf(dis.readDouble());
              }
     }
     }
```

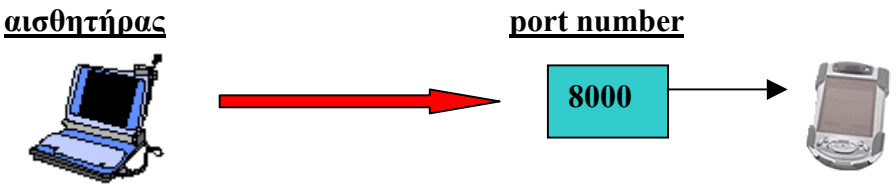
```
catch(IOException e)
        {
          e.printStackTrace();
          try
          {
                  dis.close();
          }
          catch
          (IOException fdh)
          {
                  fdh.printStackTrace();
          }
        }
```

Η επόμενη κίνηση είναι να συνδεθεί ο αισθητήρας με το PDA μέσω socket και να αρχίσει τη μετάδοση n-άδων. Η διαδικασία μετάδοσης είναι ίδια με αυτή του τύπου Α αλλά αυτή τη φορά είναι κλεισμένη σε for loop και επαναλαμβάνεται n φορές.

```
for(int b=1;b<=Integer.parseInt(num_feat);b++)
{
//μετάδοση τιμών
}
```

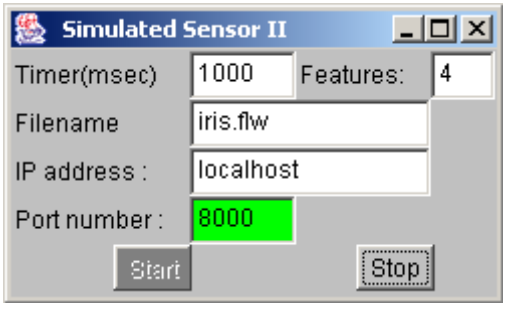
Η όλη διαδικασία επίσης ανάγεται σε ξεχωριστό νήμα (ClientII.class) [§ 6.7.1.2] ενώ ακολουθείται η ίδια λογική στον χρωματισμό του πλαισίου στο οποίο ο χρήστης καθορίζει τον αριθμό της θύρας.

Σχηματικά η λειτουργία των αισθητήρων τύπου Β (multi sensors) παρουσιάζεται στο παρακάτω σχήμα:

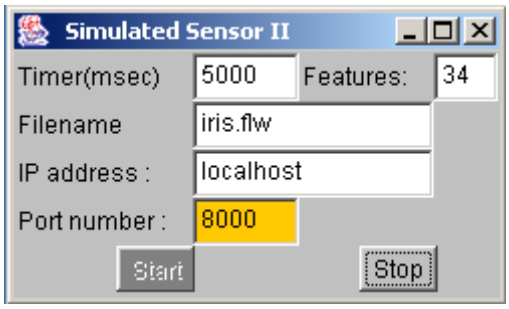


Σχήμα 7.5.2.1 Λειτουργία αισθητήρων τύπου Β

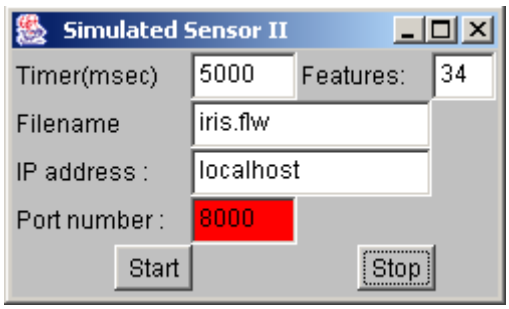
Στα παρακάτω σχήματα φαίνεται το περιβάλλον λειτουργίας των αισθητήρων



Σχήμα 7.5.5.2 Κανονική λειτουργία αισθητήρα



Σχήμα 7.5.2.3 Κανονική λειτουργία αισθητήρα αλλά το PDA δεν στέλνει δεδομένα



Σχήμα 7.5.2.4 Κατάσταση μη-λειτουργίας αισθητήρα

7.6 Φορητή Συσκευή

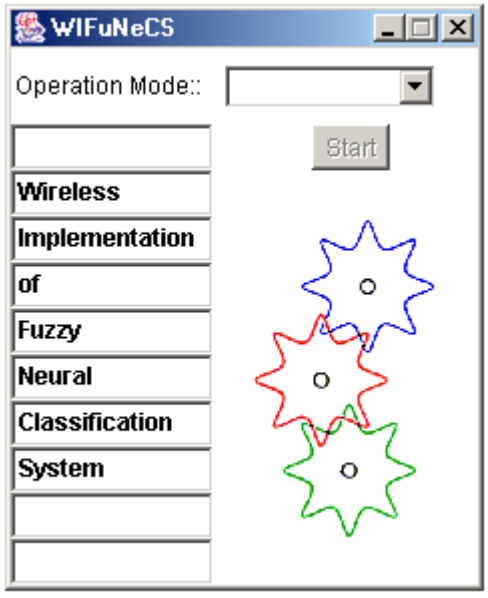
7.6.1 Η Εφαρμογή Java της Φορητής Συσκευής

Η φορητή συσκευή στην συγκεκριμένη υλοποίηση είναι ένας προσωπικός ψηφιακός βοηθός (Personal Digital Assistant) [Π 4]. Η γραφική διεπαφή που έχει κατασκευαστεί για το PDA περιλαμβάνει τρεις καταστάσεις λειτουργίας.

Η πρώτη από αυτές αποτελεί ένα μενού όπου υπάρχει η δυνατότητα επιλογής μεταξύ των λειτουργιών της ταξινόμησης και της μάθησης του WiDIFuNeCS. Ο πηγαίος κώδικας αυτής της αρχικής διεπαφής βρίσκεται στην κλάση **MainFrame**. Στην μέθοδο εγκατάστασης **MainFrame(title,path)** περνάει ως όρισμα ο υπέρτιτλος του παραθύρου της εφαρμογής, καθώς και η διαδρομή του φακέλου που περιέχει όλα τα αρχεία που σχετίζονται με το πρόγραμμα. Από την άλλη πλευρά με αυτή τη μέθοδο σχεδιάζεται το παράθυρο της εφαρμογής χρησιμοποιώντας την διάταξη δομημένων πλεγμάτων της Java και καθορίζονται πλήρως όλα τα επιμέρους συστατικά του παραθύρου.

Με τις μεθόδους **setMain(), setClassification(), setTraining()** ορίζονται τα περιεχόμενα των πεδίων κειμένου έτσι ώστε να εμφανίζονται συνοπτικές πληροφορίες σχετικές με την επιλογή που ετοιμάζεται να κάνει ο χρήστης. Με την μέθοδο **action(vnt,arg)** γίνεται χειρισμός συμβάντων. Αναλόγως της επιλογής που θα κάνει ο χρήστης θα μεταφερθεί στο μενού της αντίστοιχης λειτουργίας που επέλεξε.

Η κλάση **Poster** με τις μεθόδους **Poster(path)**, **setPoster(selection)**, **paint(g)** παρέχει την απαραίτητη λειτουργικότητα για επιλογή κάποιας εικόνας και εμφάνισης της στο παράθυρο της εφαρμογής.



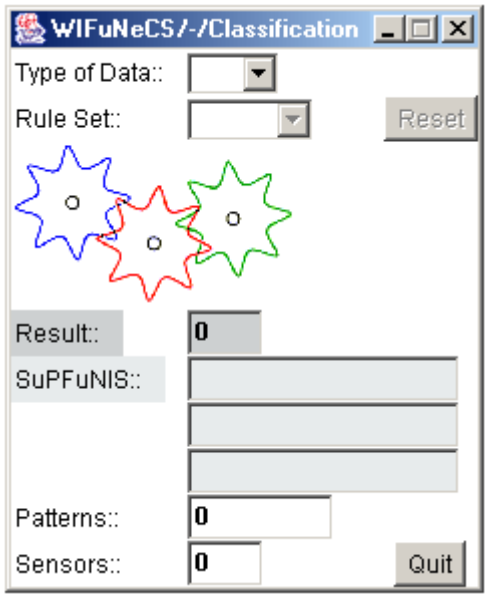
Σχήμα 7.6.1.1: Αρχικό παράθυρο της εφαρμογής στο οποίο εμφανίζονται γενικές πληροφορίες.

Σχήμα 7.6.1.2: Αρχικό παράθυρο της εφαρμογής στο οποίο εμφανίζονται πληροφορίες σχετικές με την λειτουργία της ταξινόμησης.

Σχήμα 7.6.1.3: Αρχικό παράθυρο της εφαρμογής στο οποίο εμφανίζονται πληροφορίες σχετικές με την λειτουργία της αρχικοποίησης-ανανέωσης των παραμέτρων του συστήματος.

Η κλάση **ClassificationFrame** είναι επιφορτισμένη με την δημιουργία του παραθύρου ταξινόμησης, ενώ παράλληλα υλοποιεί τις ακριβείς λειτουργίες που επιλέγει ο χρήστης. Ο πρωταρχικός ρόλος της μεθόδου εγκατάστασης **ClassificationFrame(title,real,path)** είναι να κατασκευάσει και να εμφανίσει το παράθυρο της εφαρμογής (χρησιμοποιώντας την διάταξη δομημένων πλεγμάτων της Java). Τα ορίσματα που δέχεται είναι το πρώτο συστατικό του υπέρτιτλου του παραθύρου, το εάν τα πρότυπα εισόδου θα τα λάβει από ένα αισθητήρα (false) ή πολλούς (true) και τέλος η διαδρομή του φακέλου που περιέχει όλα τα αρχεία που σχετίζονται με το πρόγραμμα. Επίσης μια σημαντική διεργασία που εκτελείται είναι ότι φορτώνονται τα διαθέσιμα προβλήματα ταξινόμησης που έχουν οριστεί από τον server [§ 7.7]. Αυτά περιέχονται στο αρχείο info.txt, μια πιθανή εκδοχή αυτού του

αρχείου θεωρώντας ότι υπάρχουν δυο διαθέσιμα προβλήματα ταξινόμησης φαίνεται στην συνέχεια:
iris 4 5 3 7 3
medical 3 8 5 11 1
Το string στην αρχή αποτελεί το αναγνωριστικό του εκάστοτε προβλήματος ταξινόμησης. Ο δεύτερος αριθμός είναι το πλήθος των κόμβων εξόδου, οι τρεις επόμενοι είναι το πλήθος των χρησιμοποιούμενων κανόνων από τον πολυεπίπεδο ταξινομητή ενώ ο τελευταίος αριθμός είναι το μέγεθος του διανύσματος εξόδου του ταξινομητή. Η συνάρτηση **setZero()** καθαρίζει τα επιμέρους πεδία του παραθύρου της εφαρμογής θέτοντας τα μηδέν ή θέτοντας τα περιεχόμενα τους ίσα με το κενό.



Σχήμα 7.6.1.4: Το παράθυρο της εφαρμογής WiDIFuNeCS που σχετίζεται με την λειτουργία της ταξινόμησης.

Ο χειρισμός των επιλογών που κάνει ο χρήστης, άρα και των επιτελούμενων διεργασιών λαμβάνει χώρα στην μέθοδο **action(vnt,arg)**. Η επιλογή ενός προβλήματος από την λίστα Type of Data, εκκινεί την διαδικασία συλλογής δεδομένων από τους αισθητήρες τα οποία αποθηκεύονται στο αντίστοιχο (ως προς το πρόβλημα) .dat αρχείο. Συγχρόνως ανακτώνται οι παράμετροι του ταξινομητή από τα byte αρχεία στα οποία είναι αποθηκευμένες. Τα δείγματα που έχουν αποθηκευτεί εμφανίζονται στο πεδίο Patterns ενώ ο αριθμός των αισθητήρων από τους οποίους λαμβάνονται πρότυπα προς ταξινόμηση αναγράφεται στο πεδίο Sensors.

Με την επιλογή ενός τύπου ταξινομητή (απλού ή πολυεπίπεδου) από την λίστα Rule Set εκκινείται η διεργασία ταξινόμησης. Το τελικό αποτέλεσμα ανεξαρτήτως ταξινομητή εμφανίζεται στο πεδίο Result. Στον πίνακα SuPFuNIS εμφανίζονται έξοδοι και στοιχεία για κάθε επιμέρους ταξινομητή εφόσον χρησιμοποιούνται τρεις ειδάλλως εμφανίζονται πληροφορίες αναφορικά με τον ταξινομητή με τον βέλτιστο αριθμό κανόνων. Για παράδειγμα θεωρώντας ότι έχει επιλεγεί πολυεπίπεδος ταξινομητής με 10, 5 , 14 κανόνες για ένα πρόβλημα τριών κατηγοριών στον πίνακα SuPFuNIS εμφανίζεται:
2 {10} 0.1243
1 {5} 1.002
3 {14} 0.518

Πατώντας το κουμπί Reset διακόπτονται οι λειτουργίες εξαγωγής συμπερασμάτων και συλλογής δεδομένων. Επιλέγοντας Quit διακόπτονται πάλι όλες οι λειτουργίες και ο χρήστης μεταφέρεται στο αρχικό παράθυρο της εφαρμογής. Οι μέθοδοι **start_multi_sensor()**, **stop_multi_sensor()** εκκινούν και σταματούν της διαδικασία λήψης δεδομένων από τον αισθητήρα που εκπέμπει ολόκληρα τα πρότυπα εισόδου. Αντίθετα οι μέθοδοι **start_simple_sensor()**, **stop_simple_sensor()** εκκινούν και σταματούν την διαδικασία λήψης δεδομένων από τους αισθητήρες που εκπέμπουν ένα χαρακτηριστικό κάθε προτύπου εισόδου έκαστος.

Στο τμήμα της εφαρμογής που σχετίζεται με την ταξινόμηση, εκτός της ClassificationFrame χρησιμοποιούνται και ορισμένες άλλες κλάσεις. Η πιο σημαντική από αυτές είναι η κλάση **Result**. Ένα αντικείμενο αυτής της κλάσης δημιουργείται κάθε φορά που εκκινείται μια διαδικασία συλλογής δεδομένων εισόδου. Η δημιουργία αυτή γίνεται με τις μεθόδους **Result(features[],patterns)**, **Result(features,patterns)** όπου ουσιαστικά ορίζεται ο αριθμός χαρακτηριστικών κάθε προτύπου εισόδου για το εξεταζόμενο πρόβλημα. Συνακόλουθα περνάει ως όρισμα ένα αντικείμενο TextField έτσι ώστε να είναι δυνατή η εμφάνιση του αριθμού των διανυσμάτων εισόδου που έχουν συλλεγεί.

Με την μέθοδο **isEqual(a[],b[])** ελέγχεται εάν δυο διαδοχικά διανύσματα εισόδου ταυτίζονται ή όχι. Η σημαντικότερη όμως λειτουργία που παρέχει η κλάση Result προέρχεται από την συγχρονισμένη μέθοδο **access(features[],name)**. Κατ' αρχάς αυτή η μέθοδος ορίζεται συγχρονισμένη αφού είναι απαραίτητο οι διαφορετικές διεργασίες που την καλούν να έχουν αποκλειστική πρόσβαση σε αυτή. Είναι λάθος δυο ή περισσότερες διεργασίες συλλογής δεδομένων να μεταβάλλουν με μη συστηματικό τρόπο τα περιεχόμενα του προτύπου εισόδου ενώ παράλληλα η διεργασία εξαγωγής συμπερασμάτων να το χρησιμοποιεί για να πραγματοποιήσει μια ταξινόμηση. Εάν το πρώτο όρισμα είναι null σημαίνει ότι η καλούσα διεργασία είναι η ταξινόμηση οπότε επιστρέφεται το πιο πρόσφατο πρότυπο εισόδου που έχει συλλεγεί. Αντίθετα εάν το πρώτο όρισμα περιέχει τις τιμές των χαρακτηριστικών κάποιου προτύπου εισόδου τότε αυτό ελέγχεται ως προς το αμέσως προηγούμενο του για να διαπιστωθεί εάν διαφέρει. Εάν πράγματι διαφέρει τότε αποθηκεύεται στο αντίστοιχο byte αρχείο όπως αυτό προσδιορίζεται από το δεύτερο όρισμα που δέχεται η μέθοδος.

Με τις μεθόδους **Weights(path,nam,inputs,aRules,bRules,cRules,outputs)**, **transfer(name,extension,elements)**, **transfer(name,extension,rows,columns)** της κλάσης **Weights** ανακτώνται οι παράμετροι του ταξινομητή (απλού ή πολυεπίπεδου) που βρίσκονται στα αντίστοιχα byte αρχεία. Σημαντικό είναι το γεγονός ότι οι ανακτώμενες τιμές αποθηκεύονται σε δομές πίνακα με τον τρόπο που περιγράφεται στο [§ 7.4].

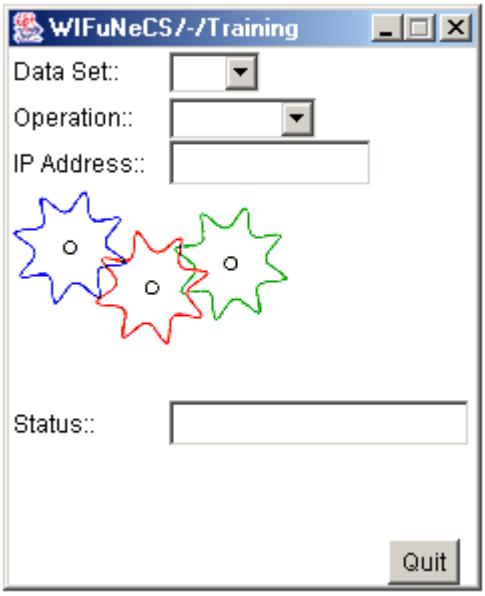
Η μέθοδος **addConstraints(gbl,c,gbc,gx,gy,gw,gh,wx,wy,ar)** της κλάσης **Constraints** δίδει την δυνατότητα ορισμού των χαρακτηριστικών κάθε κελιού στην δομημένη διάταξη πλέγματος με απλό και ευκρινή τρόπο. Η wrapper τάξη **Double** περιέχει μια static συνάρτηση την **parseDouble(str)** η οποία δεχόμενη ως όρισμα ένα αλφαριθμητικό που αναπαριστά έναν οποιοδήποτε αριθμό επιστρέφει την τιμή αυτού ως double. Ο λόγος που κατασκευάστηκε αυτή η κλάση πρέπει να αναζητηθεί στο υποσύνολο της γλώσσας Java που είναι εγκατεστημένο στο PDA το οποίο δεν περιέχει wrapper κλάσεις για τους αριθμούς κινητής υποδιαστολής.

Η κλάση **Movie** που χρησιμοποιείται και στο παράθυρο της εφαρμογής που σχετίζεται με την εκπαίδευση του ταξινομητή έχει ως σκοπό την εμφάνιση κινούμενης εικόνας στα παράθυρα της εφαρμογής. Η μέθοδος εγκατάστασης **Movie(clockwise,path)** ορίζει τον τρόπο περιστροφής των κινουμένων στοιχείων της

εικόνας και επίσης ορίζει την πλήρη διαδρομή του φακέλου που περιέχει τα frames της εικόνας. Η συναρτήσεις **start()** και **stop()** εκκινούν και διακόπτουν την κινούμενη εικόνα. Παράλληλα οι μέθοδοι **run(), update(g), paint(g)** εναλλάσσουν τα καρέ της εικόνας και τα απεικονίζουν στην οθόνη του PDA έτσι ώστε να επιτύχουν την επιθυμητή κίνηση.

Η κλάση **TrainingFrame** είναι επιφορτισμένη με την δημιουργία του παραθύρου αρχικοποίησης-επανεκπαίδευσης, παράλληλα υλοποιεί τις ακριβείς λειτουργίες που επιλέγει ο χρήστης. Ο πρωταρχικός ρόλος της μεθόδου εγκατάστασης **TrainingFrame(title,path)** είναι να κατασκευάσει και να εμφανίσει το παράθυρο της εφαρμογής (χρησιμοποιώντας την διάταξη δομημένων πλεγμάτων της Java). Τα ορίσματα που δέχεται είναι το πρώτο συστατικό του υπέρτιτλου του παραθύρου και η διαδρομή του φακέλου που περιέχει όλα τα αρχεία που σχετίζονται με το πρόγραμμα. Επίσης μια σημαντική διεργασία που εκτελείται είναι ότι φορτώνονται (από το αρχείο info.txt) τα διαθέσιμα προβλήματα ταξινόμησης που έχουν οριστεί από τον server και στα οποία μπορεί να ζητηθεί να γίνει επανεκπαίδευση του ταξινομητή.

Η μέθοδος **isValidIP(str)** ελέγχει εάν το string που δέχεται ως όρισμα αποτελεί έγκυρη IP διεύθυνση, δηλαδή εάν είναι ένας αριθμός μεταξύ 000.000.000.000 και 255.255.255.255. Σε περίπτωση που κάτι τέτοιο είναι αληθές επιστρέφει true, σε αντίθετη περίπτωση επιστρέφει false.



Σχήμα 7.6.1.5: Το παράθυρο της εφαρμογής WiDIFuNeCS που σχετίζεται με την λειτουργία της αρχικοποίησης-επανεκπαίδευσης.

Ο χειρισμός των επιλογών που κάνει ο χρήστης, άρα και των επιτελούμενων διεργασιών λαμβάνει χώρα στην μέθοδο **action(vnt,arg)**. Αξίζει να σημειωθεί το γεγονός ότι ορισμένοι μόνο συνδυασμοί επιλογών από τις λίστες Data Set και Operation είναι έγκυροι με αποτέλεσμα επιτελείται η ζητούμενη λειτουργία. Εάν ο συνδυασμός επιλογών δεν αντιστοιχεί σε κάποια διεργασία ή εάν είναι λανθασμένος τότε στο πεδίο Status εκτυπώνονται κατάλληλα μηνύματα. Έτσι εάν η IP διεύθυνση είναι μη έγκυρη εμφανίζεται το μήνυμα λάθους *Invalid IP Address*, εάν ζητηθεί επανεκπαίδευση αλλά δεν έχει επιλεγεί κάποιο σύνολο ταξινόμησης εμφανίζεται το μήνυμα λάθους *Data Set Not Selected*, τέλος εάν επιλεγεί να αρχικοποιηθεί ο

ταξινομητής σε ένα ήδη υπάρχον πρόβλημα εμφανίζεται το μήνυμα λάθους *Already Initialized*.

Ο χρήστης επιλέγοντας κάποιο Data Set, πληκτρολογώντας την IP διεύθυνση του server (που σχετίζεται με το συγκεκριμένο πρόβλημα ταξινόμησης) και διαλέγοντας Update στην λίστα Operation εισέρχεται στο στάδιο επανεκπαίδευσης του συστήματος ταξινόμησης. Εάν επιλέξει Initialize στην λίστα Operation, δεν επιλέξει Data Set και θέσει την IP διεύθυνση του server (που σχετίζεται με ένα καινούργιο πρόβλημα ταξινόμησης) τότε το πρόγραμμα εισέρχεται στο στάδιο αρχικοποίησης κατά το οποίο ορίζεται στο σύστημα ένα νέο πρόβλημα ταξινόμησης. Στο πεδίο Status εμφανίζονται επίσης μηνύματα σχετικά με την επιτελούμενη λειτουργία. Έτσι στην φάση της αρχικοποίησης ενδέχεται να εμφανιστούν τα ακόλουθα μηνύματα:

Connecting ...	//σύνδεση στο δίκτυο
Connecting Server ...	//σύνδεση με τον server
Server Not Found ...	//ο server στην συγκεκριμένη IP δεν υπάρχει
Initializing ...	//αρχικοποίηση του ταξινομητή
Disconnected from Server ...	//απώλεια σύνδεσης με τον server
New Set Added ...	//επιτυχής ορισμός καινούργιου προβλήματος
Set Not Added ...	//ανεπιτυχής ορισμός καινούργιου προβλήματος
Initialization Completed	//τερματισμός αρχικοποίησης
Disconnected – Retry	//απώλεια της σύνδεσης

Από την άλλη πλευρά στην φάση της επανεκπαίδευσης ενδέχεται να εμφανιστούν τα ακόλουθα μηνύματα:

Connecting ...	//σύνδεση στο δίκτυο
Connecting Server ...	//σύνδεση με τον server
Server Not Found ...	//ο server στην συγκεκριμένη IP δεν υπάρχει
Sending Patterns ...	//αποστολή των προτύπων που έχουν συλλεγεί
Unavailable Patterns	//δεν υπάρχουν πρότυπα προς αποστολή
Disconnected from Server	//απώλεια σύνδεσης με τον server
Updating Weights	//ανανέωση των παραμέτρων του ταξινομητή
Updating Completed	//τερματισμός της ανανέωσης των παραμέτρων
Disconnected – Retry	//απώλεια της σύνδεσης

7.6.2 Λήψη των Δεδομένων που Εκπέμπουν οι Αισθητήρες

Από την πλευρά του PDA έχουν υλοποιηθεί δύο περιπτώσεις αντίστοιχες προς τις περιπτώσεις των αισθητήρων. Η κλάση Server είναι υπεύθυνη για τη διασύνδεση των αισθητήρων τύπου Α με τη συσκευή ώστε το PDA να δέχεται δεδομένα ασύγχρονα, ενώ η ServerII είναι υπεύθυνη για τη διασύνδεση των αισθητήρων τύπου Β με την συσκευή ώστε το PDA να δέχεται δεδομένα κατά σύγχρονο τρόπο. Δημιουργούνται λοιπόν serverSocket [§ 6.7.4]:

serverSocket = new ServerSocket(Integer.parseInt(port)+8000);

Αν έχουμε την περίπτωση Α τότε δημιουργούνται ServerSocket για κάθε αισθητήρα στις θύρες 8001,8002,... τόσα όσες και οι είσοδοι. Στην άλλη περίπτωση δημιουργείται μόνο το ServerSocket στη θύρα 8000.

Εντός της μεθόδου run σε κάθε κλάση (Server, ServerII) τοποθετούνται τα:

```
while(true)
{
        clientSocket=null;
        try
        {
                clientSocket = serverSocket.accept();
        }
        catch(IOException e)
        {
                e.printStackTrace();
                continue;
        }
        if(m_chk==true)
        {
                echo=new EchoThread(clientSocket,this,inp);
                echo.start();
        }
}
```

Για κάθε ένα socket που προσπαθεί να συνδεθεί δημιουργείται ένα νήμα για να το εξυπηρετήσει. Τα νήματα που δημιουργούνται είναι ομότυπα των κλάσεων EchoThread και EchoThreadII. Αυτά τα νήματα έχουν ως σκοπό να αποθηκεύσουν τις τιμές που δέχονται στην μεταβλητή **public static String xc[]** η οποία είναι ο πίνακας που αποθηκεύονται τα δεδομένα εισόδου. Στον συγχρονισμένο τρόπο η προσπέλαση του πίνακα γίνεται μια φορά και αλλάζουν οι τιμές σε όλα τα κελιά του, ενώ στον ασύγχρονο τρόπο έχουμε προσπέλαση κάθε φορά που έρχεται μια νέα τιμή για οποιοδήποτε χαρακτηριστικό. Οι παρακάτω εντολές υλοποιούν την υποδοχή των δεδομένων εισόδου .

```
do
{
        ascii = is.read();
        if(ascii!=(int)'X')
        strb.append((char)ascii);
}
while(ascii!=(int)'X');
```

Όπως είναι φανερό το socket σταματά να διαβάζει μόλις συναντήσει το χαρακτήρα 'X'. Ανάμεσα στο socket και στο serverSocket, δηλαδή ανάμεσα στον αισθητήρα και στο PDA έχει γίνει μια σύμβαση ότι οι τιμές μεταδίδονται χαρακτήρα προς χαρακτήρα και τελειώνουν με τον χαρακτήρα 'X'. Αυτή η σύμβαση δίνει τη δυνατότητα στο PDA να καταλάβει ότι η μετάδοση μιας τιμής τέλειωσε ή ότι έγινε σφάλμα μετάδοσης γεγονός που σημαίνει ότι πρέπει να διακοπεί η σύνδεση και να εκκινηθεί μια καινούργια Είναι πολύ σημαντική αυτή η σύμβαση, γιατί έτσι μπορούμε να μεταδίδουμε τιμές χωρίς να είμαστε υποχρεωμένοι να ξέρουμε από πριν το μέγεθος τους σε byte (χαρακτήρες).

```
os.write('@');
os.flush();
os.close();
```

is.close();
s.close();

Εάν γίνει σωστή μετάδοση μιας τιμής τότε το PDA στέλνει '@', κλείνει τη σύνδεση, ανοίγει καινούργια σύνδεση και περιμένει. Συμπληρωματικά πάλι έχουν χρησιμοποιηθεί Thread για να μπορούμε να σταματάμε και να εκκινούμε την σύνδεση. Η εκκίνηση όπως και η διακοπή γίνεται μέσα από τις μεθόδους start_simple_sensor() και stop_simple_sensor() για την ασύγχρονη σύνδεση και start_multi_sensor() και stop_multi_sensor() για τη συγχρονισμένη μετάδοση δεδομένων.

7.6.3 Σύνδεση της Φορητής Συσκευής με τον Server

Η κλάση client_initial αναλαμβάνει να κάνει την αρχικοποίηση ενός νέου προβλήματος. Αναλαμβάνει λοιπόν να μεταφέρει από τον εξυπηρετητή τα τρία σύνολα κανόνων και να προσθέσει στο αρχείο info.txt τις απαραίτητες πληροφορίες για το νέο πρόβλημα. Αυτή η κλάση περιέχει τις μεθόδους:

- **connect_receive()** που παίρνει σαν ορίσματα την διεύθυνση του εξυπηρετητή, τον αριθμό της θύρας και το αρχείο στο οποίο θα αποθηκεύσει τα δεδομένα που θα στείλει ο εξυπηρετητής.

  connect_receive(ip_adress,port_number+1,fil+".tmpaxs");

  Δημιουργεί λοιπόν ένα αρχείο με το αναγνωριστικό του προβλήματος και κατάληξη ένα σύνολο έξι χαρακτήρων: tmp, ένα από τα a, b, c ανάλογα με το σύνολο των κανόνων και ένα από τα xs, wc, ws, vc, vs ανάλογα με το είδος του βάρους.

  FileOutputStream fos = new FileOutputStream(file_name);
  BufferedOutputStream bos = new BufferedOutputStream(fos);
  DataOutputStream dos = new DataOutputStream(bos);

  Με την χρήση ενός socket αποθηκεύει στο αρχείο τις τιμές που στέλνει ο εξυπηρετητής.

  Δηλώσεις
  BufferedOutputStream bos ;
  DataOutputStream dos ;
  String tmp;
  Κώδικας
  tmp=dis.readDouble();
  dos.writeDouble(tmp);
  dos.flush();

- **connect_receive_line()** δέχεται ως ορίσματα την διεύθυνση του εξυπηρετητή, τον αριθμό της θύρας και το αρχείο στο οποίο θα αποθηκεύσει τις πληροφορίες που θα στείλει ο εξυπηρετητής, δηλαδή το info.txt.

  connect_receive_line(ip_adress,port_number+16,info);

Ανοίγει το αρχείο info.txt και προσθέτει τις πληροφορίες που του στέλνει ο εξυπηρετητής. Τις πληροφορίες τις διαβάζει με readLine() και προσθέτει ένα κενό, ώστε να υπάρχει διαχωρισμός. Όπως θα επισημανθεί και παρακάτω για να λειτουργήσει η readLine() πρέπει οι πληροφορίες να στέλνονται συνοδευόμενες από '\n' για να γίνεται σαφής ο διαχωρισμός ανάμεσα στις πληροφορίες.

Δηλώσεις

```
BufferedInputStream bis ;
BufferedReader dis;
String down;
String tmp;
```

Κώδικας

```
bis = new BufferedInputStream(client.getInputStream());
dis = new BufferedReader(new InputStreamReader(bis));
down = dis.readLine();
tmp=tmp+down+" ";
tmp=tmp+dis.readLine();
```

- **connect_receive_lineII()** είναι ταυτόσημη με την **connect_receive_line()** η λειτουργία της είναι να πάρει την πρώτη πληροφορία από τον εξυπηρετητή που είναι το όνομα του προβλήματος για να χρησιμοποιηθεί στις υπόλοιπες μεταδόσεις.
- **isEqual()** που παίρνει ως ορίσματα το αρχείο info.txt και το αναγνωριστικό του προβλήματος, εάν δεν υπάρχει το όνομα αυτό στο αρχείο τότε προστίθενται οι πληροφορίες στο αρχείο. Έτσι διασφαλίζεται ότι δεν θα υπάρχουν στο info.txt πολλαπλές εγγραφές.

Η client_initial καλεί με την εξής διάταξη τις παραπάνω συναρτήσεις:

```
info=path+"info.txt";
connect_receive_lineII(ip_adress,port_number+16,info);
fil = path+new_name;
connect_receive(ip_adress,port_number+1,fil+".tmpaxs");
connect_receive(ip_adress,port_number+2,fil+".tmpawc");
connect_receive(ip_adress,port_number+3,fil+".tmpaws");
connect_receive(ip_adress,port_number+4,fil+".tmpavc");
connect_receive(ip_adress,port_number+5,fil+".tmpavs");
connect_receive(ip_adress,port_number+6,fil+".tmpbxs");
connect_receive(ip_adress,port_number+7,fil+".tmpbwc");
connect_receive(ip_adress,port_number+8,fil+".tmpbws");
connect_receive(ip_adress,port_number+9,fil+".tmpbvc");
connect_receive(ip_adress,port_number+10,fil+".tmpbvs");
connect_receive(ip_adress,port_number+11,fil+".tmpcxs");
connect_receive(ip_adress,port_number+12,fil+".tmpcwc");
connect_receive(ip_adress,port_number+13,fil+".tmpcws");
connect_receive(ip_adress,port_number+14,fil+".tmpcvc");
connect_receive(ip_adress,port_number+15,fil+".tmpcvs");
if(isEqual(new_name,info))
```

```
{
        connect_receive_line(ip_adress,port_number+16,info);
}
```

Η μεταβλητή new_name περιέχει το όνομα του νέου προβλήματος Η μεταβλητή path περιέχει τη διαδρομή στην οποία βρίσκεται το αρχείο που πρόκειται να προσπελαστεί.

Η κλάση client_checking διαφέρει με την προηγούμενη στο ότι δεν παίρνει πληροφορίες από τον εξυπηρετητή για κάποιο νέο πρόβλημα, αλλά στέλνει ότι πληροφορίες έχει συγκεντρώσει περί αυτού. Αντί λοιπόν των μεθόδων connect_receive_line() και connect_receive_lineII() υπάρχει η μέθοδος **connect_send()** που δέχεται ως ορίσματα την διεύθυνση του εξυπηρετητή, τον αριθμό της θύρας και το αρχείο το οποίο θα στείλει. Το αρχείο φέρει το όνομα του προβλήματος ταξινόμησης από το οποίο προέκυψε καθώς και την κατάληξη .dat. Μετά την μετάδοση του, διαγράφεται ώστε το PDA να μπορεί να μπορεί να συλλέξει νέες εισόδους και ο εξυπηρετητής την επόμενη φορά να μην χρειάζεται να χειρίζεται παλαιά δεδομένα. Η client_checking καλεί με την εξής διάταξη τις συναρτήσεις:

```
connect_send(ip_adress,port_number,fil+".dat");
connect_receive(ip_adress,port_number+1,fil+".tmpaxs");
connect_receive(ip_adress,port_number+2,fil+".tmpawc");
connect_receive(ip_adress,port_number+3,fil+".tmpaws");
connect_receive(ip_adress,port_number+4,fil+".tmpavc");
connect_receive(ip_adress,port_number+5,fil+".tmpavs");
connect_receive(ip_adress,port_number+6,fil+".tmpbxs");
connect_receive(ip_adress,port_number+7,fil+".tmpbwc");
connect_receive(ip_adress,port_number+8,fil+".tmpbws");
connect_receive(ip_adress,port_number+9,fil+".tmpbvc");
connect_receive(ip_adress,port_number+10,fil+".tmpbvs");
connect_receive(ip_adress,port_number+11,fil+".tmpcxs");
connect_receive(ip_adress,port_number+12,fil+".tmpcwc");
connect_receive(ip_adress,port_number+13,fil+".tmpcws");
connect_receive(ip_adress,port_number+14,fil+".tmpcvc");
connect_receive(ip_adress,port_number+15,fil+".tmpcvs");
```

Κλείνοντας αξίζει να επισημανθεί το γεγονός ότι αν για κάποιο λόγο διακοπεί η σύνδεση ανάμεσα στο PDA και τον εξυπηρετητή, τότε από την κατάληξη των αρχείων δεν αφαιρείται το πρόθεμα tmp υποδεικνύοντας έτσι ότι υπήρξε μη κανονική διακοπή στη σύνδεση. Όσα αρχεία δημιουργήθηκαν με κατάληξη tmp παραμένουν μέχρι να υπάρξει επιτυχής ενημέρωση-αρχικοποίηση οπότε και διαγράφονται.

7.6.4 Η Εσφαλμένη Υλοποίηση των Sockets στην Πλατφόρμα Jeode

Αν ανατρέξει κάποιος τις εντολές που παρουσιάστηκαν στην αρχή της παραγράφου θα παρατηρήσει ότι η while loop είναι ατέρμονη αφού περιλαμβάνει τη συνθήκη while(true). Προγραμματιστικά ένας ατέρμονος βρόγχος σημαίνει ότι εκτελείται συνεχώς για όσο χρόνο η εφαρμογή εκτελείται από την Java Virtual Machine. Εν προκειμένου αυτό σημαίνει ότι το serverSocket που ανοίγουμε είναι έτοιμο να δέχεται clientSocket (να κάνει accept) συνεχώς για όσο εκτελείται η

εφαρμογή (σε οποιαδήποτε κατάσταση λειτουργίας και αν βρισκόμαστε). Βέβαια το να δέχεται socket δεν σημαίνει ότι δέχεται και δεδομένα από τον αισθητήρα αφού αυτό καθορίζεται από το:

```
if(m_chk==true)
        {
                echo=new EchoThread(clientSocket,this,inp);
                echo.start();
        }
```

δηλαδή από τη μεταβλητή m_chk η οποία εξαρτάται από τις επιλογές του χρήστη. Το σημείο αυτό αποτελεί αδυναμία του κώδικα αλλά δεν οφείλεται σε προγραμματιστική αστοχία αλλά σε πρόβλημα της JVM, στην συνέχεια εξηγείται γιατί συμβαίνει αυτό.

Η σχέση client-server ανάμεσα σε αισθητήρες και PDA υλοποιείται δημιουργώντας ένα serverSocket στο PDA το οποίο δέχεται αιτήσεις από τους αισθητήρες, δηλαδή δέχεται συνδέσεις από socket. Η λογική υλοποίηση σε κώδικα επιβάλει ότι όταν ένα serverSocket δεν είναι πλέον απαραίτητο να πρέπει να κλείνει, εφόσον προηγουμένως έχουν κλείσει σωστά όλες οι συνδέσεις. Αυτή η υλοποίηση ενώ πραγματοποιήθηκε και είχε τα αναμενόμενα αποτελέσματα σε επίπεδο localhost, δηλαδή όσο το πείραμα γινόταν ανάμεσα σε αισθητήρες και server στον ίδιο υπολογιστή, όταν δοκιμάστηκε ανάμεσα σε φορητό υπολογιστή και σε PDA δεν λειτούργησε. Αυτό που έγινε είναι ότι ενώ έκλεινε τελείως η EVM (δηλαδή η Jeode Virtual Machine) και μετά την εκκινούσαμε κανονικά, ήταν αδύνατο για τους αισθητήρες να επικοινωνήσουν με την εφαρμογή στο PDA. Ενώ το ping έδειχνε ότι υπάρχει η IP address του PDA, ο αισθητήρας συμπεριφερόταν σαν να άνοιγε socket στο πουθενά και για αυτό εγειρόταν exception ConnectException. Η συμπεριφορά αυτή βέβαια είχε και συμπτώματα τα οποία παρουσιάζονταν μόλις γινόταν προσπάθεια serversocket.close() οπότε δημιουργούταν εξαίρεση SocketException στην εντολή serverSocket.accept(). Αυτό στην αρχή ερμηνεύθηκε ως προγραμματιστική αδυναμία έτσι δοκιμάστηκαν όλες οι δυνατές περιπτώσεις κλεισίματος τόσο του serverSocket όσο και των socket τόσο σε επίπεδο PDA όσο και σε επίπεδο αισθητήρων.

Συγκεκριμένα έγιναν οι εξής προσπάθειες:

- μέσα στην catch της SocketException που πέταγε η serverSocket.accept() να κλείσουμε όλες τις υπάρχουσες συνδέσεις (socket, serversocket).
- για να διασφαλίσουμε ότι όντως κλείναμε όλες τις συνδέσεις έγινε υλοποίηση χρησιμοποιώντας την κλάση ThreadGroup. Ορίζαμε λοιπόν κάθε νέο νήμα ως μέλος του ThreadGroup και μετά ανατρέχαμε σε αυτό και κλείναμε όλα τα μέλη του, με τη βοήθεια των μεθόδων active() και enumerate().
- χρησιμοποιήθηκε η μέθοδος setSoTimeOut() τόσο στα socket όσο και στο serverSocket. Η μέθοδος αυτή αν εφαρμοστεί σε socket θέτει χρονικό περιορισμό στο διάστημα στο οποίο θα περιμένει ένα socket για να διαβάσει από τον προορισμό. Αν εφαρμοστεί σε serverSocket τότε θέτει χρονικό περιορισμό στην accept() για το πόσο θα περιμένει την σύνδεση ενός socket. Αν περάσει το χρονικό αυτό όριο τότε εγείρεται εξαίρεση SocketTimeoutException.
- εξετάστηκε η περίπτωση τα νήματα που δημιουργούνται να είναι daemons, όποτε λογικά η JVM δεν μπορεί να τερματίσει (αυτό έγινε με τη μέθοδο isDaemon() της Thread). Ωστόσο τα νήματα δεν γίνονταν daemon, δηλαδή δεν συνεχίζονταν στο παρασκήνιο (background).

Το αποτέλεσμα για κάθε μία από τις παραπάνω περιπτώσεις καθώς και συνδυασμούς αυτών ήταν ένα. Η πλατφόρμα ενώ δούλευε για όλες τις περιπτώσεις σε επίπεδο localhost, στο PDA κατέρρεε σε ανεπανόρθωτο σημείο. Δεν μπορούσαν οι αισθητήρες να πετύχουν σύνδεση με το PDA αν για μία φορά έκλεινε το serverSocket, το παράθυρο ή το πρόγραμμα. Μοναδική λύση ήταν το hard reset της συσκευής.

Μία καθοριστική προσπάθεια για να αποφασίσουμε για τη φύση του προβλήματος ήταν όταν η πλατφόρμα δοκιμάστηκε σε δικτυακό επίπεδο, τόσο ασύρματο όσο και ενσύρματο, ανάμεσα σε PC με εγκαταστημένο JDK 1.4. Σε αυτή την περίπτωση η πλατφόρμα δούλευε χωρίς κανένα πρόβλημα. Σημειώνουμε ότι δεν υπάρχει περίπτωση χρήσης κλάσης ή μεθόδου που δεν υποστηρίζει η Jeode αφενός γιατί ο προγραμματισμός έγινε σύμφωνα με το δικό της specification (PersonalJava 1.2) αφετέρου διότι η εικονική μηχανή εγείρει exceptions αν χρησιμοποιούνται κλάσεις ή μέθοδοι που δεν υποστηρίζει (παράδειγμα ClassNotFoundException στη μέθοδο parseDouble()).

Η επόμενη κίνηση ήταν η αναζήτηση πληροφοριών στο Internet όπου συναντήσαμε πολλά γεγονότα αντίστοιχα προς το δικό μας, χωρίς όμως να προτείνεται λύση. Δυστυχώς ελλιπής είναι και η δυνατότητα χρησιμοποίησης εργαλείων στο PDA για να εξεταστεί το ποιες θύρες παραμένουν ανοικτές. Σημειώνεται ότι δεν εγείρεται Bind Exception που να δηλώνει ότι γίνεται προσπάθεια δημιουργίας ServerSocket σε ήδη υπάρχον port.

Σαν συμπέρασμα θεωρούμε ότι η Jeode εικονική μηχανή παρουσιάζει αδυναμία αντιμετώπισης της συγκεκριμένης κατάστασης, καταλήγοντας σε απρόβλεπτες καταστάσεις. Από την πλευρά μας, έχουμε να προτείνουμε τις εξής λύσεις:

- διατήρηση μόνιμης σύνδεσης ανάμεσα σε PDA και αισθητήρα, όπως και υλοποιήθηκε. Έτσι εξηγείται και ο χρησιμοποιούμενος ατέρμονος βρόχος. Αυτή η λύση δεν ξεπερνά το πρόβλημα αλλά το περιορίζει στην περίπτωση που κλείνουμε τελείως την JVM οπότε και πρέπει να γίνει hard reset. Διασφαλίζει τη λογική που έχει αναπτυχθεί συνολικά για την πλατφόρμα και δεν δημιουργεί προβλήματα στην κανονική λειτουργία της πλατφόρμας.
- αντιστροφή των ρόλων των serverSocket και socket. Το ρόλο του server θα παίξει ο αισθητήρας και του client το PDA. Η υλοποίηση αυτή ενέχει προβλήματα. Το κυριότερο είναι ότι αντικρούει την υπόθεση περί έλλειψης ευφυΐας του αισθητήρα αφού πλέον ένας αισθητήρας εμφανίζει αυξημένη πολυπλοκότητα.
- χρήση των Datagram Socket ή UDP socket [§ 6.7.4.1]. Σε αυτή την περίπτωση όμως αναφερόμαστε σε λογική ταχυδρομείου όπου δεν ξέρει ο αποστολέας αν και πότε φτάνει το μήνυμα, κάτι το οποίο είναι σημαντικό στην περίπτωση της συγχρονισμένης μετάδοσης.

Το βασικότερο ερώτημα που έμεινε όμως αναπάντητο είναι η εγκυρότητα του επιχειρήματος της Sun περί του ότι η εικονική μηχανή μπορεί να καταρρέει αλλά η ίδια μηχανή πάνω στην οποία τρέχει ποτέ.

Τέλος παρουσιάζονται τα μηνύματα που καταγράφονται από την πλευρά ενός αισθητήρα όταν δημιουργείται το πρόβλημα. Οι boolean τιμές μετά τους αριθμούς 1, 2, 3 είναι οι τιμές που προκύπτουν στις μεθόδους isBound(), isClosed(), isConnected() της κλάσης Socket αντίστοιχα.

4.4X java.net.SocketTimeoutException: Read timed out
at java.net.SocketInputStream.socketRead0(Native Method)
1:true
2:true
3:true

Εδώ κλείνει η εφαρμογή στο PDA

4.4X ? java.net.ConnectException: Connection refused: connect
at java.net.PlainSocketImpl.socketConnect(Native Method)
1:false
2:true
3:false

Εδώ διακόπτεται η λειτουργία του αισθητήρα

STOP
java.net.SocketException: Socket operation on nonsocket: connect
at java.net.PlainSocketImpl.socketConnect(Native Method)

Ενώ εδώ ξαναρχίζει, αφού ξαναρχίσει πρώτα η εφαρμογή στο PDA

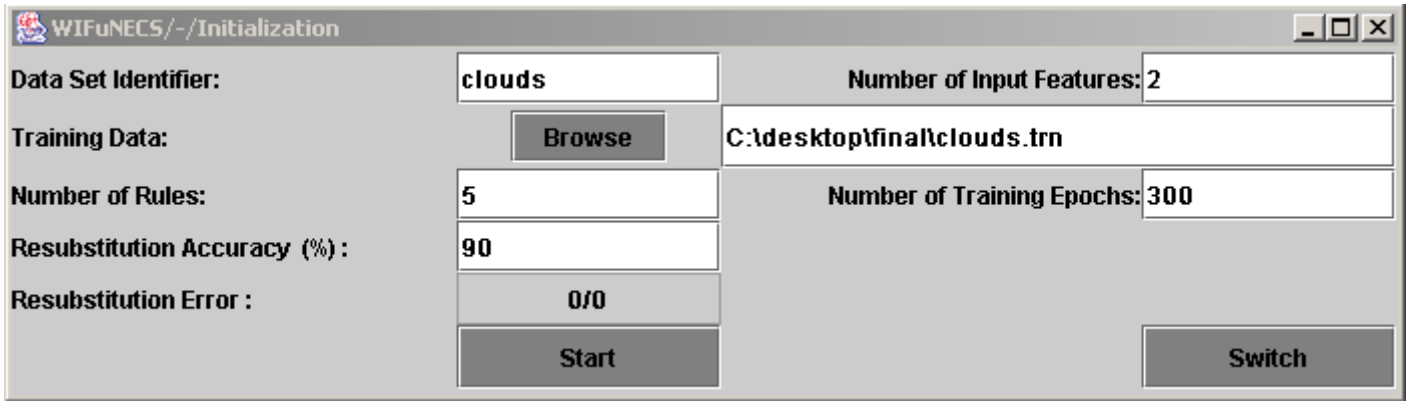
START
1:false
2:true
3:false
java.net.ConnectException: Connection refused: connect
at java.net.PlainSocketImpl.socketConnect(Native Method)

7.7 Server

Ο εξυπηρετητής είναι υπεύθυνος για την αρχικοποίηση του PDA, ώστε αυτό να μπορεί να αντιμετωπίσει ένα νέο πρόβλημα ταξινόμησης, είναι επίσης επιφορτισμένος με την συνεχή ενημέρωση των βαρών του ταξινομητή με τιμές που προκύπτουν από την υβριδική επανεκπαίδευση του συστήματος. Η υλοποίηση έγινε σε JDK 1.4 και η εφαρμογή εγκαταστάθηκε σε PC. Η σύνδεση με το PDA στηρίζεται στο πρωτόκολλο 802.11 [§ 5.2] το οποίο παρέχει πλήρη υποστήριξη του προτύπου TCP/IP. Έτσι το PDA μπορεί να συνδεθεί σε ένα εξυπηρετητή που ανήκει στο τοπικό ασύρματο δίκτυο ή σε έναν εξυπηρετητή που βρίσκεται σε ένα οποιαδήποτε σημείο του κόσμου, αρκεί να είναι γνωστή η IP διεύθυνση αυτού. Κάθε εξυπηρετητής είναι υπεύθυνος για ένα μόνο πρόβλημα και μπορεί να βρεθεί σε δύο καταστάσεις. Στην κατάσταση αρχικοποίησης και στην κατάσταση ενημέρωσης.

7.7.1 Φάση Αρχικοποίησης

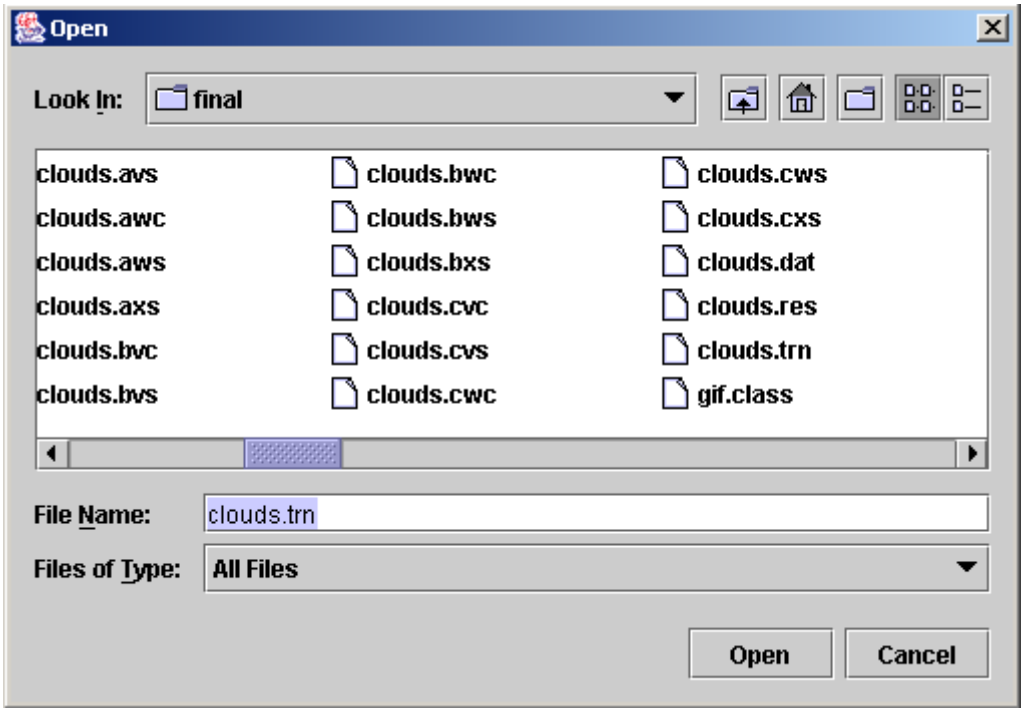
Στο σχήμα παρουσιάζεται η εφαρμογή σε κατάσταση αρχικοποίησης.



Σχήμα 7.7.1.1 Εξυπηρετητής σε κατάσταση αρχικοποίησης

Για να αρχικοποιηθεί το σύστημα ως προς ένα πρόβλημα ταξινόμησης πρέπει ο χρήστης να ορίσει κάποιες παραμέτρους.

- Data Set Identifier: ένα όνομα που χαρακτηρίζει το πρόβλημα (π.χ. iris, clouds, ionosphere)
- Training Data: το training set, το σύνολο των δειγμάτων που γνωρίζουμε τις κατηγορίες τους. Το κουμπί Browse είναι συνδεδεμένο με αντικείμενο της κλάσης JFileChooser η οποία μας δίνει τη δυνατότητα να έχουμε οπτική αναπαράσταση του συστήματος αρχείων ενός υπολογιστικού συστήματος



Σχήμα 7.7.1.2 Το παράθυρο που προκύπτει αν πατηθεί το κουμπί Browse

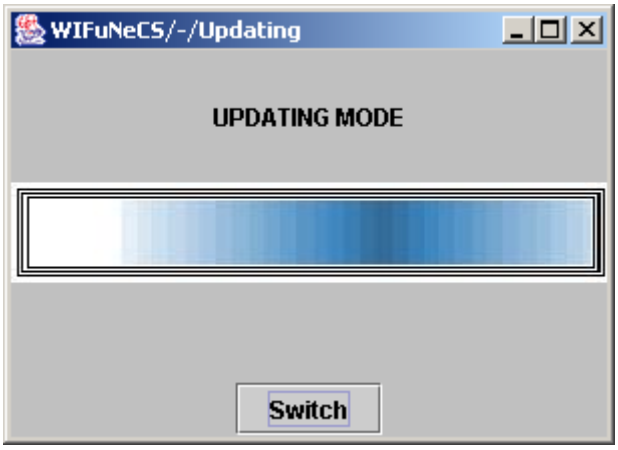
- Number of Input Features: ο αριθμός των στοιχείων των χαρακτηριστικών
- Number of Rules: ο αριθμός των κανόνων, δηλαδή ο αριθμός των κόμβων του κρυμμένου στρώματος του SuPFuNIS
- Number of Training Epochs: ο αριθμός των εποχών εκπαίδευσης
- Resubstitution Accuracy: η ακρίβεια υποκατάστασης

Μόλις δοθούν λοιπόν τιμές σε όλα τα πεδία τότε η αρχικοποίηση μπορεί να ξεκινήσει. Ανάλογα με τις τιμές που έδωσε ο χρήστης και ανάλογα με τις υπολογιστικές δυνατότητες του συστήματος, το χρονικό διάστημα στο οποίο πραγματοποιείται η εκπαίδευση-αρχικοποίηση μπορεί να διαρκέσει από μερικά λεπτά έως αρκετές ώρες.

7.7.2 Φάση Ενημέρωσης

Εφόσον λοιπόν αρχικοποιηθεί ένας εξυπηρετητής πάνω σε ένα πρόβλημα τότε πατώντας το κουμπί Switch, μεταφερόμαστε σε κατάσταση ενημέρωσης. Στην κατάσταση αυτή ο εξυπηρετητής αναλαμβάνει το ρόλο να εξυπηρετεί κάθε αίτηση που προκύπτει από ένα PDA είτε για αρχικοποίηση είτε για ενημέρωση βαρών. Επειδή ακριβώς χρησιμοποιείται η λογική πελάτη-εξυπηρετητή, ο εξυπηρετητής μπορεί να εξυπηρετήσει πολλούς πελάτες ακολουθιακά. Στην κατάσταση αυτή ο server ανοίγει τα port 6000 έως 6016 και περιμένει να αποκριθεί σε τυχόν αιτήσεις από πελάτες.

Στο σχήμα παρουσιάζεται η εφαρμογή σε κατάσταση αρχικοποίησης.



Σχήμα 7.7.2.1 Εξυπηρετητής σε κατάσταση ενημέρωσης

Είναι προφανές ότι υπάρχουν δύο είδη αιτημάτων.

7.7.2.1 Αιτήματα Αρχικοποίησης

Οι πελάτες που κάνουν αίτηση για αρχικοποίηση θέλουν να αποκτήσουν τα απαραίτητα αρχεία για να μπορούν να αντιμετωπίζουν το πρόβλημα ταξινόμησης που υποστηρίζει ο εξυπηρετητής. Τα απαραίτητα αρχεία χωρίζονται σε τρία σύνολα βαρών (a, b, c), όπου κάθε σύνολο αποτελείται από 5 αρχεία βαρών. Τα τρία σύνολα βαρών αντιστοιχούν σε διαφορετικά σύνολα κανόνων, σε βέλτιστο αριθμό κανόνων, σε υποβέλτιστο αριθμό κανόνων και σε αριθμό κανόνων μεγαλύτερο από τον βέλτιστο.

Έχουμε δηλαδή τα αρχεία:

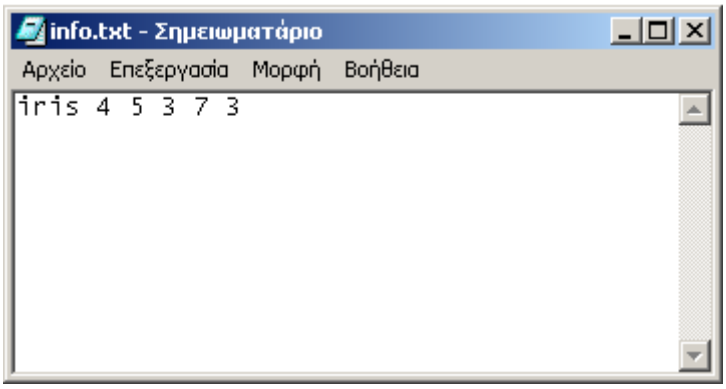
αριθμός θύρας - όνομα βάρους		
6001	axs	βάρη κανόνων a
6002	awc	
6003	aws	
6004	avc	
6005	avs	
6006	bxs	βάρη κανόνων b
6007	bwc	
6008	bws	
6009	bvc	
6010	bvs	
6011	cxs	βάρη κανόνων c
6012	cwc	
6013	cws	
6014	cvc	
6015	cvs	

Σχήμα 7.7.2.1.1 Αντιστοιχία βαρών -θυρών

Επίσης είναι απαραίτητο να προστεθεί στο αρχείο info.txt που βρίσκεται στο PDA μία γραμμή που να περιέχει τις βασικές πληροφορίες για το καινούργιο πρόβλημα ταξινόμησης. Οι πληροφορίες αυτές είναι τα Data Set Identifier, Number

of Input Features, Number of Rules. Από το τελευταίο νούμερο προκύπτουν οι αριθμοί και των υπόλοιπων κανόνων (± 40%). Είναι απαραίτητος επίσης ο αριθμός των εξόδων.

Στο σχήμα φαίνεται παράδειγμα αρχείου info.txt



Σχήμα 7.7.2.1.2 Παράδειγμα info.txt

Καταλαβαίνουμε ότι το πρόβλημα iris έχει 4 εισόδους, το πρώτο σύνολο βαρών αναφέρεται σε 5 κανόνες, το δεύτερο σε 3 και το τρίτο σε παράλληλα οι έξοδοι είναι 3.

Έτσι οι θύρες από 6001 έως 6015 χρησιμοποιούνται για να μεταδίδουν τα βάρη με την αντιστοιχία που φαίνεται στο σχήμα (7.7.2.1.1). Η θύρα 6016 χρησιμοποιείται για την αποστολή του αρχείου spec.nfo που περιέχει τις απαραίτητες πληροφορίες για αρχικοποίηση.

7.7.2.2 Αιτήματα Ενημέρωσης

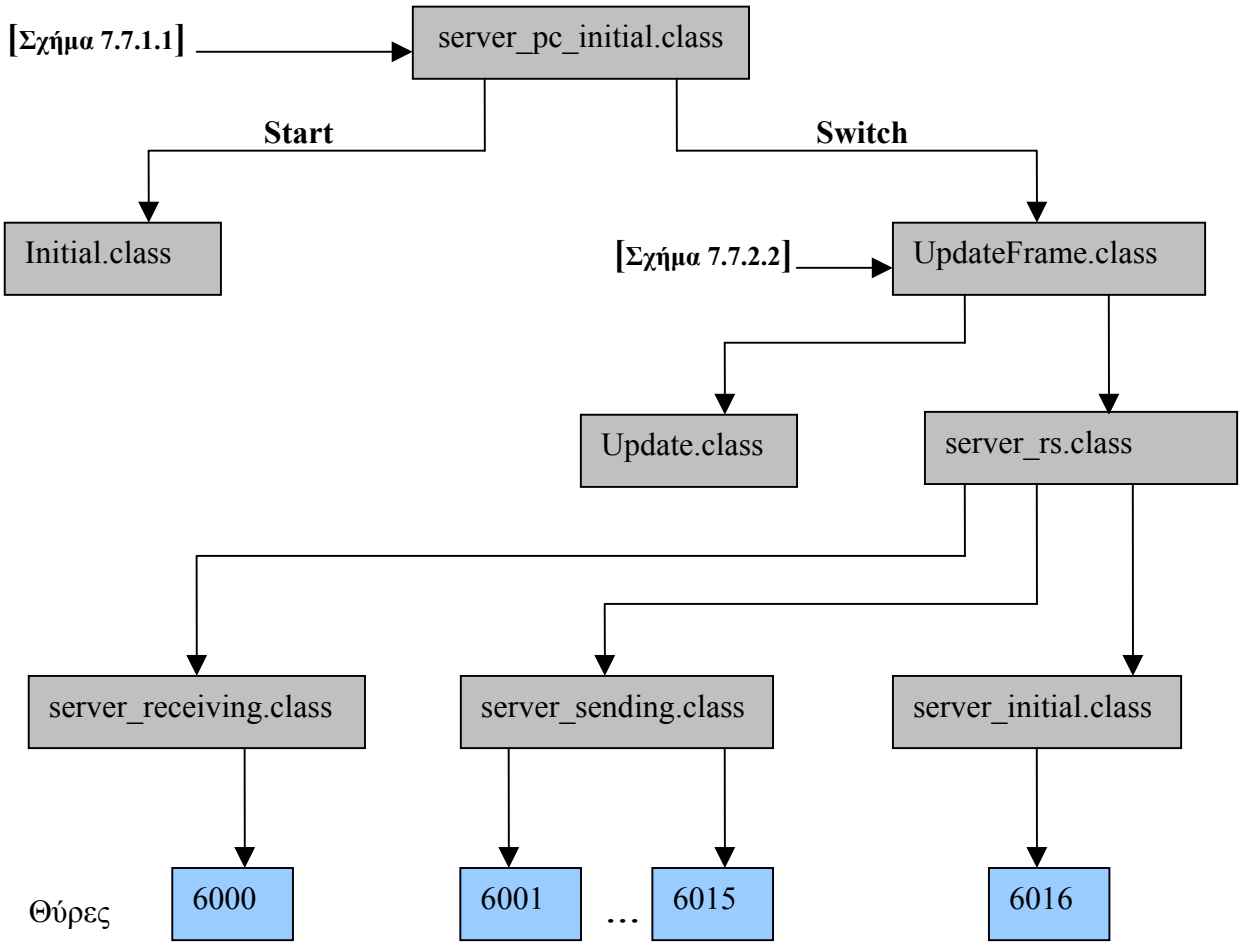
Είναι απόλυτα λογικό, τα βάρη που έχουν δημιουργηθεί κατά τη φάση της εκπαίδευσης μετά από κάποιο χρονικό διάστημα να μην είναι τόσο αποδοτικά όσο κάποιο άλλο σύνολο βαρών που έχει προκύψει από επανεκπαίδευση. Τα νέα αυτά βάρη εμφανίζουν υψηλότερη ακρίβεια υποκατάστασης, οπότε ενδείκνυται να υπάρξει σύνδεση του πελάτη με τον εξυπηρετητή για ενημέρωση βαρών .

Η ενημέρωση δεν διαφέρει σε κάτι από την αρχικοποίηση όσον αφορά το θέμα των βαρών. Η διαφορά έγκειται στο ότι αντί ο πελάτης να πάρει ένα αρχείο spec.nfo, τώρα στέλνει ένα αρχείο που έχει όνομα το όνομα του προβλήματος και κατάληξη .dat. Το αρχείο αυτό περιέχει όλα τα δείγματα που έχει συλλέξει μέχρι εκείνη τη στιγμή το PDA (ο πελάτης) και βάσει αυτών θα γίνει επανεκπαίδευση στον εξυπηρετητή. Αποτελεί την γνώση που έχει αποκτήσει το PDA μέσα από την διαδικασία ταξινόμησης, η οποία πρέπει να μεταφερθεί στο σύστημα εκπαίδευσης για να μετουσιωθεί σε αποδοτικότερες πράξεις ταξινόμησης.

Αντί λοιπόν της θύρας 6016, στην ενημέρωση χρησιμοποιείται η θύρα 6000 για να μεταδοθεί το αρχείο με την κατάληξη .dat στο server. Το αρχείο μετά την μετάδοση διαγράφεται, αφού ο server δεν θα χρειαστεί ξανά αυτά τα δεδομένα, από την στιγμή που τα έχει ήδη επεξεργαστεί.

7.7.3 Η Εφαρμογή Java του Server

Παρακάτω παρουσιάζεται ένας κατευθυνόμενος ακυκλικός γράφος για να απεικονιστεί επακριβώς η διάταξη και λειτουργία των κλάσεων από τις οποίες αποτελείται ο εξυπηρετητής.



Σχήμα 7.7.3.1 Δενδροδιάγραμμα κλάσεων εξυπηρετητή

Η κλάση server_pc_initial.class είναι η αρχική κλάση του εξυπηρετητή που δημιουργεί το πλαίσιο (frame) του σχήματος 7.7.1.1 και ανάλογα με τις επιλογές του χρήστη εκτελεί διάφορες ενέργειες. Με Start δημιουργείται αντικείμενο της κλάσης Initial.class (και εκκινείται το αντίστοιχο νήμα) ενώ με Switch έχουμε την δημιουργία τριών αντικειμένων:

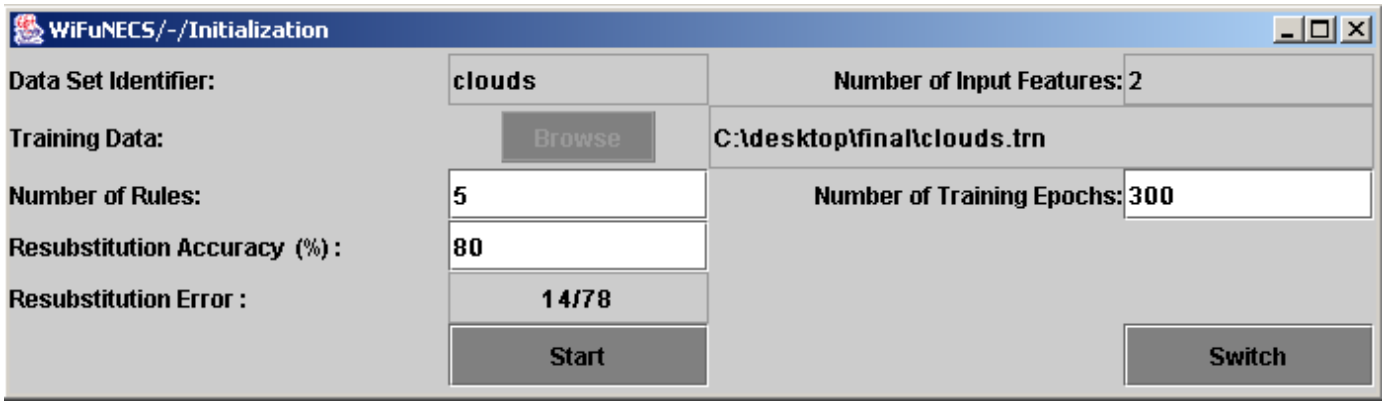
- ένα αντικείμενο της κλάσης UpdateFrame.class που δημιουργεί το πλαίσιο του σχήματος 7.7.2.1
- ένα αντικείμενο της κλάσης server_rs.class που είναι υπεύθυνη για την διασύνδεση του εξυπηρετητή με τους πελάτες
- ένα αντικείμενο της κλάσης Update

Η κλάση **Update** υλοποιεί την επανεκπαίδευση του ταξινομητή σε ένα συγκεκριμένο πρόβλημα για λογαριασμό του server και είναι κατά κάποιο τρόπο συμπληρωματική της Initial. Η μέθοδος **run()** ελέγχοντας ορισμένες συνθήκες εκκινεί την επανεκπαίδευση του συστήματος ταξινόμησης αυτόματα, εφόσον βέβαια κρίνει ότι

ικανοποιούνται ορισμένα κριτήρια. Οι συνθήκες που ελέγχει η κλάση Update είναι το κατά πόσο από κάποια φορητή συσκευή έχουν αποσταλεί καινούργια πρότυπα επανεκπαίδευσης (δηλαδή το κατά πόσο υπάρχει ένα καινούργιο .dat αρχείο). Εάν κάτι τέτοιο συμβαίνει αρχίζει η επανεκπαίδευση η οποία επιστρέφει όταν τερματίσει κανονικά ο αλγόριθμος μάθησης, όταν γίνει Switch ή όταν εμφανιστούν νέα πρότυπα επανεκπαίδευσης. Σε κάθε περίπτωση η καλύτερη δυνατή εκτίμηση των παραμέτρων του ταξινομητή μέχρι εκείνη την χρονική στιγμή αποθηκεύεται στα αντίστοιχα byte αρχεία.

Αξίζει να σημειωθεί ότι δημιουργούνται νήματα για κάθε αντικείμενο ώστε να δίνεται η δυνατότητα στον εξυπηρετητή να λειτουργεί ως εφαρμογή, δηλαδή να περιμένει για αιτήσεις από πελάτες. Επίσης νήμα είναι και το πλαίσιο UpdateFrame.class για να μπορεί ο χρήστης να επιστρέψει σε κατάσταση αρχικοποίησης πατώντας το κουμπί Switch του πλαισίου. Έτσι ο χρήστης μπορεί να αλλάξει ορισμένα στοιχεία από την διαδικασία αρχικοποίησης. Τα στοιχεία αυτά είναι ο αριθμός των κανόνων (και των τριών συνόλων), ο αριθμός των εποχών και η ακρίβεια υποκατάστασης. Τα υπόλοιπα στοιχεία δεν μπορεί να τα αλλάξει, γι' αυτό και είναι απενεργοποιημένα. Αυτό συμβαίνει από την στιγμή που έχει υπάρξει η σύμβαση ότι *κάθε εξυπηρετητής μπορεί να αντιμετωπίσει ΜΟΝΟ ένα πρόβλημα.*

Στο παρακάτω σχήμα φαίνονται τα πεδία που είναι απενεργοποιημένα.



Σχήμα 7.7.3.2 Ο εξυπηρετητής εκ νέου σε κατάσταση αρχικοποίησης.

Η κλάση server_rs.class δημιουργεί τα νήματα που παρακολουθούν τις θύρες από 6000 έως 6016 και απαντούν στις αντίστοιχες αιτήσεις.

Η κλάση server_receiving.class είναι υπεύθυνη για τη θύρα 6000 και συγκεκριμένα υποστηρίζει τις επόμενες ενέργειες:

Δημιουργεί serverSocket στη θύρα 6000 και περιμένει για κάποιο socket οπότε το δέχεται με accept(). Στη συνέχεια δημιουργεί αρχείο με όνομα temp, το όνομα του προβλήματος και κατάληξη .dat.

```
FileOutputStream fos = new FileOutputStream("temp"+file_name);
BufferedOutputStream bos = new BufferedOutputStream(fos);
DataOutputStream dos = new DataOutputStream(bos);
```

Στο αρχείο αυτό αποθηκεύει τα δεδομένα που του στέλνει ο πελάτης.

Δηλώσεις

```
String tmp;
BufferedInputStream bis;
DataInputStream dis;
Socket client;
```

Κώδικας

```
bis = new BufferedInputStream(client.getInputStream());
dis = new DataInputStream(bis);
tmp=dis.readDouble();
dos.writeDouble(tmp);
 dos.flush();
```

Μόλις τελειώσει η μετάδοση, οπότε και εγείρεται εξαίρεση EOFException, το αρχείο παίρνει το πρόθεμα new, το όνομα του προβλήματος και κατάληξη .dat, ενώ ξεκινά η διαδικασία επανεκπαίδευσης (κλάση Update.class). Εάν μια σύνδεση διακοπεί την στιγμή που γινόταν μεταφορά του αρχείου δεδομένων, τότε το αρχείο με πρόθεμα "temp" διαγράφεται, ώστε να μην υπάρχουν απομεινάρια στον εξυπηρετητή.

```
if(new File("temp"+file_name).exists())
                new File("temp"+file_name).delete();
```

Η κλάση server_sending.class είναι υπεύθυνη για τις θύρες 6001 έως 6015 και υποστηρίζει τις επόμενες ενέργειες.

Στην ουσία υπάρχει ένα αντικείμενο της κλάσης για κάθε θύρα. Κάθε ένα αντικείμενο λοιπόν δημιουργεί ένα serverSocket στην θύρα που του αντιστοιχεί και περιμένει για κάποιο socket, οπότε το δέχεται με accept(). Στην συνέχεια ανοίγει το αντίστοιχο αρχείο βάρους που πρόκειται να μεταδώσει:

```
FileInputStream fis = new FileInputStream(file_name);
BufferedInputStream bis = new BufferedInputStream(fis);
DataInputStream dis = new DataInputStream(bis);
```

Οπότε ξεκινά την μετάδοση:

Δηλώσεις

```
String tmp;
BufferedOutputStream bos;
DataOutputStream dos;
```

Κώδικας

```
bos = new BufferedOutputStream(client.getOutputStream());
dos = new DataOutputStream(bos);
tmp=String.valueOf(dis.readDouble());
dos.writeDouble(Double.parseDouble(tmp));
dos.flush();
```

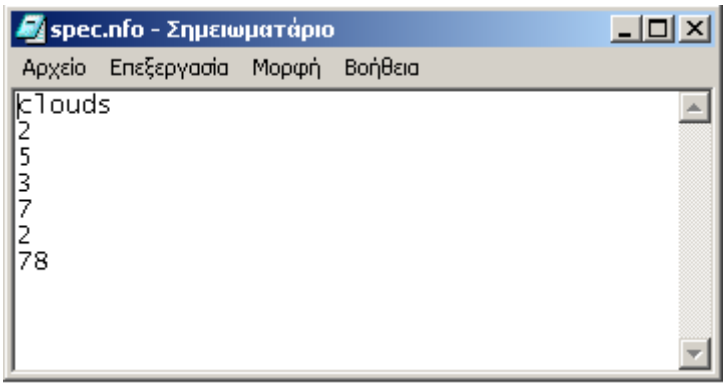
Μόλις τελειώσει η μετάδοση των στοιχείων του αρχείου τότε εγείρεται εξαίρεση EOFException και τελειώνει η σύνδεση, ενώ το αντικείμενο αποδεσμεύει τους πόρους.

Εδώ πρέπει να αναφερθούμε στη φύση των αρχείων και στον τρόπο με τον οποίο διαβάζουν τα αντικείμενα από αυτά. Τα αρχεία είναι διαμορφωμένα σε σειρές byte, όπου κάθε σειρά των 8 byte αντιστοιχεί σε ένα double (αριθμό κινητής υποδιαστολής), ενώ ενδιάμεσα στις σειρές δεν παρεμβάλλονται κενά. Η ανάγνωση γίνεται χρησιμοποιώντας την μέθοδο readDouble() της DataInputStream ενώ η εγγραφή γίνεται με τη μέθοδο writeDouble() της DataOutputStream.

Η κλάση server_sending.class είναι υπεύθυνη για τη θύρα 6016 και για τη μεταφορά από τον εξυπηρετητή στον πελάτη, του αρχείου spec.nfo που περιέχει τις πληροφορίες για αρχικοποίηση ενός πελάτη πάνω σε ένα πρόβλημα.

Το αρχείο spec.nfo όπως φαίνεται και στο παρακάτω σχήμα περιέχει σε κάθε γραμμή του τα εξής:

1. το όνομα του προβλήματος για το οποίο γίνεται αρχικοποίηση
2. τον αριθμό των εισόδων
3. τον αριθμό των κανόνων του πρώτου
4. του δεύτερου και
5. του τρίτου συνόλου κανόνων
6. τον αριθμό των εξόδων και τέλος
7. τον αριθμό των προτύπων του συνόλου εκπαίδευσης, το οποίο βέβαια δεν χρειάζεται να μεταδοθεί σαν πληροφορία.



Σχήμα 7.7.3.3 Παράδειγμα αρχείου spec.nfo

Τώρα η ανάγνωση γίνεται με τη μέθοδο readLine() ενώ προστίθεται ο χαρακτήρας '\n' σε κάθε γραμμή. Αυτή είναι μια σύμβαση μεταξύ εξυπηρετητή-πελάτη για να μπορεί να καταλαβαίνει ο πελάτης τις πληροφορίες που του στέλνει ο εξυπηρετητής.

Τελειώνοντας αναφέρουμε ότι πατώντας Switch στο πλαίσιο του σχήματος 7.7.2.1 αφενός επιστρέφουμε σε κατάσταση αρχικοποίησης, αφετέρου κλείνουν όλες οι συνδέσεις όπως και τα serverSocket από το 6000 έως το 6016, χρησιμοποιώντας τη μέθοδο close(). Αυτό είναι καίριας σημασίας επειδή ακριβώς δεν πρέπει ένας πελάτης να μπορεί να εξυπηρετηθεί εάν πραγματοποιούνται δομικές αλλαγές στο αντιμετωπιζόμενο πρόβλημα ταξινόμησης.

Κεφάλαιο 8

Πειραματικές Μετρήσεις

8 Πειραματικές Μετρήσεις

8.1 Περιγραφή Πειραμάτων

Το αντικείμενο της παρούσας διπλωματικής εργασίας στρέφεται γύρω από το πεδίο των νευρο-ασαφών συστημάτων ταξινόμησης. Σε επίπεδο λειτουργίας καθώς και σε επίπεδο προγραμματιστικής υλοποίησης το WiDIFuNeCS αναλύθηκε εκτενώς στο [§ 7]. Επιπλέον οι χρησιμοποιούμενες προγραμματιστικές πλατφόρμες Java εμφανίζονται στο [§ 6], ενώ το χρησιμοποιούμενο hardware παρουσιάζεται στο [Π 4]. Σύμφωνα με όσα αναφέρθηκαν σε προηγούμενα κεφάλαια, βάση για την ανάπτυξη του WiDIFuNeCS αποτέλεσε το SuPFuNIS. Το νευρο-ασαφές αυτό σύστημα χρησιμοποιήθηκε κατά κύριο λόγο ως ταξινομητής, ενώ τελικός στόχος ήταν η ασύρματη κατανεμημένη λειτουργία αυτού σε πραγματικό περιβάλλον.

Όπως ίσως να έχει διαφανεί από προηγούμενα κεφάλαια, βαρύτητα έχει δοθεί επίσης σε θέματα που σχετίζονται με την εκπαίδευση του νευρο-ασαφούς δικτύου καθώς και σε θέματα που αφορούν την βελτίωση των επιδόσεων του ταξινομητή. Κατά συνέπεια μελετήθηκε ο αλγόριθμος εκπαίδευσης του συστήματος ταξινόμησης όπως επίσης εξετάστηκε η αύξηση των επιδόσεων ταξινόμησης με την εισαγωγή της έννοιας του πολυεπίπεδου ταξινομητή. Παράλληλα προτάθηκε μια τροποποίηση πάνω σε μια υφιστάμενη μέθοδο εκτίμησης της αξιοπιστίας του αποτελέσματος ταξινόμησης. Τέλος στην προσπάθεια να ικανοποιηθεί το αίτημα για μη επιβλεπόμενη επανεκπαίδευση του ταξινομητή, αίτημα το οποίο απορρέει τόσο από την χρησιμοποιούμενη αρχιτεκτονική όσο και από τους πραγματικούς περιορισμούς της υλοποίησης, προτείνεται ένας καινούργιος αλγόριθμος υβριδικής επανεκπαίδευσης.

Ορισμένοι από αυτούς τους αλγορίθμους έχουν αναλυθεί και επαληθευτεί πλήρως άλλοι όμως, όπως για παράδειγμα η προτεινόμενη υβριδική επανεκπαίδευση, δεν έχουν μελετηθεί καθόλου (τουλάχιστον στην ακριβή μορφή τους όπως αυτή περιγράφεται στην παρούσα διπλωματική εργασία).

Ο έλεγχος των αλγορίθμων εκπαίδευσης-επανεκπαίδευσης, της μεθόδου υπολογισμού της αξιοπιστίας και της αρχιτεκτονικής του πολυεπίπεδου ταξινομητή θα μπορούσε να γίνει χρησιμοποιώντας το WiDIFuNeCS σε πραγματικές συνθήκες. Μια τέτοια επιλογή αφενός μεν δεν παρέχει τις εγγυήσεις που απαιτούνται για την επαλήθευση της ορθότητας τους, αφετέρου δε περιορίζει το εύρος των πειραματικών επαληθεύσεων που μπορούν να επιτευχθούν. Η αρνητική αυτή συνέπεια υφίσταται από την στιγμή που στο WiDIFuNeCS χρησιμοποιούμενοι αλγόριθμοι, μέθοδοι και αρχιτεκτονικές είναι άρρηκτα συνδεδεμένες μεταξύ τους. Έτσι γίνεται επιτακτική η ανάγκη για χρήση της λογικής της προσομοίωσης αφού καθίσταται δυνατόν σε πλήρως ελεγχόμενο περιβάλλον να μελετηθούν τα επιμέρους στοιχεία του συστήματος, ανεπηρέαστα από τις πιθανές μεταξύ τους διασυνδέσεις.

Παρόλα αυτά δόθηκε ιδιαίτερη μέριμνα στον τρόπο οργάνωσης των επιτελούμενων προσομοιώσεων έτσι ώστε να αντικατοπτρίζουν σε μεγάλο βαθμό την λειτουργία του πραγματικού συστήματος. Συνακόλουθα οι παράμετροι που χρησιμοποιήθηκαν ήταν πανομοιότυπες με αυτές του WiDIFuNeCS κατά την πραγματική του λειτουργία. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για τις προσομοιώσεις ήταν η Java, η επεξεργασία των μετρήσεων έγινε σε Matlab ενώ οι πίνακες των αποτελεσμάτων και οι αντίστοιχες γραφικές παραστάσεις έγιναν με την χρήση Matlab και Excel.

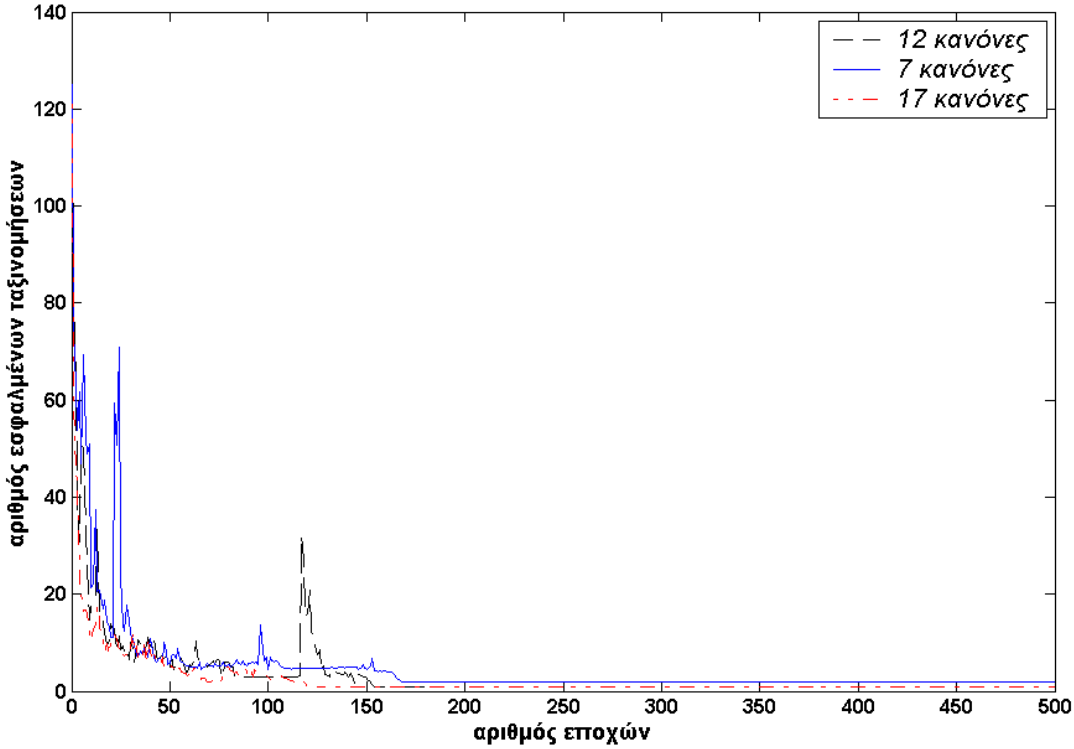
Τα σύνολα προτύπων που χρησιμοποιούνται είναι τα ionosphere, clouds και iris [Π 2]. Για κάθε ένα από αυτά τα προβλήματα έγιναν τρία πειράματα μέσω προσομοιώσεων.

- Αρχικά τα σύνολα προτύπων ανακατεύονταν και στην συνέχεια χωρίζονταν σε δύο υποσύνολα με τυχαίο τρόπο. Το πρώτο περιελάμβανε 60% των προτύπων ενώ το δεύτερο το υπολειπόμενο 40%.
- Κατά την φάση της αρχικής εκπαίδευσης του συστήματος οι τιμές του ρυθμού εκπαίδευσης και του όρου ορμής μειώνονταν γραμμικά μεταξύ των τιμών 0.1 και 0.0001, πράγμα που έρχεται σε συμφωνία με τα όσα περιγράφονται στο [§ 3.2.2]. Επιπλέον ο μέγιστος αριθμός εποχών εκπαίδευσης ορίστηκε να είναι 500, τιμή η οποία επελέγη μετά από ορισμένα προπαρασκευαστικά πειράματα. Σε αυτή την φάση γίνεται χρήση μεγαλύτερου υποσυνόλου προτύπων (60%).
- Σε αντίθεση με τα clouds και ionosphere, όπου το κριτήριο τερματισμού της εκπαίδευσης ήταν η ικανοποίηση του ορίου των 500 εποχών ή η επίτευξη μηδενικού αριθμού λανθασμένων ταξινομήσεων, στο iris πρόβλημα το κριτήριο τερματισμού της εκπαίδευσης ήταν η ικανοποίηση του ορίου των 500 εποχών ανεξαρτήτως του αριθμού των λανθασμένων ταξινομήσεων.
- Ο αριθμός των κανόνων του απλού ταξινομητή επελέγη ακολουθώντας το [§ 4.7]. Ο αριθμός κανόνων του τμήματος του επίπεδου ταξινομητή με μεγαλύτερο αριθμό κανόνων από τον βέλτιστο ορίστηκε να είναι 40% μεγαλύτερος του βέλτιστου. Αυτό έγινε για να κρατηθεί η υπολογιστική πολυπλοκότητα κάτω από ορισμένα όρια αλλά και για να αποφευχθούν φαινόμενα υπερεκπαίδευσης. Από την άλλη πλευρά το τρίτο τμήμα του πολυεπίπεδου ταξινομητή επελέγη να έχει 60% του βέλτιστου αριθμού κανόνων αφού μικρότερο ποσοστό οδηγούσε σε εκφυλιστικές καταστάσεις (για παράδειγμα ταξινομητής με ένα ή κανένα κανόνα).
- Τέλος στην φάση της υβριδικής επανεκπαίδευσης [§ 7.3] ο αριθμός εποχών ορίστηκε στις 100. Επιπλέον ανά πρότυπο επανεκπαίδευσης γίνονται τέσσερις προσπάθειες ανανέωσης των βαρών αρχίζοντας από τιμές η και α ίσες με 0.1. Σε αυτή την φάση γίνεται χρήση του μικρότερου υποσυνόλου προτύπων (40%).

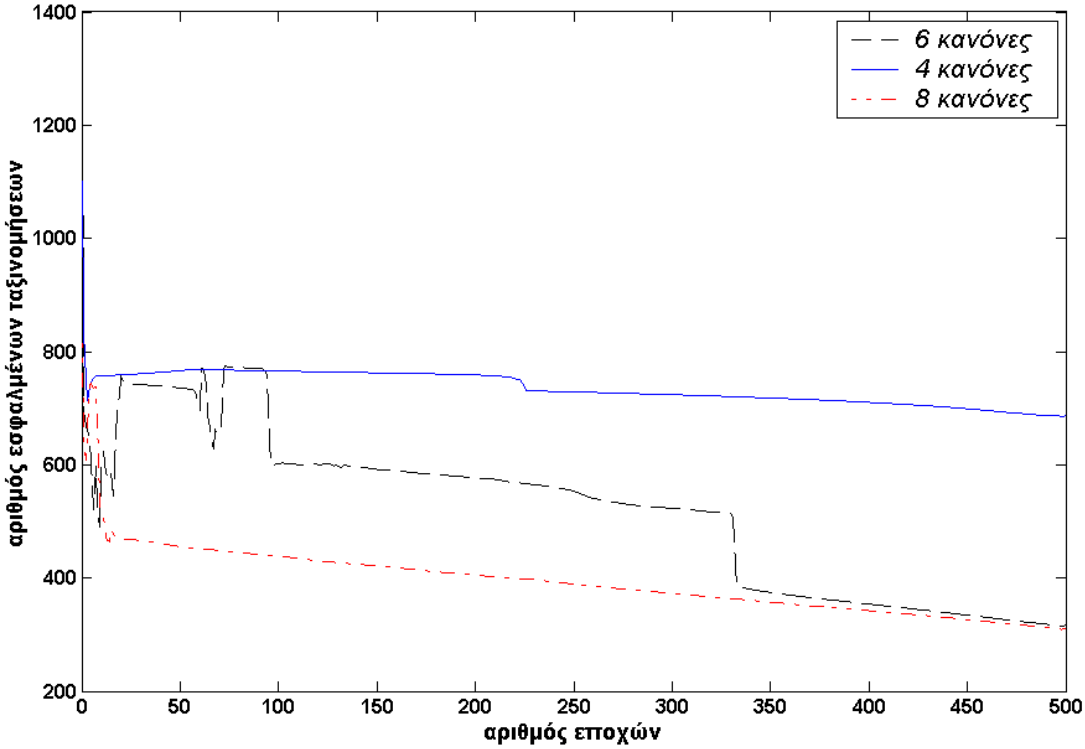
Σε αυτό το σημείο αξίζει να σημειωθεί ότι οι γραφικές παραστάσεις που παρουσιάζονται στην συνέχεια αφορούν τα αποτελέσματα ενός πειράματος προσομοίωσης ανά πρόβλημα (από τα τρία που έχουν γίνει). Ακολουθήθηκε αυτή η λύση γιατί σε αντίθετη περίπτωση ο αριθμός των γραφημάτων θα ήταν πολύ μεγάλος (συγκεκριμένα 36) χωρίς να προσθέτει κάτι το ουσιαστικό στις επισημάνσεις-παρατηρήσεις που γίνονται. Επίσης στο [Π 3] υπάρχουν όλες οι παράμετροι του συστήματος από τα δεύτερα πειράματα των iris, ionosphere καθώς και οι παράμετροι από το τρίτο πείραμα που αφορά το σύνολο clouds.

8.2 Αλγόριθμος Εκπαίδευσης

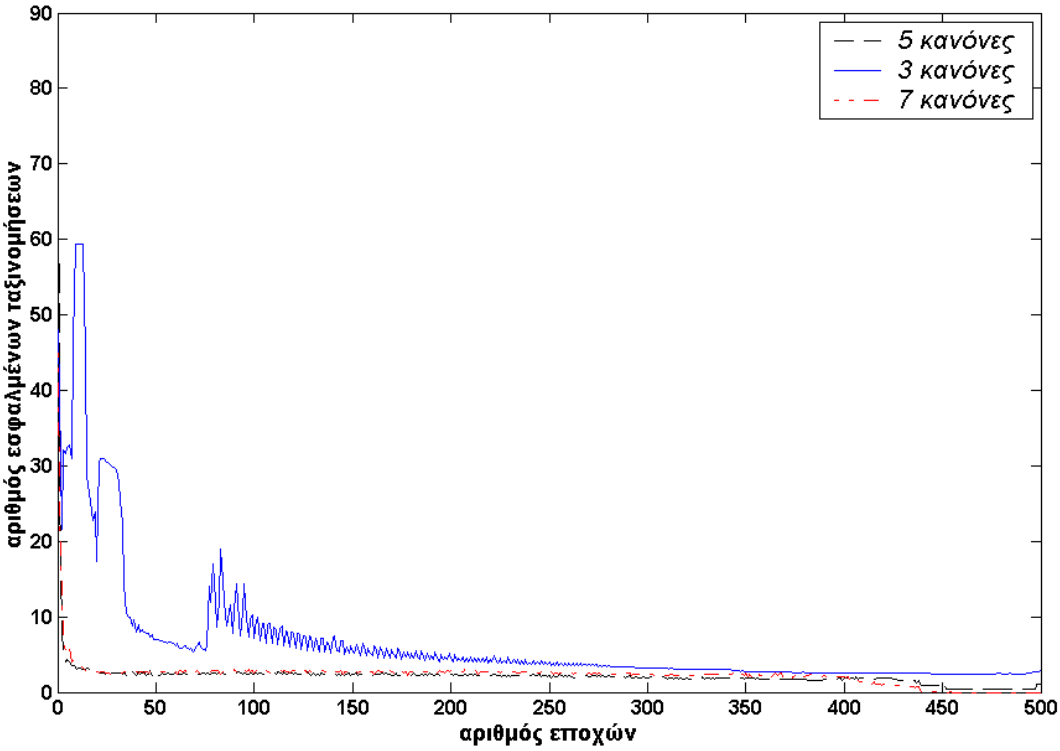
Επαναλαμβάνοντας τα όσα έχουν γραφεί σε διάφορα σημεία της παρούσας διπλωματικής εργασίας, αναφέρουμε ότι ο χρησιμοποιούμενος αλγόριθμος επιβλεπόμενης μάθησης είναι ο backpropagation με την μέθοδο κλίσης [§ 3.2.1] και [§ 4.4]. Ο μοναδική τροποποίηση-βελτίωση που γίνεται είναι αυτή που περιγράφεται στο [§ 4.8]. Ωστόσο η αλλαγή αυτή δεν σχετίζεται με τον αλγόριθμο αυτόν καθ' εαυτό αλλά αφορά στην διόρθωση μιας μικρής σχεδιαστική αστοχίας.



Σχήμα 8.2.1: Τροχιές εκπαίδευσης για το σύνολο ionosphere.



Σχήμα 8.2.2: Τροχιές εκπαίδευσης για το σύνολο clouds.



Σχήμα 8.2.3: Τροχιές εκπαίδευσης για το σύνολο iris.

Τα σύνολα προτύπων επί των οποίων υπολογίζεται ο εσφαλμένος αριθμός ταξινομήσεων περιλαμβάνουν 211 πρότυπα για το ionosphere, 3000 πρότυπα για το clouds και 90 πρότυπα για το iris. Έχοντας κατά νου αυτούς τους αριθμούς και παρατηρώντας τις γραφικές παραστάσεις κανείς μπορεί να παρατηρήσει ότι η απόδοση του αλγορίθμου ανάστροφής διάδοσης παραμένει υψηλή ανεξαρτήτως της φύσης των δεδομένων που συνιστούν το σύνολο εκπαίδευσης και προφανώς ανεξαρτήτως του μεγέθους του. Συνακόλουθα κυρίως εκ του αποτελέσματος, συνάγεται το συμπέρασμα ότι η τροποποίηση του αλγορίθμου λειτουργεί ικανοποιητικά παρέχοντας ευστάθεια στον αλγόριθμο.

Η εκλογή του αριθμού των 500 εποχών εκπαίδευσης αποδεικνύεται επιτυχής μιας και σε ορισμένες περιπτώσεις (clouds data set), η σύγκλιση του αλγορίθμου και η εύρεση ενός ελαχίστου στην συνάρτηση σφάλματος καθυστερεί μέχρι περίπου τις 330 εποχές. Από την άλλη πλευρά οι αρχικές τιμές που επελέγησαν για τα η και α καθώς και η γραμμική τους μείωση παρέχουν την δυνατότητα στον στοχαστικό αλγόριθμο backpropagation να ξεφεύγει από τοπικά ελάχιστα, γεγονός που αποτυπώνεται στην τροχιά εκπαίδευσης των 6 κανόνων στο πρόβλημα clouds.

Επιπρόσθετα αξίζει να σημειωθεί η ιδιαίτερα καλή συμπεριφορά του αλγορίθμου στα προβλήματα iris και ionosphere, με αποτέλεσμα κατά την διάρκεια των τελευταίων εποχών εκπαίδευσης να εμφανίζουν λιγότερες από 5 λανθασμένες ταξινομήσεις και στα τρία διαφορετικά σύνολα από κανόνες.

Μια σημαντική παρατήρηση που πρέπει να γίνει αναφορικά με τις παραπάνω γραφικές παραστάσεις είναι ότι ο αριθμός εσφαλμένων ταξινομήσεων που απεικονίζεται αντιστοιχεί στον μέσο αριθμό λανθασμένων ταξινομήσεων που λαμβάνουν χώρα κατά την διάρκεια της εκάστοτε εποχής εκπαίδευσης.

8.3 Αλγόριθμος Επανεκπαίδευσης

Ο αλγόριθμος που προτείνεται στην παρούσα διπλωματική εργασία είναι αυτός της υβριδικής μάθησης όπως περιγράφεται στο [§ 7.3]. Ακολουθώντας την περιγραφή του αλγορίθμου κανείς παρατηρεί ότι στο αρχικό του στάδιο περιλαμβάνει εκτέλεση του αλγορίθμου backpropagation (επιβλεπόμενη μάθηση) ενώ στην συνέχεια αρχίζει η εκτέλεση του αλγορίθμου αυτού καθ' εαυτού. Έτσι ο αλγόριθμος ανάστροφης διάδοσης εφαρμόζεται στο υποσύνολο με το 60% των συνολικών προτύπων ενώ ο αλγόριθμος υβριδικής μάθησης στο υποσύνολο με το 40% των συνολικών προτύπων.

	Λάθος (5 κανόνες)	Σωστές (5 κανόνες)	Λάθος (3 κανόνες)	Σωστές (3 κανόνες)	Λάθος (7 κανόνες)	Σωστές (7 κανόνες)
Φάση Αρχικοποίησης	58	92	85	65	114	36
Φάση Εκπαίδευσης	4	146	5	145	4	146
Φάση Επανεκπαίδευσης	4	146	5	145	3	147

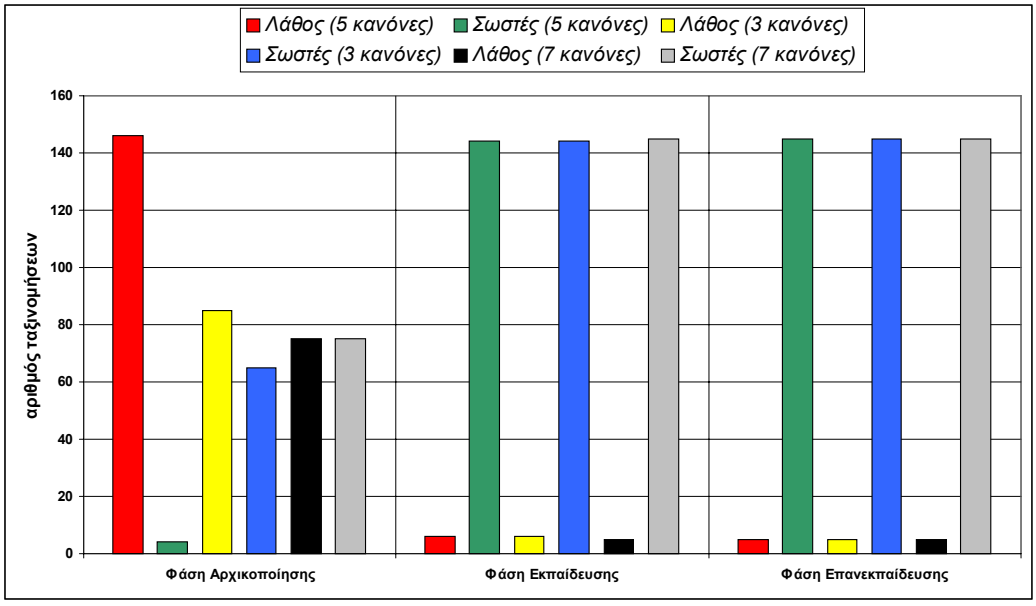
Πίνακας 8.3.1: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο iris data set.

	Λάθος (5 κανόνες)	Σωστές (5 κανόνες)	Λάθος (3 κανόνες)	Σωστές (3 κανόνες)	Λάθος (7 κανόνες)	Σωστές (7 κανόνες)
Φάση Αρχικοποίησης	146	4	85	65	75	75
Φάση Εκπαίδευσης	6	144	6	144	5	145
Φάση Επανεκπαίδευσης	5	145	5	145	5	145

Πίνακας 8.3.2: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο iris data set.

	Λάθος (5 κανόνες)	Σωστές (5 κανόνες)	Λάθος (3 κανόνες)	Σωστές (3 κανόνες)	Λάθος (7 κανόνες)	Σωστές (7 κανόνες)
Φάση Αρχικοποίησης	148	2	99	51	43	107
Φάση Εκπαίδευσης	4	146	4	146	5	145
Φάση Επανεκπαίδευσης	4	146	4	146	5	145

Πίνακας 8.3.3: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο iris data set.



Σχήμα 8.3.1: Επίδοση αλγορίθμων εκπαίδευσης και επανεκπαίδευσης στο iris data set.

	Λάθος (6 κανόνες)	Σωστές (6 κανόνες)	Λάθος (4 κανόνες)	Σωστές (4 κανόνες)	Λάθος (8 κανόνες)	Σωστές (8 κανόνες)
Φάση Αρχικοποίησης	2495	2505	2373	2627	2143	2857
Φάση Εκπαίδευσης	583	4417	897	4103	530	4470
Φάση Επανεκπαίδευσης	573	4427	878	4122	515	4485

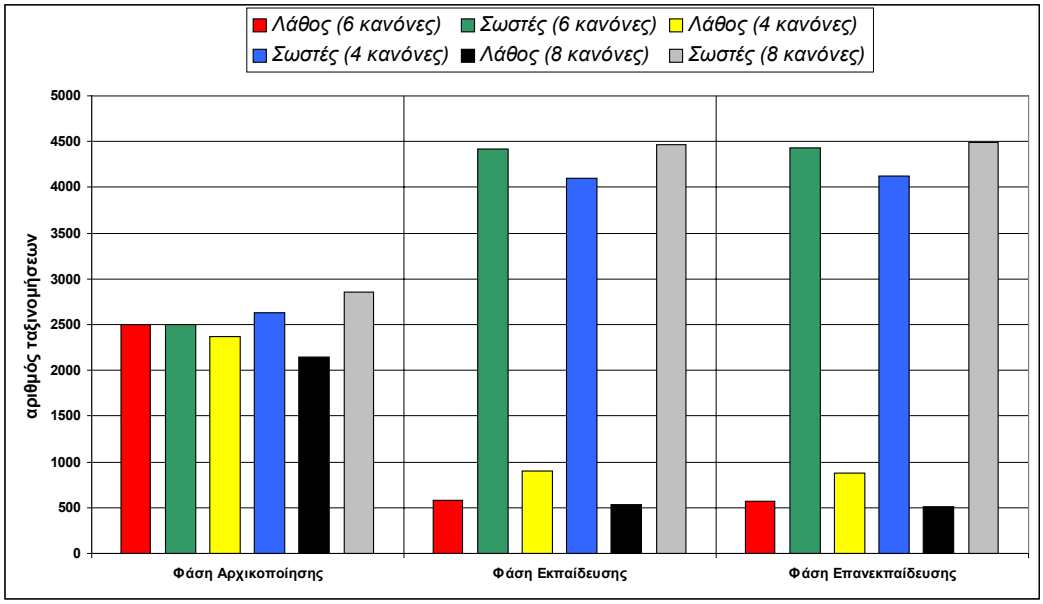
Πίνακας 8.3.4: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο clouds data set.

	Λάθος (6 κανόνες)	Σωστές (6 κανόνες)	Λάθος (4 κανόνες)	Σωστές (4 κανόνες)	Λάθος (8 κανόνες)	Σωστές (8 κανόνες)
Φάση Αρχικοποίησης	2007	2993	2229	2771	2226	2774
Φάση Εκπαίδευσης	798	4202	1217	3783	767	4233
Φάση Επανεκπαίδευσης	772	4228	1020	3980	732	4268

Πίνακας 8.3.5: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο clouds data set.

	Λάθος (6 κανόνες)	Σωστές (6 κανόνες)	Λάθος (4 κανόνες)	Σωστές (4 κανόνες)	Λάθος (8 κανόνες)	Σωστές (8 κανόνες)
Φάση Αρχικοποίησης	2320	2680	1874	3126	1367	3633
Φάση Εκπαίδευσης	539	4461	1136	3864	541	4459
Φάση Επανεκπαίδευσης	520	4480	948	4052	525	4475

Πίνακας 8.3.6: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο clouds data set.



Σχήμα 8.3.2: Επίδοση αλγορίθμων εκπαίδευσης και επανεκπαίδευσης στο clouds data set.

	Λάθος (12 κανόνες)	Σωστές (12 κανόνες)	Λάθος (7 κανόνες)	Σωστές (7 κανόνες)	Λάθος (17 κανόνες)	Σωστές (17 κανόνες)
Φάση Αρχικοποίησης	234	117	148	203	116	235
Φάση Εκπαίδευσης	12	339	126	225	16	335
Φάση Επανεκπαίδευσης	12	339	126	225	16	335

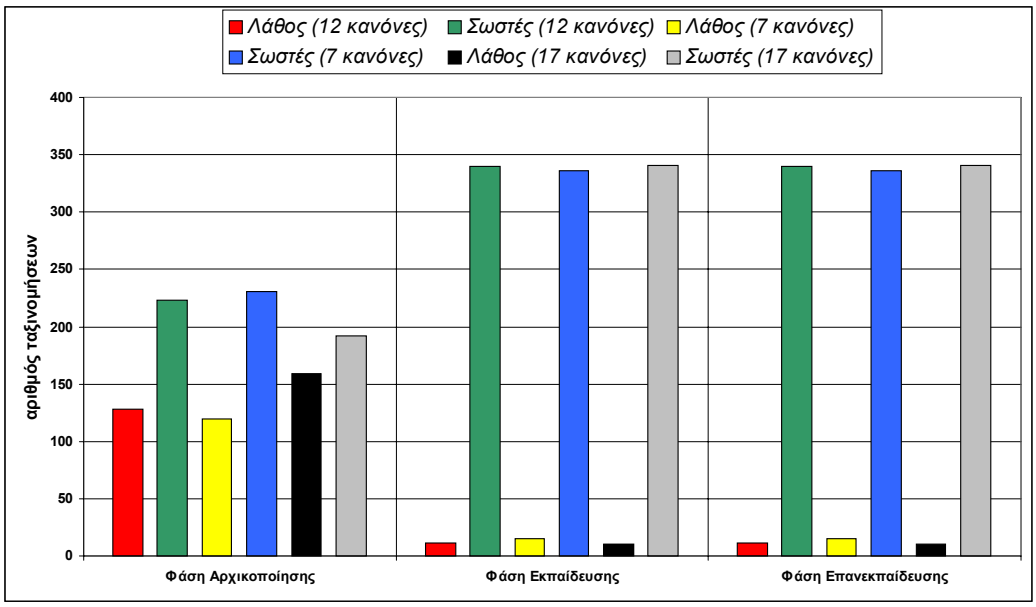
Πίνακας 8.3.7: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο ionosphere data set.

	Λάθος (12 κανόνες)	Σωστές (12 κανόνες)	Λάθος (7 κανόνες)	Σωστές (7 κανόνες)	Λάθος (17 κανόνες)	Σωστές (17 κανόνες)
Φάση Αρχικοποίησης	205	146	215	136	196	155
Φάση Εκπαίδευσης	14	337	12	339	9	342
Φάση Επανεκπαίδευσης	14	337	12	339	9	342

Πίνακας 8.3.8: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο ionosphere data set.

	Λάθος (12 κανόνες)	Σωστές (12 κανόνες)	Λάθος (7 κανόνες)	Σωστές (7 κανόνες)	Λάθος (17 κανόνες)	Σωστές (17 κανόνες)
Φάση Αρχικοποίησης	128	223	120	231	159	192
Φάση Εκπαίδευσης	11	340	15	336	10	341
Φάση Επανεκπαίδευσης	11	340	15	336	10	341

Πίνακας 8.3.9: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο ionosphere data set.



Σχήμα 8.3.3: Επίδοση αλγορίθμων εκπαίδευσης και επανεκπαίδευσης στο ionosphere data set.

Σε πρώτη θεώρηση πρέπει να τονιστεί το γεγονός ότι ο αλγόριθμος επανεκπαίδευσης εκπληρώνει σχεδόν πλήρως τις προδιαγραφές της κατασκευής του. Έτσι παρατηρείται πειραματικά τουλάχιστον η επαλήθευση της ιδέας που υπάρχει πίσω από τον αλγόριθμο υβριδικής μάθησης. Η λογική του αλγορίθμου είναι να μην επηρεαστεί η τιμή της ακρίβειας υποκατάστασης και επιπλέον να δοθεί η δυνατότητα σε εκτιμήσεις των (άγνωστων) κατηγοριών των προτύπων επανεκπαίδευσης για να τροποποιήσουν τα διανύσματα των βαρών με σκοπό την περαιτέρω αύξηση της resubstitution accuracy.

Παρατηρώντας το σύνολο των πινάκων και των γραφημάτων αυτό που ισχύει με βεβαιότητα είναι ότι η τιμή της ακρίβειας υποκατάστασης μεταξύ εκπαίδευσης-επανεκπαίδευσης διατηρείται ή βελτιώνεται. Στο πρόβλημα iris η βελτίωση είναι της τάξης του ενός προτύπου και στο πρόβλημα ionosphere η ακρίβεια υποκατάστασης διατηρείται. Τέλος στο πρόβλημα clouds η μείωση του αριθμού των σφαλμάτων γίνεται σαφώς αισθητή. Λαμβάνοντας υπόψη αυτό καθώς και την τροχιά εκπαίδευσης στο σύνολο clouds (η οποία δεν είχε συγκλίνει πλήρως) ένας μπορεί να βγάλει το συμπέρασμα ότι ο προτεινόμενος αλγόριθμος επανεκπαίδευσης εμφανίζει αυξημένη επίδοση στις περιπτώσεις που η επιβλεπόμενη εκπαίδευση που προηγείται αυτού δεν είναι πλήρης.

8.4 Μέθοδος Υπολογισμού της Αξιοπιστίας

Σε συνδυασμό με την έξοδο κάθε στοιχειώδους νευρο-ασαφούς ταξινομητή υπολογίζεται η τιμή του τροποποιημένου εκτιμητή αξιοπιστίας [§ 3.3.3]. Στον απλό ταξινομητή η γνώση της τιμής του, δίνει την δυνατότητα εκτίμησης της εμπιστοσύνης που αντιστοιχεί στην κατηγοριοποίηση που έχει γίνει. Αντίθετα η τιμή του μέτρου αξιοπιστίας στον πολυεπίπεδο ταξινομητή είναι καίριας σημασίας αφού αποτελεί βασικό στοιχείο της διαδικασίας συγχώνευσης των επιμέρους αποτελεσμάτων στο τελικό αποτέλεσμα.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	50	91	1	0	1	0
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	8	1	2	13	2	13
0.8<Αξιοπιστία	0	0	1	133	1	133

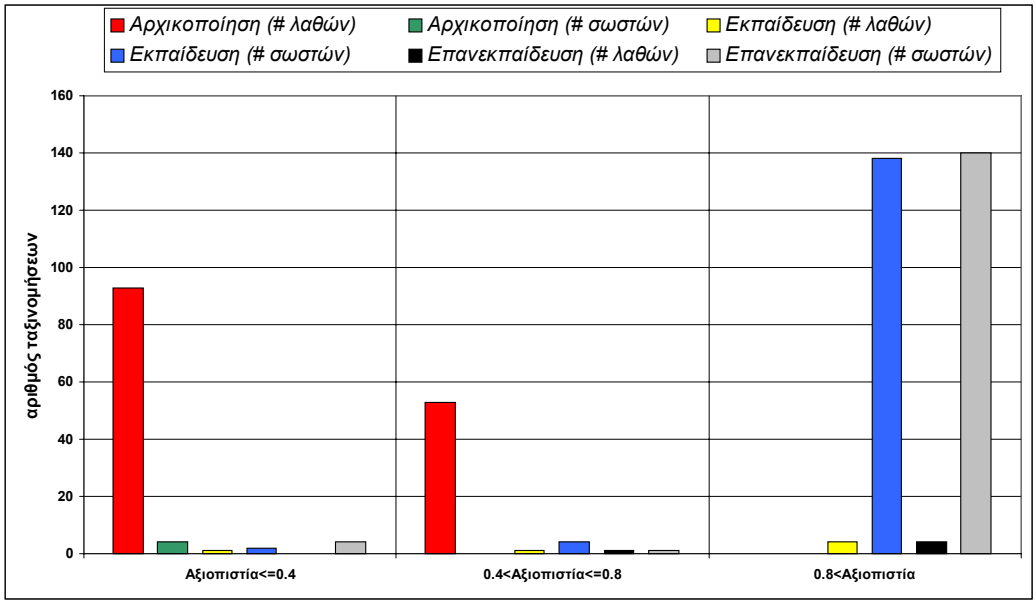
Πίνακας 8.4.1: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο iris data set.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	93	4	1	2	0	4
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	53	0	1	4	1	1
0.8<Αξιοπιστία	0	0	4	138	4	140

Πίνακας 8.4.2: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο iris data set.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	82	2	0	1	0	1
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	66	0	0	0	0	0
0.8<Αξιοπιστία	0	0	4	145	4	145

Πίνακας 8.4.3: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο iris data set.



Σχήμα 8.4.1: Επίδοση του συντελεστή αξιοπιστίας για το iris σύνολο.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	2495	2505	396	849	376	836
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	0	0	133	1240	144	1243
0.8<Αξιοπιστία	0	0	54	2328	53	2348

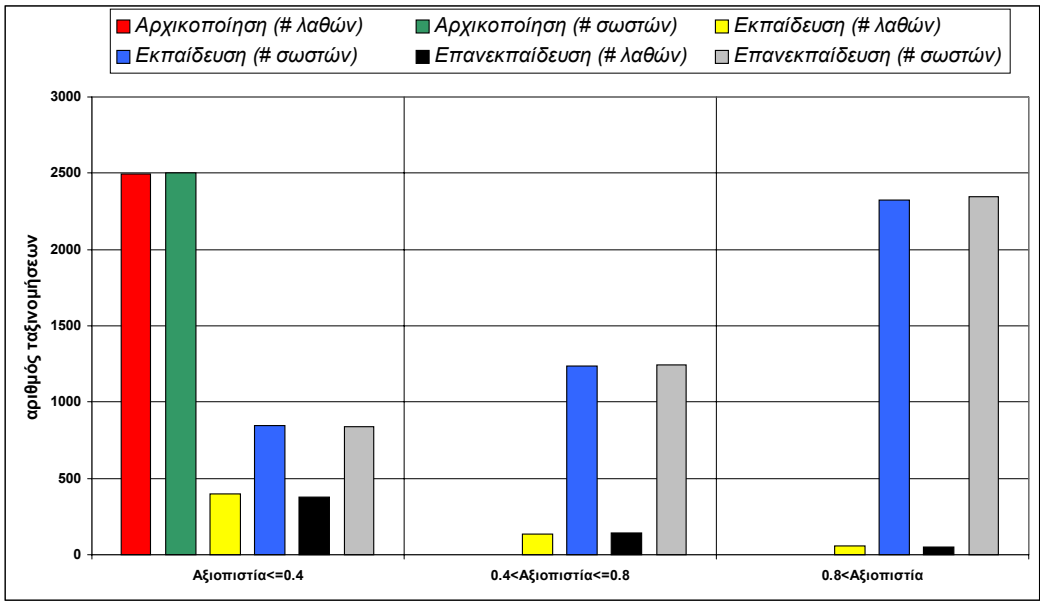
Πίνακας 8.4.4: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο clouds data set.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	2007	2993	662	1038	580	1056
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	0	0	103	1268	133	854
0.8<Αξιοπιστία	0	0	33	1896	59	2318

Πίνακας 8.4.5: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο clouds data set.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	1844	2300	384	809	390	957
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	476	380	113	1120	109	1351
0.8<Αξιοπιστία	0	0	42	2532	21	2172

Πίνακας 8.4.6: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο clouds data set.



Σχήμα 8.4.2: Επίδοση του συντελεστή αξιοπιστίας για το clouds σύνολο.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	224	108	1	8	1	8
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	10	9	5	19	5	19
0.8<Αξιοπιστία	0	0	6	312	6	312

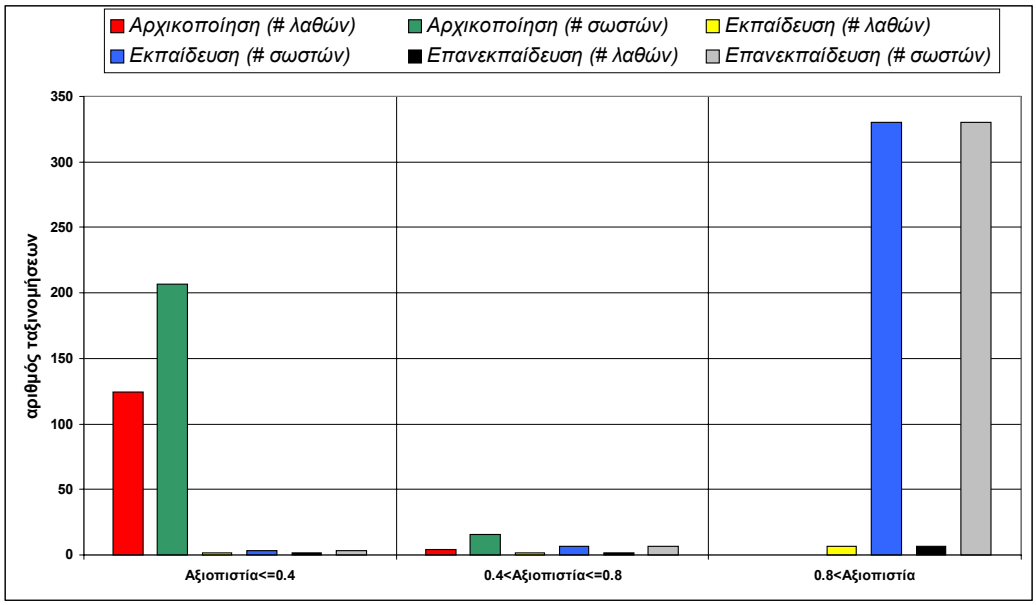
Πίνακας 8.4.7: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο ionosphere data set.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	205	140	7	18	7	18
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	0	6	5	38	5	38
0.8<Αξιοπιστία	0	0	2	281	2	281

Πίνακας 8.4.8: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο ionosphere data set.

	Αρχικοποίηση (# λαθών)	Αρχικοποίηση (# σωστών)	Εκπαίδευση (# λαθών)	Εκπαίδευση (# σωστών)	Επανεκπαίδευση (# λαθών)	Επανεκπαίδευση (# σωστών)
Αξιοπιστία<=0.4	124	207	2	3	2	3
0.4<Αξιοπιστία & Αξιοπιστία <=0.8	4	16	2	7	2	7
0.8<Αξιοπιστία	0	0	7	330	7	330

Πίνακας 8.4.9: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο ionosphere data set.



Σχήμα 8.4.3: Επίδοση του συντελεστή αξιοπιστίας για το ionosphere σύνολο.

Το ζητούμενο από έναν εκτιμητή αξιοπιστίας είναι να λαμβάνει αυξημένες τιμές (εν προκειμένου μεγαλύτερες από 0.8) όταν το συμπέρασμα που εξάγει ο ταξινομητής είναι ορθό, σε αντίθετη περίπτωση ενδείκνυται η προκύπτουσα τιμή αξιοπιστίας να είναι χαμηλή (εν προκειμένου μικρότερη από 0.4). Στα προβλήματα iris και ionosphere κάτι τέτοιο πράγματι συμβαίνει, ανεξαρτήτως της κατάστασης του συστήματος (δηλαδή ανεξαρτήτως του εάν βρίσκεται στις φάσεις της αρχικοποίησης, εκπαίδευσης, επανεκπαίδευσης) τιμές του συντελεστή αξιοπιστίας μεγαλύτερες από 0.8 αντιστοιχούν σε σωστές ταξινομήσεις. Επιπλέον στο πρόβλημα iris τιμές του συντελεστή μικρότερες του 0.4 αντιστοιχούν με βεβαιότητα σε λανθασμένες ταξινομήσεις. Αυτές οι παρατηρήσεις ισχύουν εν μέρει και για το σύνολο clouds, μόνο που σε αυτή την περίπτωση παραβιάζεται το φράγμα του 0.8 όσον αφορά στις επιτυχείς ταξινομήσεις.

Μια ιδιαίτερα σημαντική παρατήρηση που μπορεί να γίνει είναι ότι εάν ο ταξινομητής δεν έχει περάσει από τα στάδια της εκπαίδευσης και της επανεκπαίδευσης τότε εμφανίζει σχεδόν αποκλειστικά χαμηλές τιμές αξιοπιστίας. Έτσι παρέχεται η δυνατότητα εξετάζοντας μόνο της τιμές της αξιοπιστίας να αποφανθεί κάποιος περί του εάν ένας (εν γένει άγνωστος ταξινομητής) είναι εκπαιδευμένος για το πρόβλημα ταξινόμησης που αντιμετωπίζει ή όχι.

8.5 Αρχιτεκτονική Πολυεπίπεδου Ταξινομητή

Στην προσπάθεια βελτίωσης της λειτουργίας της ταξινόμησης ενός συστήματος εξαγωγής συμπερασμάτων, είναι δυνατό και θεμιτό να χρησιμοποιηθεί ένας συνδυασμός στοιχειωδών ταξινομητών. Μια τέτοια αρχιτεκτονική προτείνεται στο [§ 7.2], αυτή η αρχιτεκτονική στην φάση της συγχώνευσης των μερικών συμπερασμάτων κάνει χρήση κατά πρώτον μιας διαδικασίας ψηφοφορίας και κατά δεύτερον του τροποποιημένου εκτιμητή αξιοπιστίας.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	4	146	4	146
Φάση Επανεκπαίδευσης	4	146	3	147

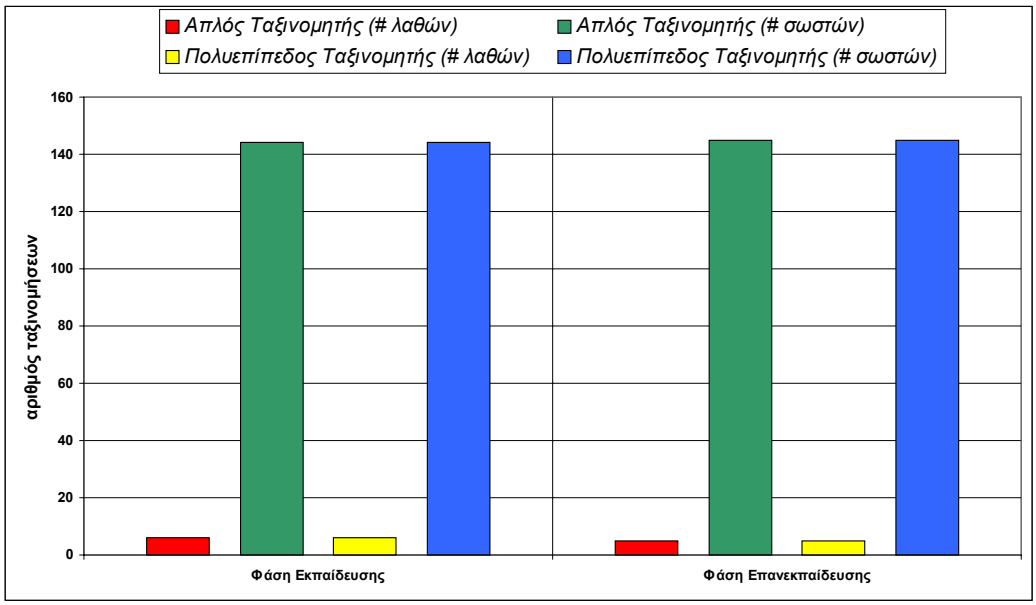
Πίνακας 8.5.1: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα iris.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	6	144	6	144
Φάση Επανεκπαίδευσης	5	145	5	145

Πίνακας 8.5.2: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα iris.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	4	146	3	147
Φάση Επανεκπαίδευσης	4	146	3	147

Πίνακας 8.5.3: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα iris.



Σχήμα 8.5.1: Σύγκριση των επιδόσεων του απλού και του πολυεπίπεδου ταξινομητή αναφορικά με το iris data set.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	583	4417	562	4438
Φάση Επανεκπαίδευσης	573	4427	546	4454

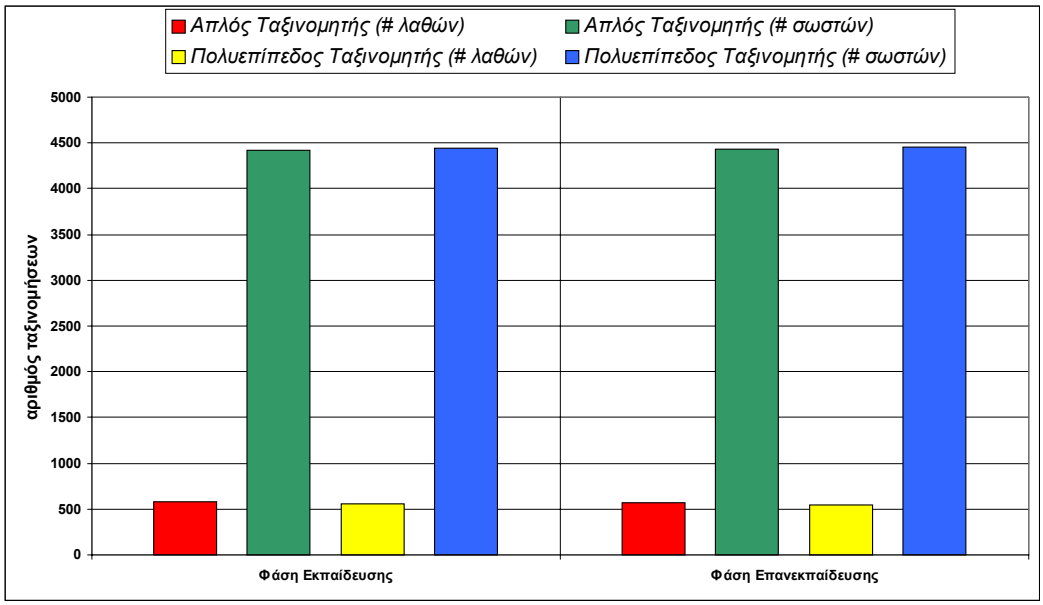
Πίνακας 8.5.4: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα clouds.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	798	4202	782	4218
Φάση Επανεκπαίδευσης	772	4228	741	4259

Πίνακας 8.5.5: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα clouds.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	539	4461	553	4447
Φάση Επανεκπαίδευσης	520	4480	525	4475

Πίνακας 8.5.6: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα clouds.



Σχήμα 8.5.2: Σύγκριση των επιδόσεων του απλού και του πολυεπίπεδου ταξινομητή αναφορικά με το clouds data set.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	12	339	15	336
Φάση Επανεκπαίδευσης	12	339	15	336

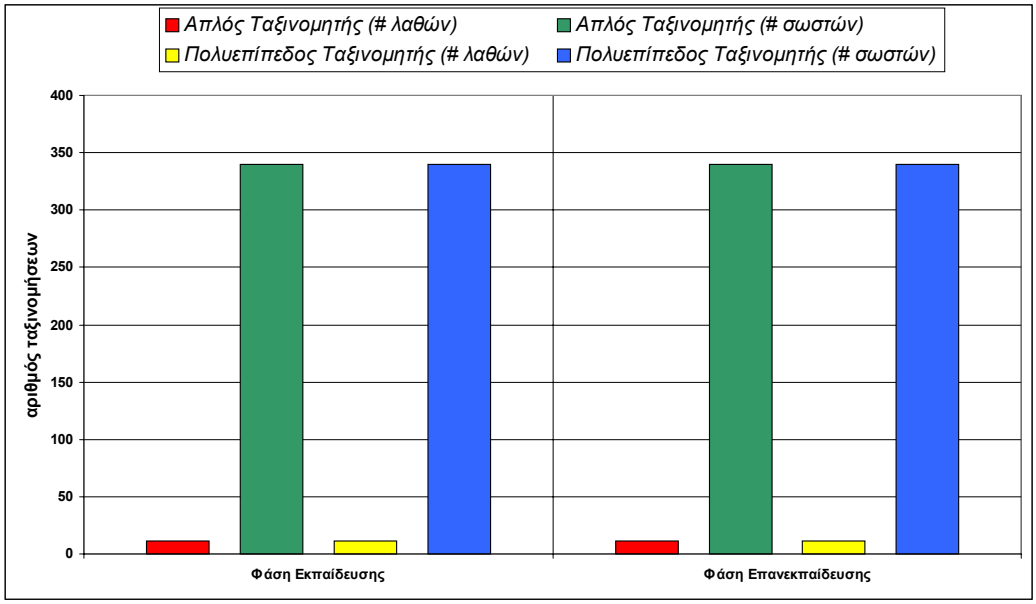
Πίνακας 8.5.7: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα ionosphere.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	14	337	10	341
Φάση Επανεκπαίδευσης	14	337	10	341

Πίνακας 8.5.8: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα ionosphere.

	Απλός Ταξινομητής (# λαθών)	Απλός Ταξινομητής (# σωστών)	Πολυεπίπεδος Ταξινομητής (# λαθών)	Πολυεπίπεδος Ταξινομητής (# σωστών)
Φάση Εκπαίδευσης	11	340	11	340
Φάση Επανεκπαίδευσης	11	340	11	340

Πίνακας 8.5.9: Πλήθος ορθών και λανθασμένων ταξινομήσεων στο πρόβλημα ionosphere.



Σχήμα 8.5.3: Σύγκριση των επιδόσεων του απλού και του πολυεπίπεδου ταξινομητή αναφορικά με το ionosphere data set.

Στο πρόβλημα iris ο απλός ταξινομητής χρησιμοποιεί 5 κανόνες και ο πολυεπίπεδος 5, 3 και 7 κανόνες. Στο πρόβλημα clouds ο απλός ταξινομητής χρησιμοποιεί 6 κανόνες και ο πολυεπίπεδος 6, 4 και 8 κανόνες. Τέλος στο πρόβλημα ionosphere ο απλός ταξινομητής χρησιμοποιεί 12 κανόνες και ο πολυεπίπεδος 12, 7 και 17 κανόνες. Το αναμενόμενο από την χρήση της πολυεπίπεδης αρχιτεκτονικής είναι μείωση του αριθμού των εσφαλμένων ταξινομήσεων. Στις πέντε από τις έξι πειραματικές προσομοιώσεις που αφορούν τα iris και clouds σύνολα ο αριθμός των εσφαλμένων ταξινομήσεων μειώνεται ή εμφανίζεται σταθερός (όταν ο αριθμός των λαθών είναι ήδη πολύ μικρός). Από την άλλη πλευρά ο πολυεπίπεδος ταξινομητής δεν μεταβάλει τον αριθμό των εσφαλμένων ταξινομήσεων που αφορούν στο πρόβλημα ionosphere.

Κλείνοντας μια σημείωση που σχετίζεται με το σύνολο των πειραμάτων που εκτελέστηκαν είναι ότι η χρήση αυξημένου αριθμού κανόνων ή πολλών συνόλων κανόνων ταυτόχρονα σημειώνει καλύτερες επιδόσεις αλλά έχει και αρνητικές συνέπειες. Οι πιο σημαντικές από αυτές είναι ότι σπαταλάται υπολογιστική ισχύς και ότι δεν γίνεται οικονομία στις παραμέτρους που χρειάζεται ο ταξινομητής για να λειτουργήσει.

Βιβλιογραφία

Βιβλιογραφία

2.1 Ασαφή Σύνολα

[1] Σπύρος Γ. Τζαφέστας, "Υπολογιστική Νοημοσύνη", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, τόμος Α, 2002

[2] Γ. Στάμου, "Νευρο-Ασαφή Συστήματα", pp 2-4

[3] J. S. R. Jang, C. T. Sun, E. Mizutani, "Neuro-Fuzzy and Soft Computing", Prentice Hall, 1997

[4] Σπύρος Γ. Τζαφέστας, "Ευφυής Αυτόματος Έλεγχος", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, 1995, pp 134-135

2.3 Νευρο-Ασαφή Συστήματα

[1] Σπύρος Γ. Τζαφέστας, "Υπολογιστική Νοημοσύνη", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, 2002

3.1 Νευρωνικά Δίκτυα Ταξινόμησης

[1] Σπύρος Γ. Τζαφέστας, "Υπολογιστική Νοημοσύνη", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, τόμος Α, 2002

[2] Αριστείδης Λύκας, "Υπολογιστική Νοημοσύνη", 1999

3.2.1 Περιγραφή του Αλγορίθμου Ανάστροφης Διάδοσης

[1] Σπύρος Γ. Τζαφέστας, "Υπολογιστική Νοημοσύνη", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, τόμος Α, 2002, pp 131-137

[2] J. S. R. Jang, C. T. Sun, E. Mizutani, "Neuro-Fuzzy and Soft Computing", Prentice Hall, 1997, pp 129-134 & pp 205-210

3.2.2 Σημειώσεις στον Αλγόριθμο Ανάστροφης Διάδοσης

[1] Z. Q. Luo, "On the Convergence of the LMS Algorithm with Adaptive Learning Rate for Linear Feedforward Networks", Neural Computation, 1991, pp 226-245

[2] Αριστείδης Λύκας, "Υπολογιστική Νοημοσύνη", 1999, pp 63-64

[3] Vogl, T. P., J. K. Mangis, A. K. Rigler, W. T. Zink, D. L. Alkon, "Accelerating the Convergence of the Back-Propagation Method", Biological Cybernetics, 1988, pp 257-263

[4] Christopher M. Bishop, "Neural Networks for Pattern Recognition", Oxford University Press, 1999, pp 140-148 & pp 263-271

3.3.2 Εκτιμητές Αξιοπιστίας

[1] Aarnoud Hoekstra, "Generalisation in Feed Forward Neural Classifiers", www.ph.tn.tudelft.nl/PHDTheses/AHoekstra/thesis_hoekstra.html, 1998

[2] J. A. Anderson, "Logistic Discrimination", Handbook of Statistics, vol. 2, 1982, pp 169-191

[3] L. P. Cordella, P. Foggia, C. Sansone, F. Tortorella, M. Vento, "Reliability Parameters to Improve Combination Strategies in Multi-Expert Systems", Pattern Analysis & Applications, 1999, pp 205-214

3.3.3 Τροποποιημένος Εκτιμητής Αξιοπιστίας

[1] John C. Platt, "Probabilities for SV Machines", Advances in Large Margin Classifiers, The MIT Press, 2000, pp 61-73

4 Νευρο-Ασαφές Σύστημα Εξαγωγής Συμπερασμάτων στηριζόμενο στο Γινόμενο Υποσυνόλων

[1] Sandeep Paul, Satish Kumar, "Subsethood-Product Fuzzy Neural Inference System", IEEE Transactions on Neural Networks, vol. 13, 2002, pp 578-599
[2] L. M. Fu, "Learning capacity and sample complexity on expert networks", IEEE Transactions on Neural Networks, vol. 7, 1996, pp 1517–1520.
[3] S. Paul, S. Kumar, "Rule based neuro-fuzzy linguistic networks for inference and function approximation", Knowledge Based Computer Systems, 1998, pp 287-298
[4] "Adaptive rule-based linguistic networks for function approximation", Advances in Pattern Recognition and Digital Techniques, 1999, pp 246-250.
[5] "Subsethood based adaptive linguistic networks for pattern classification", IEEE Transactions on Systems, 2002

4.7 Προσαρμοζόμενος Νευρο-Ασαφής Ταξινομητής

[1] M. Pertselakis, D. Frossyniotis, A. Stafylopatis, "An Adaptable Gaussian Neuro-Fuzzy Classifier", Modular hybrid artefacts with adaptive functionality
[2] D. Frossyniotis, M. Pertselakis, A. Stafylopatis, "A Multi-Clustering Fusion Algorithm", Proceedings of the Second Hellenic Conference on Artificial Intelligence, 2002, pp 225-236

5.2 Πρότυπο IEEE 802.11

[1] Institute of Electrical and Electronics Engineers, "The Standard IEEE 802.11", grouper.ieee.org/groups/802/11/, 1997
[2] Μ. Ε. Θεολόγου, "Δίκτυα Κινητών και Προσωπικών Επικοινωνιών", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, 2002, pp 12.1-12.38
[3] Pablo Brenner, "A Technical Tutorial on the IEEE 802.11 Protocol", 1997
[4] J. Zyren, A. Petrick, "IEEE 802.11 Tutorial", 1998

5.3 Το Στρώμα Δικτύου στο Internet

5.4 Τα Πρωτόκολλα Μεταφοράς του Internet

[1] Andrew S. Tanenbaum, "Computer Networks", Prentice Hall PTR, 1996
[2] Jean Walrand, "Communication Networks", McGraw Hill, 1998
[3] Ι. Στ. Βενιέρης, Ε. Νικολούζου, "Τεχνολογίες Διαδικτύου", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, 2002
[4] Β. Μάγκλαρης, Τ. Χιώτης, Θ. Καρούνος, Φ. Σταματελόπουλος, "Διαχείριση Δικτύων Υπολογιστών", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, 1994, pp 9-62

6 Πλατφόρμα Υλοποίησης Λογισμικού Java

[1] L. Lemay, R. Cadenhead, "Sams Teach Yourself Java 2 Platform in 21 Days", Sams Publishing, 1999
[2] "Εισαγωγή στη Γλώσσα Προγραμματισμού Java", Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, 2000
[3] "Introduction to Wireless Java Technology", www.java.sun.com, 2003
[4] "Java Community Process - Java Specification Request", http://jcp.org/jsr, 2003
[5] Qusay H. Mahmoud, "J2ME for Home Appliances and Consumer Electronic Devices", www.java.sun.com, 2003
[6] Eric Giguere, "Personal Basis Profile vs. Personal Profile: What's the Difference?", www.java.sun.com, 2003

[7] Jonathan Knudsen, “Java Programming on the Sharp Zaurus”, www.java.sun.com, 2002
[8] “Java 2 Platform, Standard Edition, v1.4.1 API Specification”, www.java.sun.com, 2002
[9] Ανδρέας Σταφυλοπάτης, “Γλώσσες Προγραμματισμού”, Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, 1996, pp 155-160
[10] “Why Are Thread.stop, Thread.suspend, Thread.resume and Runtime. runFinalizersOnExit Deprecated? ”, www.java.sun.com, 2002
[11] Γ. Παπακωνσταντίνου, Π. Τσανάκας, “Λειτουργικά Συστήματα: Εισαγωγή στον Προγραμματισμό των Sockets στο Unix”, pp 10

7.5.1 Ασύγχρονη Εκπομπή Ενός Χαρακτηριστικού Κάθε Δείγματος

[1] Α. Ν. Σταφυλοπάτης, “Ανάλυση Επίδοσης Υπολογιστικών Συστημάτων”, Έκδοση Εθνικού Μετσόβιου Πολυτεχνείου, 1996, pp 77-85

Π.1 Συνάρτηση Λάθους

[1] M. Abramowitz, I. A. Stegun, “Handbook of Mathematical Functions”, Dover Publications, 1965, pp 297-316
[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, “Numerical Recipes in C”, Cambridge University Press, 2002, pp 216-221

Παράρτημα 1

Συνάρτηση Λάθους

Παράρτημα 1

Συνάρτηση Λάθους

Η συνάρτηση λάθους (error function) που συμβολίζεται με erf(x), ορίζεται με πολλούς διαφορετικούς τρόπους στην βιβλιογραφία [1]. Σε αυτό το παράρτημα θα χρησιμοποιηθεί ο κάτωθι ορισμός:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-z^2} dz \qquad (\Pi\ 1.1)$$

Η συνάρτηση λάθους έχει δυο χρήσιμες ιδιότητες:

$erf(-x) = -erf(x)$ (Π 1.2)

Αυτή είναι γνωστή σαν σχέση συμμετρίας (symmetry relation).

Όταν το x τείνει στο άπειρο, η erf(x) τείνει στην μονάδα, δηλαδή:

$$\frac{2}{\sqrt{\pi}} \int_0^\infty e^{-z^2} dz = 1 \qquad (\Pi\ 1.3)$$

Η συμπληρωματική συνάρτηση λάθους (complementary error function) ορίζεται από:

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-z^2} dz \qquad (\Pi\ 1.4)$$

και συνδέεται με την συνάρτηση λάθους μέσω της σχέσης:

$erfc(x) = 1-erf(x)$ (Π 1.5)

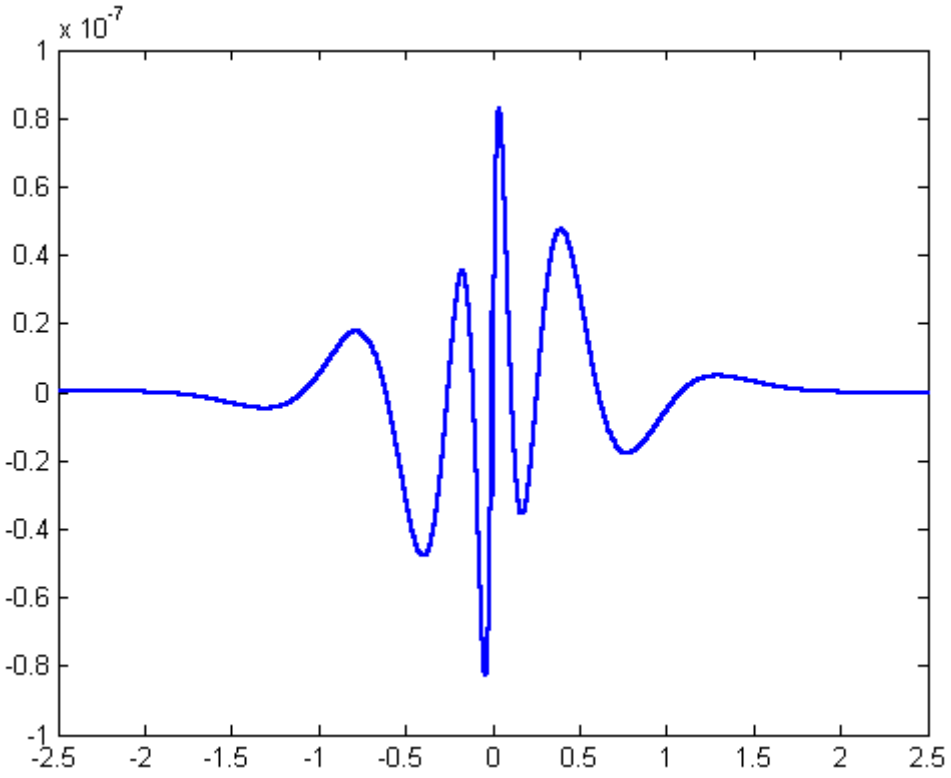
Ο υπολογισμός της συνάρτησης erfc(x) είναι δυνατόν να επιτευχθεί προσεγγίζοντας την με πολυώνυμα Chebyshev [2]. Συνεπώς ο υπολογισμός της erf(x) μπορεί ομοίως να γίνει προσεγγίζοντας την με πολυώνυμα Chebyshev:

```
double erf(double x)
{
    double t,z,value;

    z = Math.abs(x);
    t = 1.0/(1.0+0.5*z);
    value = t*Math.exp(-z*z-1.26551223+t*(1.00002368+t*(0.37409196+
        t*(0.09678418+t*(-0.18628806+t*(0.27886807+t*(-1.13520398+
        t*(1.48851587+t*(-0.82215223+t*0.17087277)))))))));
    if(x<0.0)
      value = 2.0-value;
    value = 1.0-value;
    return value;
}
```

(Π 1.6)

Αυτή η μέθοδος υπολογισμού της συνάρτησης erf(x) εμφανίζει ανώτερο όριο σφάλματος απόκλισης ίσο με 1.2×10^{-7}. Αυτό είναι εμφανές στο διάγραμμα που ακολουθεί όπου εμφανίζεται η διαφορά των τιμών της erf(x) όπως αυτές υπολογίζονται με την παραπάνω προσεγγιστική μέθοδο και των τιμών της συνάρτησης erf(x) όπως αυτές μπορούν να βρεθούν σε αντίστοιχους αναλυτικούς πίνακες.



Σχήμα Π 1.1: Το σφάλμα απόκλισης κατά τον υπολογισμό της συνάρτησης erf(x).

Παράρτημα 2

Σύνολα Δεδομένων

Παράρτημα 2

Σύνολα Δεδομένων

Σύνολα δεδομένων μου σχετίζονται με την μηχανική μάθηση (machine learning) υπάρχουν στο διαδίκτυο σε βάση δεδομένων που βρίσκεται στην URL διεύθυνση:

http://www.ics.uci.edu/~mlearn/MLRepository.html

Τα σύνολα δεδομένων που χρησιμοποιήθηκαν κατά την διενέργεια των πειραμάτων εξομοίωσης ήταν τα iris, clouds και ionosphere. Συνοπτικές πληροφορίες αναφορικά με αυτά τα τρία είδη συνόλων παρατίθενται στην συνέχεια:

`Iris Plants Database:` Το iris σύνολο δεδομένων αποτελείται από 150 δείγματα τα οποία ανήκουν σε 3 κατηγορίες. Σε κάθε κατηγορία αντιστοιχούν 50 δείγματα. Κάθε κλάση είναι γραμμικώς διαχωρίσιμη από τις άλλες δυο, ενώ οι υπόλοιπες δυο δεν είναι γραμμικώς διαχωρίσιμες μεταξύ τους. Κάθε δείγμα αποτελείται από 4 αριθμητικά χαρακτηριστικά.

	Ελάχιστη Τιμή	Μέγιστη Τιμή
Χαρακτηριστικό 1	4.3	7.9
Χαρακτηριστικό 2	2	4.4
Χαρακτηριστικό 3	1	6.9
Χαρακτηριστικό 4	0.1	2.5

Πίνακας Π2.1: Η ελάχιστη και η μέγιστη τιμή των χαρακτηριστικών κάθε iris δείγματος.

Κάθε κατηγορία αναφέρεται σε ένα είδος του iris φυτού, δηλαδή σε ένα από τα Iris Setosa, Iris Versicolour, Iris Virginica. Τα χαρακτηριστικά αναφέρονται στο μήκος σέπαλου (cm), στο πλάτος σέπαλου (cm), στο μήκος πετάλου (cm) και στο πλάτος πετάλου (cm).

`Clouds Database:` Το clouds σύνολο δεδομένων αποτελείται από 5000 δείγματα τα οποία ανήκουν σε 2 κατηγορίες. Σε κάθε κατηγορία αντιστοιχούν 2500 δείγματα. Κάθε δείγμα αποτελείται από 2 αριθμητικά χαρακτηριστικά.

	Ελάχιστη τιμή	Μέγιστη Τιμή
Χαρακτηριστικό 1	-3.2919	3.4806
Χαρακτηριστικό 2	-3.6399	4.7569

Πίνακας Π2.2: Η ελάχιστη και η μέγιστη τιμή των χαρακτηριστικών κάθε clouds δείγματος.

Τα δεδομένα αυτά έχουν δημιουργηθεί με τεχνητό τρόπο και δεν αντιστοιχούν σε κάποιο φυσικό μοντέλο.

Ionosphere Database: Το ionosphere σύνολο δεδομένων αποτελείται από 351 δείγματα τα οποία ανήκουν σε 2 κατηγορίες. Στην πρώτη κατηγορία αντιστοιχούν 225 δείγματα ενώ στην δεύτερη 126 δείγματα. Κάθε δείγμα αποτελείται από 34 συνεχή αριθμητικά χαρακτηριστικά.

Χαρακτηριστικό (#)	Ελάχιστη Τιμή	Μέγιστη Τιμή	Χαρακτηριστικό (#)	Ελάχιστη Τιμή	Μέγιστη Τιμή
1	0	1	18	-1	1
2	0	0	19	-1	1
3	-1	1	20	-1	1
4	-1	1	21	-1	1
5	-1	1	22	-1	1
6	-1	1	23	-1	1
7	-1	1	24	-1	1
8	-1	1	25	-1	1
9	-1	1	26	-1	1
10	-1	1	27	-1	1
11	-1	1	28	-1	1
12	-1	1	29	-1	1
13	-1	1	30	-1	1
14	-1	1	31	-1	1
15	-1	1	32	-1	1
16	-1	1	33	-1	1
17	-1	1	34	-1	1

Πίνακας Π2.3: Η ελάχιστη και η μέγιστη τιμή των χαρακτηριστικών κάθε ionosphere δείγματος.

Τα δεδομένα αυτά προέκυψαν από πειραματική διάταξη κεραιών οι οποίες εξέπεμπαν ηλεκτρομαγνητικά κύματα. Τα ελεύθερα ηλεκτρόνια της ιονόσφαιρας αποτελούσαν τους στόχους. Η κατηγορία good αναφέρεται στα ανακλώμενα σήματα τα οποία μαρτυρούσαν κάποια δομή στην ιονόσφαιρα, η κατηγορία bad αναφέρεται στα σήματα που διέρχονταν της ιονόσφαιρας. Τα λαμβανόμενα σήματα επεξεργάζονταν από μια συνάρτηση αυτοσυσχέτισης η οποία δεχόταν ως ορίσματα την περίοδο και τον αριθμό μιας παλμοσειράς. Υπήρχαν 17 παλμοσειρές στην πειραματική διάταξη. Κάθε δείγμα περιλαμβάνει 2 χαρακτηριστικά (από κάθε παλμοσειρά) τα οποία είναι οι σύνθετες τιμές που επέστρεφε η συνάρτηση αυτοσυσχέτισης.

Παράρτημα 3

Συγκεντρωτικοί Πίνακες Παραμέτρων

Παράρτημα 3

Συγκεντρωτικοί Πίνακες Παραμέτρων

Π 3.1 iris
Π 3.1.1 Τιμές αρχικές πριν την εκπαίδευση (τυχαίες)

για το σύνολο με 5 κανόνες

x_1^{σ}	x_2^{σ}	x_3^{σ}	x_4^{σ}
0.43962	0.45421	0.37858	0.50802

Πινάκας 3.1.1.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5
w_{1j}^{c}	5.5981	5.671	5.2929	5.9401	4.6369
w_{2j}^{c}	2.0205	3.4926	3.2746	4.3858	2.2887
w_{3j}^{c}	5.4009	1.923	4.0033	3.5748	4.2395
w_{4j}^{c}	0.87708	0.24601	1.5002	2.0994	1.9591

Πινάκας 3.1.1.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5
w_{1j}^{σ}	0.43962	0.45421	0.37858	0.50802	0.24738
w_{2j}^{σ}	0.20599	0.63535	0.57177	0.89585	0.28422
w_{3j}^{σ}	0.74046	0.31335	0.56883	0.51621	0.59783
w_{4j}^{σ}	0.42665	0.24259	0.60839	0.78315	0.74223

Πινάκας 3.1.1.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}	v_{j3}^{c}
1	0.35791	0.86243	0.023364
2	0.3233	0.65104	0.92416
3	0.96021	0.35694	0.15991
4	0.84537	0.50574	0.12189
5	0.86966	0.36701	0.52278

Πινάκας 3.1.1.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}	v_{j3}^{σ}
1	0.45053	0.8037	0.21636
2	0.42631	0.65572	0.84691
3	0.87215	0.44986	0.31194
4	0.79176	0.55402	0.28532
5	0.80876	0.45691	0.56595

Πινάκας 3.1.1.5: Διασπορές των μετασυναπτικών βαρών

Τιμές αρχικές πριν την εκπαίδευση (τυχαίες), για το σύνολο με 3 κανόνες

x_1^{σ}	x_2^{σ}	x_3^{σ}	x_4^{σ}
0.43962	0.45421	0.37858	0.50802

Πινάκας 3.1.1.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3
w_{1j}^{c}	5.5981	5.671	5.2929
w_{2j}^{c}	3.0561	2.1624	2.0205
w_{3j}^{c}	4.545	4.0272	6.6662
w_{4j}^{c}	0.38874	1.953	0.48862

Πινάκας 3.1.1.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3
w_{1j}^{σ}	0.43962	0.45421	0.37858
w_{2j}^{σ}	0.50802	0.24738	0.20599
w_{3j}^{σ}	0.63535	0.57177	0.89585
w_{4j}^{σ}	0.28422	0.74046	0.31335

Πινάκας 3.1.1.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}	v_{j3}^{c}
1	0.35791	0.86243	0.023364
2	0.3233	0.65104	0.92416
3	0.96021	0.35694	0.15991

Πινάκας 3.1.1.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}	v_{j3}^{σ}
1	0.45053	0.8037	0.21636
2	0.42631	0.65572	0.84691
3	0.87215	0.44986	0.31194

Πινάκας 3.1.1.10: Διασπορές των μετασυναπτικών βαρών

Τιμές αρχικές πριν την εκπαίδευση (τυχαίες), για το σύνολο με 7 κανόνες

x_1^{σ}	x_2^{σ}	x_3^{σ}	x_4^{σ}
0.43962	0.45421	0.37858	0.50802

Πινάκας 3.1.1.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{c}	5.5981	5.671	5.2929	5.9401	4.6369	4.4299	6.5768
w_{2j}^{c}	3.2746	4.3858	2.2887	3.853	2.3886	3.2645	3.0841
w_{3j}^{c}	4.2395	2.8456	1.3468	4.3255	5.7485	5.4153	1.1472
w_{4j}^{c}	1.8309	1.2946	1.2123	1.1914	1.926	1.3787	1.5629

Πινάκας 3.1.1.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{σ}	0.43962	0.45421	0.37858	0.50802	0.24738	0.20599	0.63535
w_{2j}^{σ}	0.57177	0.89585	0.28422	0.74046	0.31335	0.56883	0.51621
w_{3j}^{σ}	0.59783	0.42665	0.24259	0.60839	0.78315	0.74223	0.21808
w_{4j}^{σ}	0.70485	0.54844	0.52441	0.51834	0.73258	0.57294	0.62668

Πινάκας 3.1.1.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}	v_{j3}^{c}
1	0.35791	0.86243	0.023364
2	0.3233	0.65104	0.92416
3	0.96021	0.35694	0.15991
4	0.84537	0.50574	0.12189
5	0.86966	0.36701	0.52278
6	0.071251	0.42088	0.084929
7	0.50445	0.53425	0.49065

Πινάκας 3.1.1.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}	v_{j3}^{σ}
1	0.45053	0.8037	0.21636
2	0.42631	0.65572	0.84691
3	0.87215	0.44986	0.31194
4	0.79176	0.55402	0.28532
5	0.80876	0.45691	0.56595
6	0.24988	0.49462	0.25945
7	0.55312	0.57398	0.54345

Πινάκας 3.1.1.15: Διασπορές των μετασυναπτικών βαρών

Π 3.1.2 Τιμές μετά την εκπαίδευση

για το σύνολο με 5 κανόνες

x_1^{σ}	x_2^{σ}	x_3^{σ}	x_4^{σ}
0.88925	0.10054	0.041476	0.161

Πινάκας 3.1.2.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5
w_{1j}^{c}	5.9219	5.6721	5.9201	6.0583	4.3644
w_{2j}^{c}	2.7975	3.4946	3.3958	4.6744	2.4649
w_{3j}^{c}	5.35	1.9153	4.7879	3.4615	4.4612
w_{4j}^{c}	1.9349	0.24212	1.4568	2.2902	2.1132

Πινάκας 3.1.2.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5
w_{1j}^{σ}	0.89325	0.45336	0.98936	0.34054	0.0082511
w_{2j}^{σ}	0.10209	0.63225	0.45969	0.33927	0.19852
w_{3j}^{σ}	0.36958	0.28558	0.25359	0.50563	0.01983
w_{4j}^{σ}	0.22559	0.23575	0.11068	0.59057	0.55083

Πινάκας 3.1.2.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}	v_{j3}^{c}
1	-0.00091887	-0.011685	1.0116
2	1	9.3742e-006	1.4075e-005
3	-0.00098404	1.0144	-0.013401
4	0.64256	0.58998	0.14057
5	0.00017987	-0.16535	1.1443

Πινάκας 3.1.2.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}	v_{j3}^{σ}
1	0.66807	0.98246	0.91465
2	0.42563	0.65535	0.84689
3	0.99054	0.68376	0.63595
4	0.68034	0.53614	0.28883
5	0.59849	0.59796	0.63159

Πινάκας 3.1.2.5: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την εκπαίδευση , για το σύνολο με 3 κανόνες

x_1^{σ}	x_2^{σ}	x_3^{σ}	x_4^{σ}
0.61233	1.4211	0.3503	3.0019

Πινάκας 3.1.2.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3
w_{1j}^{c}	6.1157	5.7639	5.2829
w_{2j}^{c}	4.5297	2.7003	1.9875
w_{3j}^{c}	4.5492	5.3566	6.7246
w_{4j}^{c}	-1.1155	3.5807	0.46077

Πινάκας 3.1.2.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3
w_{1j}^{σ}	0.4272	0.91313	0.38159
w_{2j}^{σ}	0.13549	1.4213	0.088969
w_{3j}^{σ}	0.46525	0.4198	0.80404
w_{4j}^{σ}	0.13765	4.0648	0.24059

Πινάκας 3.1.2.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}	v_{j3}^{c}
1	1.006	0.003361	-0.052742
2	-0.018899	0.48914	1.0548
3	0.94775	0.33749	0.19973

Πινάκας 3.1.2.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}	v_{j3}^{σ}
1	4.077	1.8179	6.5895
2	5.108	4.7485	0.030734
3	0.871	0.47536	0.20011

Πινάκας 3.1.2.10: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την εκπαίδευση , για το σύνολο με 7 κανόνες

x_1^{σ}	x_2^{σ}	x_3^{σ}	x_4^{σ}
0.72294	0.088538	0.12838	0.12909

Πινάκας 3.1.2.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{c}	6.1732	5.7896	5.3051	6.1051	4.9573	4.4425	6.6825
w_{2j}^{c}	3.6281	4.4597	2.5496	3.379	2.794	3.3426	2.9889
w_{3j}^{c}	4.8644	2.8382	1.3119	5.3445	5.2701	5.4862	0.93563
w_{4j}^{c}	1.3622	1.4003	1.0512	2.059	1.7999	1.4368	1.6879

Πινάκας 3.1.2.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{σ}	1.2864	0.31812	0.48132	0.79721	0.86066	0.00025344	0.48432
w_{2j}^{σ}	0.46282	0.89894	0.6105	0.91038	0.093019	0.37178	0.44908
w_{3j}^{σ}	0.2645	0.010204	0.29581	0.28829	0.30594	0.52066	0.096622
w_{4j}^{σ}	0.26617	0.46196	0.69067	0.083637	0.12939	0.56835	0.47064

Πινάκας 3.1.2.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}	v_{j3}^{c}
1	-0.00027278	1.0172	-0.015308
2	0.34909	0.59914	0.80377
3	1	-2.6627e-005	-3.4809e-005
4	-4.2276e-006	0.0045196	0.99551
5	7.6652e-005	-0.11398	1.1131
6	0.02314	0.42366	0.10022
7	0.53548	0.45774	0.41282

Πινάκας 3.1.2.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}	v_{j3}^{σ}
1	0.75208	0.64049	0.58496
2	0.40708	0.64243	0.81782
3	0.8899	0.49592	0.39824
4	0.78712	0.88058	0.80866
5	0.52894	0.82993	0.76135
6	0.29622	0.4584	0.2476
7	0.53875	0.5443	0.51693

Πινάκας 3.1.2.15: Διασπορές των μετασυναπτικών βαρών

Π 3.1.3 Τιμές μετά την επανεκπαίδευση

για το σύνολο με 5 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ
0.88911	0.11416	0.044968	0.13353

Πινάκας 3.1.3.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5
w_{1j}^c	5.9192	5.6721	5.9226	6.0583	4.3644
w_{2j}^c	2.7769	3.4946	3.3867	4.6744	2.4649
w_{3j}^c	5.3634	1.9153	4.788	3.4615	4.4612
w_{4j}^c	1.9523	0.24212	1.4449	2.2902	2.1132

Πινάκας 3.1.3.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5
w_{1j}^σ	0.89194	0.45336	0.99054	0.34054	0.0082511
w_{2j}^σ	0.10473	0.63225	0.46785	0.33927	0.19852
w_{3j}^σ	0.3546	0.28558	0.24578	0.50563	0.01983
w_{4j}^σ	0.20155	0.23575	0.1218	0.59057	0.55083

Πινάκας 3.1.3.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^c	v_{j2}^c	v_{j3}^c
1	-0.00091349	-0.010604	1.0105
2	1	9.3742e-006	1.4075e-005
3	-0.00089136	1.0231	-0.022238
4	0.64256	0.58998	0.14057
5	0.00017987	-0.16535	1.1443

Πινάκας 3.1.3.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^σ	v_{j2}^σ	v_{j3}^σ
1	0.66807	0.98146	0.91356
2	0.42563	0.65535	0.84689
3	0.99054	0.6852	0.63752
4	0.68034	0.53614	0.28883
5	0.59849	0.59796	0.63159

Πινάκας 3.1.3.5: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την επανεκπαίδευση , για το σύνολο με 3 κανόνες

x_1^{σ}	x_2^{σ}	x_3^{σ}	x_4^{σ}
0.61238	1.421	0.35029	3.0019

Πινάκας 3.1.3.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3
w_{1j}^{c}	6.1158	5.7639	5.2829
w_{2j}^{c}	4.5298	2.7003	1.9875
w_{3j}^{c}	4.5491	5.3564	6.7246
w_{4j}^{c}	-1.1155	3.5807	0.46077

Πινάκας 3.13.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3
w_{1j}^{σ}	0.42715	0.91307	0.38159
w_{2j}^{σ}	0.13511	1.4213	0.088962
w_{3j}^{σ}	0.46516	0.41988	0.80404
w_{4j}^{σ}	0.13725	4.0648	0.24059

Πινάκας 3.1.3.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}	v_{j3}^{c}
1	1.006	0.00336	-0.052685
2	-0.018715	0.48425	1.0553
3	0.94775	0.33749	0.19973

Πινάκας 3.1.3.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}	v_{j3}^{σ}
1	4.077	1.8179	6.5895
2	5.108	4.7485	0.032598
3	0.871	0.47536	0.20011

Πινάκας 3.1.3.10: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την επανεκπαίδευση , για το σύνολο με 7 κανόνες

x_1^{σ}	x_2^{σ}	x_3^{σ}	x_4^{σ}
0.72294	0.088538	0.12838	0.12909

Πινάκας 3.1.3.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{c}	6.1732	5.7896	5.3051	6.1051	4.9573	4.4425	6.6825
w_{2j}^{c}	3.6281	4.4597	2.5496	3.379	2.794	3.3426	2.9889
w_{3j}^{c}	4.8644	2.8382	1.3119	5.3445	5.2701	5.4862	0.93563
w_{4j}^{c}	1.3622	1.4003	1.0512	2.059	1.7999	1.4368	1.6879

Πινάκας 3.1.3.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{σ}	1.2864	0.31812	0.48132	0.79721	0.86066	0.00025344	0.48432
w_{2j}^{σ}	0.46282	0.89894	0.6105	0.91038	0.093019	0.37178	0.44908
w_{3j}^{σ}	0.2645	0.010204	0.29581	0.28829	0.30594	0.52066	0.096622
w_{4j}^{σ}	0.26617	0.46196	0.69067	0.083637	0.12939	0.56835	0.47064

Πινάκας 3.1.3.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}	v_{j3}^{c}
1	-0.00027278	1.0172	-0.015308
2	0.34909	0.59914	0.80377
3	1	-2.6627e-005	-3.4809e-005
4	-4.2276e-006	0.0045196	0.99551
5	7.6652e-005	-0.11398	1.1131
6	0.02314	0.42366	0.10022
7	0.53548	0.45774	0.41282

Πινάκας 3.1.3.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}	v_{j3}^{σ}
1	0.75208	0.64049	0.58496
2	0.40708	0.64243	0.81782
3	0.8899	0.49592	0.39824
4	0.78712	0.88058	0.80866
5	0.52894	0.82993	0.76135
6	0.29622	0.4584	0.2476
7	0.53875	0.5443	0.51693

Πινάκας 3.1.3.15: Διασπορές των μετασυναπτικών βαρών

Π 3.2 clouds

Π 3.2.1 Τιμές αρχικές πριν την εκπαίδευση (τυχαίες)

για το σύνολο με 6 κανόνες

x_1^{σ}	x_2^{σ}
0.89693	0.48708

Πινάκας 3.2.1.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6
w_{1j}^{c}	3.3014	-0.48229	0.40037	-2.721	-1.9097	1.8697
w_{2j}^{c}	3.6897	-2.2928	-2.1324	1.3697	3.1667	-0.34049

Πινάκας 3.2.1.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6
w_{1j}^{σ}	0.89693	0.48708	0.58269	0.24458	0.33246	0.74185
w_{2j}^{σ}	0.80419	0.26709	0.28149	0.5959	0.75724	0.44237

Πινάκας 3.2.1.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.99562	0.41012
2	0.5467	0.063687
3	0.18923	0.77408
4	0.86313	0.095848
5	0.11641	0.56558
6	0.79606	0.34624

Πινάκας 3.2.1.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.89693	0.48708
2	0.58269	0.24458
3	0.33246	0.74185
4	0.80419	0.26709
5	0.28149	0.5959
6	0.75724	0.44237

Πινάκας 3.2.1.5: Διασπορές των μετασυναπτικών βαρών

Τιμές αρχικές πριν την εκπαίδευση (τυχαίες), για το σύνολο με 4 κανόνες

x_1^{σ}	x_2^{σ}
0.89693	0.48708

Πινάκας 3.2.1.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4
w_{1j}^{c}	3.3014	-0.48229	0.40037	-2.721
w_{2j}^{c}	-1.5647	2.9954	3.6897	-2.2928

Πινάκας 3.2.1.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4
w_{1j}^{σ}	0.89693	0.48708	0.58269	0.24458
w_{2j}^{σ}	0.33246	0.74185	0.80419	0.26709

Πινάκας 3.2.1.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.99562	0.41012
2	0.5467	0.063687
3	0.18923	0.77408
4	0.86313	0.095848

Πινάκας 3.2.1.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.89693	0.48708
2	0.58269	0.24458
3	0.33246	0.74185
4	0.80419	0.26709

Πινάκας 3.2.1.10: Διασπορές των μετασυναπτικών βαρών

Τιμές αρχικές πριν την εκπαίδευση (τυχαίες), για το σύνολο με 8 κανόνες

x_1^{σ}	x_2^{σ}
0.89693	0.48708

Πινάκας 3.2.1.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8
w_{1j}^{c}	3.3014	-0.48229	0.40037	-2.721	-1.9097	1.8697	2.4452	-2.5132
w_{2j}^{c}	-2.1324	1.3697	3.1667	-0.34049	1.2519	3.4936	2.0798	-2.6136

Πινάκας 3.2.1.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8
w_{1j}^{σ}	0.89693	0.48708	0.58269	0.24458	0.33246	0.74185	0.80419	0.26709
w_{2j}^{σ}	0.28149	0.5959	0.75724	0.44237	0.58533	0.78658	0.65965	0.23829

Πινάκας 3.2.1.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.99562	0.41012
2	0.5467	0.063687
3	0.18923	0.77408
4	0.86313	0.095848
5	0.11641	0.56558
6	0.79606	0.34624
7	0.55047	0.83797
8	0.65665	0.054695

Πινάκας 3.2.1.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.99562	0.41012
2	0.5467	0.063687
3	0.18923	0.77408
4	0.86313	0.095848
5	0.11641	0.56558
6	0.79606	0.34624
7	0.55047	0.83797
8	0.65665	0.054695

Πινάκας 3.2.1.15: Διασπορές των μετασυναπτικών βαρών

Π 3.2.2 Τιμές μετά την εκπαίδευση

για το σύνολο με 6 κανόνες

x_1^{σ}	x_2^{σ}
0.0012284	0.0011951

Πινάκας 3.2.2.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6
w_{1j}^{c}	1.0018	0.045465	0.13883	0.3316	1.0046	1.2274
w_{2j}^{c}	1.9595	0.6099	-1.4435	0.96989	4.7734	2.2954

Πινάκας 3.2.2.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6
w_{1j}^{σ}	0.46314	5.1409	0.81978	0.61733	0.89235	0.84226
w_{2j}^{σ}	5.7399	5.0811	0.5002	0.35732	4.8164	7.3447

Πινάκας 3.2.2.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	2.9293	-1.9292
2	1.0423	-0.04233
3	1.0386	-0.03863
4	1.1894	-0.18942
5	-0.98735	1.9872
6	-0.09878	1.0988

Πινάκας 3.2.2.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	1.7698	1.7694
2	4.3365	4.3353
3	4.4969	4.4958
4	0.59791	0.59777
5	2.3366	2.3363
6	3.1722	3.1711

Πινάκας 3.2.2.5: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την εκπαίδευση ,

για το σύνολο με 4 κανόνες

x_1^{σ}	x_2^{σ}
0.00085224	0.00067695

Πινάκας 3.2.2.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4
w_{1j}^{c}	1.9508	0.63096	0.69184	0.18758
w_{2j}^{c}	-2.1485	0.79545	3.3929	0.14164

Πινάκας 3.2.2.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4
w_{1j}^{σ}	2.7446	1.158	0.20666	1.1374
w_{2j}^{σ}	30.3566	1.4192	0.037405	1.4004

Πινάκας 3.2.2.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.76385	0.23524
2	-0.48828	1.4868
3	1.0239	-0.022599
4	1.3394	-0.33926

Πινάκας 3.2.2.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	4.6601	4.616
2	2.2669	2.2677
3	0.75847	0.80256
4	2.9553	2.9527

Πινάκας 3.2.2.10: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την εκπαίδευση ,

για το σύνολο με 8 κανόνες

x_1^{σ}	x_2^{σ}
0.0010693	0.002379

Πινάκας 3.2.2.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8
w_{1j}^{c}	0.99484	0.19295	0.87772	0.20744	0.29971	0.96963	0.9824	-0.88079
w_{2j}^{c}	-2.4853	0.88723	3.2169	-1.0608	0.97169	1.2283	1.146	1.7356

Πινάκας 3.2.2.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8
w_{1j}^{σ}	2.6933	13.6234	0.97269	0.72128	0.66918	1.0248	0.42014	0.26413
w_{2j}^{σ}	0.27795	1.3698	2.6807	0.42305	0.41488	1.2592	1.4681	3.1384

Πινάκας 3.2.2.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	1.0237	-0.024283
2	1.0251	-0.025093
3	0.10675	0.89312
4	1.0578	-0.057749
5	1.5561	-0.55598
6	-0.6976	1.6971
7	1.8702	-0.87002
8	1.0443	-0.04423

Πινάκας 3.2.2.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	1.6641	1.6089
2	5.4762	5.4652
3	3.8627	3.8495
4	1.4272	1.4246
5	0.91198	0.91013
6	1.4195	1.4176
7	1.7384	1.7347
8	2.3574	2.3521

Πινάκας 3.2.2.15: Διασπορές των μετασυναπτικών βαρών

Π 3.2.3 Τιμές μετά την επανεκπαίδευση

για το σύνολο με 6 κανόνες

x_1^{σ}	x_2^{σ}
0.0013477	0.0015386

Πινάκας 3.2.3.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6
w_{1j}^{c}	1.0204	0.045596	0.13878	0.3327	0.99272	1.2289
w_{2j}^{c}	1.9596	0.61062	-1.4463	0.97033	4.7727	2.2954

Πινάκας 3.2.3.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6
w_{1j}^{σ}	0.49419	5.1404	0.82004	0.61515	0.89284	0.84413
w_{2j}^{σ}	5.7395	5.081	0.49441	0.35761	4.8185	7.3445

Πινάκας 3.2.3.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	2.9296	-1.9295
2	1.0355	-0.03547
3	1.0384	-0.03836
4	1.1887	-0.18868
5	-0.9836	1.9835
6	-0.09506	1.0951

Πινάκας 3.2.3.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	1.7703	1.77
2	4.3368	4.3356
3	4.4969	4.4957
4	0.59767	0.59754
5	2.3354	2.3351
6	3.1724	3.1713

Πινάκας 3.2.3.5: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την επανεκπαίδευση ,

για το σύνολο με 4 κανόνες

x_1^{σ}	x_2^{σ}
0.00091576	0.0067239

Πινάκας 3.2.3.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4
w_{1j}^{c}	1.9508	0.62496	0.69184	0.23885
w_{2j}^{c}	-2.1485	0.82106	3.3929	0.15514

Πινάκας 3.2.3.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4
w_{1j}^{σ}	2.7432	1.2392	0.20666	1.0751
w_{2j}^{σ}	30.3564	1.5066	0.037405	1.2925

Πινάκας 3.2.3.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.76762	0.23151
2	-0.4273	1.4258
3	1.0239	-0.022599
4	1.4171	-0.41693

Πινάκας 3.2.3.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	4.6605	4.6165
2	2.2398	2.2406
3	0.75847	0.80256
4	2.9753	2.9727

Πινάκας 3.2.3.10: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την επανεκπαίδευση ,

για το σύνολο με 8 κανόνες

x_1^{σ}	x_2^{σ}
0.0059771	0.00075676

Πινάκας 3.2.3.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8
w_{1j}^{c}	0.99484	0.19294	0.88067	0.20766	0.29026	0.97321	0.98855	-0.87986
w_{2j}^{c}	-2.4853	0.88586	3.2154	-1.0607	0.98655	1.2268	1.1471	1.7356

Πινάκας 3.2.3.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8
w_{1j}^{σ}	2.6933	13.6237	0.96599	0.72109	0.66996	1.0187	0.4352	0.25829
w_{2j}^{σ}	0.27795	1.37	2.6814	0.42314	0.4288	1.2553	1.4738	3.1382

Πινάκας 3.2.3.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	1.0237	-0.024283
2	1.0229	-0.022911
3	0.1006	0.89926
4	1.0559	-0.055834
5	1.5536	-0.55342
6	-0.70267	1.7022
7	1.8643	-0.86418
8	1.0427	-0.042701

Πινάκας 3.2.3.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	1.6641	1.6089
2	5.4759	5.4649
3	3.863	3.8498
4	1.4271	1.4246
5	0.91148	0.90963
6	1.4235	1.4216
7	1.7358	1.732
8	2.3575	2.3521

Πινάκας 3.2.3.15: Διασπορές των μετασυναπτικών βαρών

Π 3.3 ionosphere

Π 3.3.1 Τιμές αρχικές πριν την εκπαίδευση (τυχαίες)

για το σύνολο με 12 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
0.35917	0.64388	0.70529	0.82015	0.64204	0.84929	0.64534	0.51983	0.50574	0.64949	0.55684	0.84314	0.39285	0.80736	0.27734	0.29728	0.22498

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.44378	0.26132	0.63034	0.38002	0.43799	0.82743	0.53251	0.79549	0.78117	0.54879	0.55726	0.78957	0.74248	0.77736	0.29169	0.80267	0.37728

Πινάκας 3.3.1.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12
w_{1j}^c	0.22685	0.65138	0.14363	0.57961	0.85274	0.96495	0.1073	0.18704	0.84485	0.61712	0.2431	0.2305
w_{2j}^c	0	0	0	0	0	0	0	0	0	0	0	0
w_{3j}^c	-0.17834	0.29808	-0.54312	-0.23661	0.00242	0.012151	0.38707	0.81018	0.26977	-0.2796	-0.37144	-0.50013
w_{4j}^c	0.60256	0.19058	0.31279	-0.13745	-0.76161	0.98629	0.19307	0.16297	0.64495	0.6136	-0.05678	-0.85571
w_{5j}^c	0.63158	-0.98475	-0.60414	-0.47048	-0.95191	0.53774	0.30936	0.27435	0.81735	-0.12606	0.83242	0.97313
w_{6j}^c	-0.59127	-0.57679	-0.52223	0.29083	-0.71594	0.012791	-0.87818	-0.67335	-0.66644	-0.011	0.31441	0.41086
w_{7j}^c	-0.13181	0.31339	0.012918	-0.00534	-0.65367	-0.51123	0.74775	-0.98667	-0.31366	-0.10827	0.18113	-0.61351
w_{8j}^c	0.75294	0.40286	0.038683	-0.83531	-0.59572	-0.89717	0.47359	-0.7601	0.34764	0.24181	-0.93466	-0.49183
w_{9j}^c	0.97588	-0.30808	0.86051	-0.09412	0.34282	0.65655	-0.8696	-0.16329	-0.5154	0.53758	-0.14957	0.16478
w_{10j}^c	0.40552	0.55081	0.29886	-0.83012	0.22056	-0.06792	-0.37259	-0.35584	-0.8168	0.39073	0.77397	0.6552
w_{11j}^c	0.62586	-0.15574	0.66769	0.4086	-0.7982	-0.59265	-0.68081	0.11407	-0.75903	0.11034	0.093903	0.79187
w_{12j}^c	-0.4813	-0.11303	-0.766	-0.44731	0.28505	0.99775	0.52444	-0.95835	-0.98639	-0.75503	0.89136	-0.54173
w_{13j}^c	-0.99698	-0.91239	0.67593	0.84727	-0.30001	-0.92188	0.34459	0.20076	-0.58784	-0.0791	-0.5235	-0.67937

w_{14j}^{c}	-0.06001	0.89096	0.01158	-0.6394	0.58097	-0.64624	-0.05216	0.5007	-0.05442	0.69138	-0.056	0.30255
w_{15j}^{c}	0.073356	0.067208	-0.38638	-0.04808	0.44881	-0.84919	0.6895	-0.18382	-0.30938	0.1766	0.69229	0.52674
w_{16j}^{c}	-0.07609	0.52427	-0.74545	-0.8569	-0.85564	-0.61388	-0.168	-0.13735	0.24406	0.025618	0.17582	-0.78859
w_{17j}^{c}	-0.20709	0.63365	-0.35462	0.68411	-0.84425	-0.14664	0.20791	-0.60244	-0.92581	-0.32871	0.80215	-0.5205
w_{18j}^{c}	0.45337	0.56222	-0.3648	-0.36397	0.76783	-0.47317	-0.06281	-0.37646	-0.76305	0.97588	-0.28433	0.19711
w_{19j}^{c}	0.71922	-0.70841	-0.86797	-0.48453	0.1359	0.33618	0.26449	-0.04697	0.15859	-0.82021	-0.77247	0.80327
w_{20j}^{c}	0.88142	0.88405	-0.11481	0.45283	-0.75248	0.30491	-0.27451	0.2643	0.11054	-0.43245	-0.14376	-0.3737
w_{21j}^{c}	-0.01384	0.11821	0.53711	-0.58748	0.78451	0.072282	-0.81918	-0.56279	0.11354	0.95485	-0.66394	0.40283
w_{22j}^{c}	-0.89561	0.86949	0.80761	0.3062	0.13409	-0.36716	0.77394	-0.69735	-0.69335	-0.5228	-0.46959	-0.37636
w_{23j}^{c}	0.10306	0.55015	0.31602	0.30111	-0.77843	-0.37034	0.57445	0.17282	-0.69637	-0.77467	-0.04495	0.92622
w_{24j}^{c}	-0.02269	0.55044	0.3771	0.78362	0.44856	-0.27307	-0.43917	0.84028	-0.26765	0.024457	-0.42913	0.037188
w_{25j}^{c}	0.53921	-0.06305	0.12359	-0.71618	-0.18071	-0.53572	0.6765	0.34178	-0.98526	-0.90617	-0.35122	0.73841
w_{26j}^{c}	0.1826	-0.33398	-0.48606	0.20015	0.32419	0.56004	0.96117	0.87871	-0.50668	0.45008	-0.16093	0.98875
w_{27j}^{c}	0.55449	-0.74015	-0.82393	-0.61604	-0.17651	-0.11788	-0.78615	-0.93085	0.24692	-0.26768	-0.90447	0.6793
w_{28j}^{c}	0.26296	-0.73047	0.61793	0.55184	0.035117	-0.75491	-0.71286	0.17721	0.36163	0.62876	0.80928	0.53285
w_{29j}^{c}	0.23718	0.91403	0.10355	0.089406	0.77979	-0.18127	0.9419	-0.87651	0.034893	0.27426	-0.59975	-0.62057
w_{30j}^{c}	-0.97063	-0.96427	-0.55183	-0.90657	0.37281	-0.75221	0.20954	0.92637	0.68899	0.86505	-0.30609	0.44758
w_{31j}^{c}	-0.7901	-0.04701	-0.87147	-0.53013	0.97552	0.58478	-0.58628	-0.24313	0.25678	-0.69859	0.67888	0.40692
w_{32j}^{c}	0.055125	-0.08316	-0.56536	0.56353	0.8312	-0.17299	-0.6038	0.010553	-0.02365	-0.07917	-0.30425	-0.31911
w_{33j}^{c}	0.96452	0.69338	-0.09311	0.14092	-0.39164	-0.77411	-0.9284	0.70823	0.39233	-0.67594	-0.24984	-0.98108
w_{34j}^{c}	0.46689	0.2958	0.007371	0.58084	0.56151	0.60129	0.20315	0.13286	0.81088	0.9228	-0.52662	-0.52426

Πινάκας 3.3.1.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12
w_{1j}^{σ}	0.35879	0.65597	0.30054	0.60573	0.79692	0.87546	0.27511	0.33093	0.7914	0.63199	0.37017	0.36135
w_{2j}^{σ}	0.31216	0.73699	0.5681	0.88268	0.45767	0.5765	0.82442	0.75683	0.53777	0.37357	0.51432	0.35044
w_{3j}^{σ}	0.48758	0.65433	0.35991	0.46719	0.55085	0.55425	0.68548	0.83356	0.64442	0.45214	0.42	0.37495
w_{4j}^{σ}	0.7609	0.6167	0.65948	0.50189	0.28344	0.8952	0.61758	0.60704	0.77573	0.76476	0.53013	0.2505
w_{5j}^{σ}	0.77105	0.20534	0.33855	0.38533	0.21683	0.73821	0.65828	0.64602	0.83607	0.50588	0.84135	0.8906

w_{6j}^{σ}	0.34305	0.34812	0.36722	0.65179	0.29942	0.55448	0.24264	0.31433	0.31675	0.54615	0.66004	0.6938
w_{7j}^{σ}	0.50387	0.65969	0.55452	0.54813	0.32121	0.37107	0.81171	0.20466	0.44022	0.51211	0.61339	0.33527
w_{8j}^{σ}	0.81353	0.691	0.56354	0.25764	0.3415	0.23599	0.71576	0.28397	0.67167	0.63463	0.22287	0.37786
w_{9j}^{σ}	0.89156	0.44217	0.85118	0.51706	0.66999	0.77979	0.24564	0.49285	0.36961	0.73815	0.49765	0.60767
w_{10j}^{σ}	0.69193	0.74278	0.6546	0.25946	0.6272	0.52623	0.41959	0.42546	0.26412	0.68676	0.82089	0.77932
w_{11j}^{σ}	0.76905	0.49549	0.78369	0.69301	0.27063	0.34257	0.31172	0.58992	0.28434	0.58862	0.58287	0.82715
w_{12j}^{σ}	0.38155	0.51044	0.2819	0.39344	0.64977	0.89921	0.73356	0.21458	0.20476	0.28574	0.86198	0.36039
w_{13j}^{σ}	0.20106	0.23066	0.78657	0.84654	0.445	0.22734	0.67061	0.62027	0.34426	0.52232	0.36678	0.31222
w_{14j}^{σ}	0.529	0.86184	0.55405	0.32621	0.75334	0.32382	0.53174	0.72525	0.53095	0.79198	0.5304	0.65589
w_{15j}^{σ}	0.57567	0.57352	0.41477	0.53317	0.70708	0.25278	0.79133	0.48566	0.44172	0.61181	0.7923	0.73436
w_{16j}^{σ}	0.52337	0.73349	0.28909	0.25009	0.25053	0.33514	0.4912	0.50193	0.63542	0.55897	0.61154	0.27399
w_{17j}^{σ}	0.47752	0.77178	0.42588	0.78944	0.25451	0.49868	0.62277	0.33914	0.22597	0.43495	0.83075	0.36783
w_{18j}^{σ}	0.70868	0.74678	0.42232	0.42261	0.81874	0.38439	0.52802	0.41824	0.28293	0.89156	0.45049	0.61899
w_{19j}^{σ}	0.80173	0.30206	0.24621	0.38041	0.59757	0.66766	0.64257	0.53356	0.60551	0.26293	0.27964	0.83115
w_{20j}^{σ}	0.8585	0.85942	0.50982	0.70849	0.28663	0.65672	0.45392	0.6425	0.58869	0.39864	0.49968	0.41921
w_{21j}^{σ}	0.54516	0.59137	0.73799	0.34438	0.82458	0.5753	0.26329	0.35302	0.58974	0.8842	0.31762	0.69099
w_{22j}^{σ}	0.23654	0.85432	0.83266	0.65717	0.59693	0.42149	0.82088	0.30593	0.30733	0.36702	0.38564	0.41828
w_{23j}^{σ}	0.58607	0.74255	0.66061	0.65539	0.27755	0.42038	0.75106	0.61049	0.30627	0.27887	0.53427	0.87418
w_{24j}^{σ}	0.54206	0.74265	0.68199	0.82427	0.707	0.45442	0.39629	0.8441	0.45632	0.55856	0.39981	0.56302
w_{25j}^{σ}	0.73872	0.52793	0.59326	0.29934	0.48675	0.3625	0.78677	0.66962	0.20516	0.23284	0.42707	0.80844
w_{26j}^{σ}	0.61391	0.43311	0.37988	0.62005	0.66347	0.74601	0.88641	0.85755	0.37266	0.70753	0.49367	0.89606
w_{27j}^{σ}	0.74407	0.29095	0.26163	0.33438	0.48822	0.50874	0.27485	0.2242	0.63642	0.45631	0.23343	0.78775
w_{28j}^{σ}	0.64204	0.29434	0.76627	0.74314	0.56229	0.28578	0.3005	0.61202	0.67657	0.77007	0.83325	0.7365
w_{29j}^{σ}	0.63301	0.86991	0.58624	0.58129	0.82293	0.48656	0.87967	0.24322	0.56221	0.64599	0.34009	0.3328
w_{30j}^{σ}	0.21028	0.2125	0.35686	0.2327	0.68048	0.28673	0.62334	0.87423	0.79115	0.85277	0.44287	0.70665
w_{31j}^{σ}	0.27346	0.53355	0.24499	0.36445	0.89143	0.75467	0.3448	0.4649	0.63987	0.30549	0.78761	0.69242
w_{32j}^{σ}	0.56929	0.52089	0.35212	0.74724	0.84092	0.48945	0.33867	0.55369	0.54172	0.52229	0.44351	0.43831
w_{33j}^{σ}	0.88758	0.79268	0.51741	0.59932	0.41293	0.27906	0.22506	0.79788	0.68731	0.31342	0.46256	0.20662
w_{34j}^{σ}	0.71341	0.65353	0.55258	0.75329	0.74653	0.76045	0.6211	0.5965	0.83381	0.87298	0.36568	0.36651

Πινάκας 3.3.1.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.22685	0.65138
2	0.14363	0.57961
3	0.85274	0.96495
4	0.1073	0.18704
5	0.84485	0.61712
6	0.2431	0.2305
7	0.16023	0.76713
8	0.52586	0.97526
9	0.3681	0.53785
10	0.89203	0.79547
11	0.48252	0.24796
12	0.44903	0.21492

Πινάκας 3.3.1.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.35879	0.65597
2	0.30054	0.60573
3	0.79692	0.87546
4	0.27511	0.33093
5	0.7914	0.63199
6	0.37017	0.36135
7	0.31216	0.73699
8	0.5681	0.88268
9	0.45767	0.5765
10	0.82442	0.75683
11	0.53777	0.37357
12	0.51432	0.35044

Πινάκας 3.3.1.5: Διασπορές των μετασυναπτικών βαρών

Τιμές αρχικές πριν την εκπαίδευση (τυχαίες), για το σύνολο με 7 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
0.35917	0.64388	0.70529	0.82015	0.64204	0.84929	0.64534	0.51983	0.50574	0.64949	0.55684	0.84314	0.39285	0.80736	0.27734	0.29728	0.22498

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.44378	0.26132	0.63034	0.38002	0.43799	0.82743	0.53251	0.79549	0.78117	0.54879	0.55726	0.78957	0.74248	0.77736	0.29169	0.80267	0.37728

Πινάκας 3.3.1.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^c	0.22685	0.65138	0.14363	0.57961	0.85274	0.96495	0.1073
w_{2j}^c	0	0	0	0	0	0	0
w_{3j}^c	0.051712	0.95052	-0.2638	0.075709	0.78406	0.59094	-0.034952
w_{4j}^c	-0.50409	-0.10193	-0.57016	-0.17834	0.29808	-0.54312	-0.23661
w_{5j}^c	0.0024201	0.012151	0.38707	0.81018	0.26977	-0.2796	-0.37144
w_{6j}^c	-0.50013	0.60256	0.19058	0.31279	-0.13745	-0.76161	0.98629
w_{7j}^c	0.19307	0.16297	0.64495	0.6136	-0.056781	-0.85571	0.63158
w_{8j}^c	-0.98475	-0.60414	-0.47048	-0.95191	0.53774	0.30936	0.27435
w_{9j}^c	0.81735	-0.12606	0.83242	0.97313	-0.59127	-0.57679	-0.52223
w_{10j}^c	0.29083	-0.71594	0.012791	-0.87818	-0.67335	-0.66644	-0.011
w_{11j}^c	0.31441	0.41086	-0.13181	0.31339	0.012918	-0.0053428	-0.65367
w_{12j}^c	-0.51123	0.74775	-0.98667	-0.31366	-0.10827	0.18113	-0.61351
w_{13j}^c	0.75294	0.40286	0.038683	-0.83531	-0.59572	-0.89717	0.47359
w_{14j}^c	-0.7601	0.34764	0.24181	-0.93466	-0.49183	0.97588	-0.30808
w_{15j}^c	0.86051	-0.094118	0.34282	0.65655	-0.8696	-0.16329	-0.5154
w_{16j}^c	0.53758	-0.14957	0.16478	0.40552	0.55081	0.29886	-0.83012
w_{17j}^c	0.22056	-0.067924	-0.37259	-0.35584	-0.8168	0.39073	0.77397
w_{18j}^c	0.6552	0.62586	-0.15574	0.66769	0.4086	-0.7982	-0.59265

w_{19j}^{c}	-0.68081	0.11407	-0.75903	0.11034	0.093903	0.79187	-0.4813
w_{20j}^{c}	-0.11303	-0.766	-0.44731	0.28505	0.99775	0.52444	-0.95835
w_{21j}^{c}	-0.98639	-0.75503	0.89136	-0.54173	-0.99698	-0.91239	0.67593
w_{22j}^{c}	0.84727	-0.30001	-0.92188	0.34459	0.20076	-0.58784	-0.079097
w_{23j}^{c}	-0.5235	-0.67937	-0.06001	0.89096	0.01158	-0.6394	0.58097
w_{24j}^{c}	-0.64624	-0.05216	0.5007	-0.054419	0.69138	-0.055999	0.30255
w_{25j}^{c}	0.073356	0.067208	-0.38638	-0.048075	0.44881	-0.84919	0.6895
w_{26j}^{c}	-0.18382	-0.30938	0.1766	0.69229	0.52674	-0.076093	0.52427
w_{27j}^{c}	-0.74545	-0.8569	-0.85564	-0.61388	-0.168	-0.13735	0.24406
w_{28j}^{c}	0.025618	0.17582	-0.78859	-0.20709	0.63365	-0.35462	0.68411
w_{29j}^{c}	-0.84425	-0.14664	0.20791	-0.60244	-0.92581	-0.32871	0.80215
w_{30j}^{c}	-0.5205	0.45337	0.56222	-0.3648	-0.36397	0.76783	-0.47317
w_{31j}^{c}	-0.062811	-0.37646	-0.76305	0.97588	-0.28433	0.19711	0.71922
w_{32j}^{c}	-0.70841	-0.86797	-0.48453	0.1359	0.33618	0.26449	-0.046972
w_{33j}^{c}	0.15859	-0.82021	-0.77247	0.80327	0.88142	0.88405	-0.11481
w_{34j}^{c}	0.45283	-0.75248	0.30491	-0.27451	0.2643	0.11054	-0.43245

Πινάκας 3.3.1.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{σ}	0.35879	0.65597	0.30054	0.60573	0.79692	0.87546	0.27511
w_{2j}^{σ}	0.33093	0.7914	0.63199	0.37017	0.36135	0.31216	0.73699
w_{3j}^{σ}	0.5681	0.88268	0.45767	0.5765	0.82442	0.75683	0.53777
w_{4j}^{σ}	0.37357	0.51432	0.35044	0.48758	0.65433	0.35991	0.46719
w_{5j}^{σ}	0.55085	0.55425	0.68548	0.83356	0.64442	0.45214	0.42
w_{6j}^{σ}	0.37495	0.7609	0.6167	0.65948	0.50189	0.28344	0.8952
w_{7j}^{σ}	0.61758	0.60704	0.77573	0.76476	0.53013	0.2505	0.77105
w_{8j}^{σ}	0.20534	0.33855	0.38533	0.21683	0.73821	0.65828	0.64602
w_{9j}^{σ}	0.83607	0.50588	0.84135	0.8906	0.34305	0.34812	0.36722
w_{10j}^{σ}	0.65179	0.29942	0.55448	0.24264	0.31433	0.31675	0.54615

w_{11j}^{σ}	0.66004	0.6938	0.50387	0.65969	0.55452	0.54813	0.32121
w_{12j}^{σ}	0.37107	0.81171	0.20466	0.44022	0.51211	0.61339	0.33527
w_{13j}^{σ}	0.81353	0.691	0.56354	0.25764	0.3415	0.23599	0.71576
w_{14j}^{σ}	0.28397	0.67167	0.63463	0.22287	0.37786	0.89156	0.44217
w_{15j}^{σ}	0.85118	0.51706	0.66999	0.77979	0.24564	0.49285	0.36961
w_{16j}^{σ}	0.73815	0.49765	0.60767	0.69193	0.74278	0.6546	0.25946
w_{17j}^{σ}	0.6272	0.52623	0.41959	0.42546	0.26412	0.68676	0.82089
w_{18j}^{σ}	0.77932	0.76905	0.49549	0.78369	0.69301	0.27063	0.34257
w_{19j}^{σ}	0.31172	0.58992	0.28434	0.58862	0.58287	0.82715	0.38155
w_{20j}^{σ}	0.51044	0.2819	0.39344	0.64977	0.89921	0.73356	0.21458
w_{21j}^{σ}	0.20476	0.28574	0.86198	0.36039	0.20106	0.23066	0.78657
w_{22j}^{σ}	0.84654	0.445	0.22734	0.67061	0.62027	0.34426	0.52232
w_{23j}^{σ}	0.36678	0.31222	0.529	0.86184	0.55405	0.32621	0.75334
w_{24j}^{σ}	0.32382	0.53174	0.72525	0.53095	0.79198	0.5304	0.65589
w_{25j}^{σ}	0.57567	0.57352	0.41477	0.53317	0.70708	0.25278	0.79133
w_{26j}^{σ}	0.48566	0.44172	0.61181	0.7923	0.73436	0.52337	0.73349
w_{27j}^{σ}	0.28909	0.25009	0.25053	0.33514	0.4912	0.50193	0.63542
w_{28j}^{σ}	0.55897	0.61154	0.27399	0.47752	0.77178	0.42588	0.78944
w_{29j}^{σ}	0.25451	0.49868	0.62277	0.33914	0.22597	0.43495	0.83075
w_{30j}^{σ}	0.36783	0.70868	0.74678	0.42232	0.42261	0.81874	0.38439
w_{31j}^{σ}	0.52802	0.41824	0.28293	0.89156	0.45049	0.61899	0.80173
w_{32j}^{σ}	0.30206	0.24621	0.38041	0.59757	0.66766	0.64257	0.53356
w_{33j}^{σ}	0.60551	0.26293	0.27964	0.83115	0.8585	0.85942	0.50982
w_{34j}^{σ}	0.70849	0.28663	0.65672	0.45392	0.6425	0.58869	0.39864

Πινάκας 3.3.1.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.22685	0.65138
2	0.14363	0.57961
3	0.85274	0.96495
4	0.1073	0.18704
5	0.84485	0.61712
6	0.2431	0.2305
7	0.16023	0.76713

Πινάκας 3.3.1.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.35879	0.65597
2	0.30054	0.60573
3	0.79692	0.87546
4	0.27511	0.33093
5	0.7914	0.63199
6	0.37017	0.36135
7	0.31216	0.73699

Πινάκας 3.3.1.10: Διασπορές των μετασυναπτικών βαρών

Τιμές αρχικές πριν την εκπαίδευση (τυχαίες), για το σύνολο με 17 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
0.35917	0.64388	0.70529	0.82015	0.64204	0.84929	0.64534	0.51983	0.50574	0.64949	0.55684	0.84314	0.39285	0.80736	0.27734	0.29728	0.22498

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.44378	0.26132	0.63034	0.38002	0.43799	0.82743	0.53251	0.79549	0.78117	0.54879	0.55726	0.78957	0.74248	0.77736	0.29169	0.80267	0.37728

Πινάκας 3.3.1.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
w_{1j}^c	0.22685	0.65138	0.14363	0.57961	0.85274	0.96495	0.1073	0.18704	0.84485	0.61712	0.2431	0.2305	0.16023	0.76713	0.52586	0.97526	0.3681
w_{2j}^c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w_{3j}^c	-0.37144	-0.50013	0.60256	0.19058	0.31279	-0.13745	-0.76161	0.98629	0.19307	0.16297	0.64495	0.6136	-0.05678	-0.85571	0.63158	-0.98475	-0.60414
w_{4j}^c	-0.47048	-0.95191	0.53774	0.30936	0.27435	0.81735	-0.12606	0.83242	0.97313	-0.59127	-0.57679	-0.52223	0.29083	-0.71594	0.012791	-0.87818	-0.67335
w_{5j}^c	-0.66644	-0.011	0.31441	0.41086	-0.13181	0.31339	0.012918	-0.0053428	-0.65367	-0.51123	0.74775	-0.98667	-0.31366	-0.10827	0.18113	-0.61351	0.75294
w_{6j}^c	0.40286	0.038683	-0.83531	-0.59572	-0.89717	0.47359	-0.7601	0.34764	0.24181	-0.93466	-0.49183	0.97588	-0.30808	0.86051	-0.09412	0.34282	0.65655
w_{7j}^c	-0.8696	-0.16329	-0.5154	0.53758	-0.14957	0.16478	0.40552	0.55081	0.29886	-0.83012	0.22056	-0.067924	-0.37259	-0.35584	-0.8168	0.39073	0.77397
w_{8j}^c	0.6552	0.62586	-0.15574	0.66769	0.4086	-0.7982	-0.59265	-0.68081	0.11407	-0.75903	0.11034	0.093903	0.79187	-0.4813	-0.11303	-0.766	-0.44731
w_{9j}^c	0.28505	0.99775	0.52444	-0.95835	-0.98639	-0.75503	0.89136	-0.54173	-0.99698	-0.91239	0.67593	0.84727	-0.30001	-0.92188	0.34459	0.20076	-0.58784
w_{10j}^c	-0.079097	-0.5235	-0.67937	-0.06001	0.89096	0.01158	-0.6394	0.58097	-0.64624	-0.05216	0.5007	-0.054419	0.69138	-0.056	0.30255	0.073356	0.067208
w_{11j}^c	-0.38638	-0.048075	0.44881	-0.84919	0.6895	-0.18382	-0.30938	0.1766	0.69229	0.52674	-0.076093	0.52427	-0.74545	-0.8569	-0.85564	-0.61388	-0.168
w_{12j}^c	-0.13735	0.24406	0.025618	0.17582	-0.78859	-0.20709	0.63365	-0.35462	0.68411	-0.84425	-0.14664	0.20791	-0.60244	-0.92581	-0.32871	0.80215	-0.5205
w_{13j}^c	0.45337	0.56222	-0.3648	-0.36397	0.76783	-0.47317	-0.062811	-0.37646	-0.76305	0.97588	-0.28433	0.19711	0.71922	-0.70841	-0.86797	-0.48453	0.1359
w_{14j}^c	0.33618	0.26449	-0.046972	0.15859	-0.82021	-0.77247	0.80327	0.88142	0.88405	-0.11481	0.45283	-0.75248	0.30491	-0.27451	0.2643	0.11054	-0.43245
w_{15j}^c	-0.14376	-0.3737	-0.013839	0.11821	0.53711	-0.58748	0.78451	0.072282	-0.81918	-0.56279	0.11354	0.95485	-0.66394	0.40283	-0.89561	0.86949	0.80761
w_{16j}^c	0.3062	0.13409	-0.36716	0.77394	-0.69735	-0.69335	-0.5228	-0.46959	-0.37636	0.10306	0.55015	0.31602	0.30111	-0.77843	-0.37034	0.57445	0.17282
w_{17j}^c	-0.69637	-0.77467	-0.044953	0.92622	-0.022693	0.55044	0.3771	0.78362	0.44856	-0.27307	-0.43917	0.84028	-0.26765	0.024457	-0.42913	0.037188	0.53921

w_{18j}^{c}	-0.063049	0.12359	-0.71618	-0.18071	-0.53572	0.6765	0.34178	-0.98526	-0.90617	-0.35122	0.73841	0.1826	-0.33398	-0.48606	0.20015	0.32419	0.56004
w_{19j}^{c}	0.96117	0.87871	-0.50668	0.45008	-0.16093	0.98875	0.55449	-0.74015	-0.82393	-0.61604	-0.17651	-0.11788	-0.78615	-0.93085	0.24692	-0.26768	-0.90447
w_{20j}^{c}	0.6793	0.26296	-0.73047	0.61793	0.55184	0.035117	-0.75491	-0.71286	0.17721	0.36163	0.62876	0.80928	0.53285	0.23718	0.91403	0.10355	0.089406
w_{21j}^{c}	0.77979	-0.18127	0.9419	-0.87651	0.034893	0.27426	-0.59975	-0.62057	-0.97063	-0.96427	-0.55183	-0.90657	0.37281	-0.75221	0.20954	0.92637	0.68899
w_{22j}^{c}	0.86505	-0.30609	0.44758	-0.7901	-0.047007	-0.87147	-0.53013	0.97552	0.58478	-0.58628	-0.24313	0.25678	-0.69859	0.67888	0.40692	0.055125	-0.08316
w_{23j}^{c}	-0.56536	0.56353	0.8312	-0.17299	-0.6038	0.010553	-0.023649	-0.079169	-0.30425	-0.31911	0.96452	0.69338	-0.09311	0.14092	-0.39164	-0.77411	-0.9284
w_{24j}^{c}	0.70823	0.39233	-0.67594	-0.24984	-0.98108	0.46689	0.2958	0.0073709	0.58084	0.56151	0.60129	0.20315	0.13286	0.81088	0.9228	-0.52662	-0.52426
w_{25j}^{c}	-0.7129	-0.92131	0.86689	-0.15175	-0.10746	0.64535	0.24696	0.89326	0.19127	0.15293	0.27098	0.79124	0.74368	0.28509	-0.97067	-0.76516	-0.44232
w_{26j}^{c}	0.63986	-0.74533	0.5759	0.38106	-0.70156	0.40544	0.69739	-0.26639	-0.29964	0.38236	-0.054315	0.81369	0.1381	0.15447	-0.19557	0.71305	-0.43873
w_{27j}^{c}	-0.84891	0.89938	0.73476	-0.68035	-0.081126	0.39442	0.033223	-0.023367	-0.75193	-0.12809	0.28113	-0.6112	-0.26024	0.3866	-0.38231	0.87462	0.61588
w_{28j}^{c}	-0.4598	-0.25691	0.48434	-0.5336	0.14453	0.69854	0.13546	-0.41678	-0.67549	-0.39273	0.69233	0.54769	0.75772	0.45069	-0.05439	0.81907	-0.84445
w_{29j}^{c}	-0.1845	0.88663	0.21344	-0.93821	0.47098	-0.32639	-0.632	-0.42244	0.62241	0.25135	0.55313	0.71716	-0.26497	-0.28723	-0.47554	-0.24216	-0.06537
w_{30j}^{c}	-0.81981	-0.49978	-0.8331	-0.83558	-0.76384	-0.19614	-0.52704	0.51429	0.74829	-0.77047	0.79389	-0.56268	-0.6909	-0.23165	-0.35806	0.92983	0.45179
w_{31j}^{c}	-0.024319	-0.90238	0.17505	-0.69547	0.96488	0.47673	0.63669	0.4103	-0.21957	-0.78905	-0.86227	-0.93347	0.73366	0.82036	0.070936	0.7278	-0.82344
w_{32j}^{c}	0.57717	0.56149	-0.23265	0.14454	0.74297	-0.86159	-0.7929	0.31629	0.10763	0.27033	0.90319	-0.18067	-0.03387	-0.01758	-0.88669	0.55193	0.64031
w_{33j}^{c}	-0.30474	-0.62586	-0.10223	-0.81428	0.41258	-0.81844	0.97662	0.01954	-0.56634	-0.90652	0.905	-0.1364	0.87781	0.56424	0.93296	0.61732	-0.51845
w_{34j}^{c}	0.071787	0.77832	0.55862	0.36618	-0.086796	-0.0026182	-0.38355	-0.22832	0.42099	-0.18894	0.36695	-0.88556	0.29214	-0.552	0.87576	-0.84975	-0.77613

Πινάκας 3.3.1.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
w_{1j}^{σ}	0.35879	0.65597	0.30054	0.60573	0.79692	0.87546	0.27511	0.33093	0.7914	0.63199	0.37017	0.36135	0.31216	0.73699	0.5681	0.88268	0.45767
w_{2j}^{σ}	0.5765	0.82442	0.75683	0.53777	0.37357	0.51432	0.35044	0.48758	0.65433	0.35991	0.46719	0.55085	0.55425	0.68548	0.83356	0.64442	0.45214
w_{3j}^{σ}	0.42	0.37495	0.7609	0.6167	0.65948	0.50189	0.28344	0.8952	0.61758	0.60704	0.77573	0.76476	0.53013	0.2505	0.77105	0.20534	0.33855
w_{4j}^{σ}	0.38533	0.21683	0.73821	0.65828	0.64602	0.83607	0.50588	0.84135	0.8906	0.34305	0.34812	0.36722	0.65179	0.29942	0.55448	0.24264	0.31433
w_{5j}^{σ}	0.31675	0.54615	0.66004	0.6938	0.50387	0.65969	0.55452	0.54813	0.32121	0.37107	0.81171	0.20466	0.44022	0.51211	0.61339	0.33527	0.81353
w_{6j}^{σ}	0.691	0.56354	0.25764	0.3415	0.23599	0.71576	0.28397	0.67167	0.63463	0.22287	0.37786	0.89156	0.44217	0.85118	0.51706	0.66999	0.77979
w_{7j}^{σ}	0.24564	0.49285	0.36961	0.73815	0.49765	0.60767	0.69193	0.74278	0.6546	0.25946	0.6272	0.52623	0.41959	0.42546	0.26412	0.68676	0.82089
w_{8j}^{σ}	0.77932	0.76905	0.49549	0.78369	0.69301	0.27063	0.34257	0.31172	0.58992	0.28434	0.58862	0.58287	0.82715	0.38155	0.51044	0.2819	0.39344
w_{9j}^{σ}	0.64977	0.89921	0.73356	0.21458	0.20476	0.28574	0.86198	0.36039	0.20106	0.23066	0.78657	0.84654	0.445	0.22734	0.67061	0.62027	0.34426
w_{10j}^{σ}	0.52232	0.36678	0.31222	0.529	0.86184	0.55405	0.32621	0.75334	0.32382	0.53174	0.72525	0.53095	0.79198	0.5304	0.65589	0.57567	0.57352

w_{11j}^{σ}	0.41477	0.53317	0.70708	0.25278	0.79133	0.48566	0.44172	0.61181	0.7923	0.73436	0.52337	0.73349	0.28909	0.25009	0.25053	0.33514	0.4912
w_{12j}^{σ}	0.50193	0.63542	0.55897	0.61154	0.27399	0.47752	0.77178	0.42588	0.78944	0.25451	0.49868	0.62277	0.33914	0.22597	0.43495	0.83075	0.36783
w_{13j}^{σ}	0.70868	0.74678	0.42232	0.42261	0.81874	0.38439	0.52802	0.41824	0.28293	0.89156	0.45049	0.61899	0.80173	0.30206	0.24621	0.38041	0.59757
w_{14j}^{σ}	0.66766	0.64257	0.53356	0.60551	0.26293	0.27964	0.83115	0.8585	0.85942	0.50982	0.70849	0.28663	0.65672	0.45392	0.6425	0.58869	0.39864
w_{15j}^{σ}	0.49968	0.41921	0.54516	0.59137	0.73799	0.34438	0.82458	0.5753	0.26329	0.35302	0.58974	0.8842	0.31762	0.69099	0.23654	0.85432	0.83266
w_{16j}^{σ}	0.65717	0.59693	0.42149	0.82088	0.30593	0.30733	0.36702	0.38564	0.41828	0.58607	0.74255	0.66061	0.65539	0.27755	0.42038	0.75106	0.61049
w_{17j}^{σ}	0.30627	0.27887	0.53427	0.87418	0.54206	0.74265	0.68199	0.82427	0.707	0.45442	0.39629	0.8441	0.45632	0.55856	0.39981	0.56302	0.73872
w_{18j}^{σ}	0.52793	0.59326	0.29934	0.48675	0.3625	0.78677	0.66962	0.20516	0.23284	0.42707	0.80844	0.61391	0.43311	0.37988	0.62005	0.66347	0.74601
w_{19j}^{σ}	0.88641	0.85755	0.37266	0.70753	0.49367	0.89606	0.74407	0.29095	0.26163	0.33438	0.48822	0.50874	0.27485	0.2242	0.63642	0.45631	0.23343
w_{20j}^{σ}	0.78775	0.64204	0.29434	0.76627	0.74314	0.56229	0.28578	0.3005	0.61202	0.67657	0.77007	0.83325	0.7365	0.63301	0.86991	0.58624	0.58129
w_{21j}^{σ}	0.82293	0.48656	0.87967	0.24322	0.56221	0.64599	0.34009	0.3328	0.21028	0.2125	0.35686	0.2327	0.68048	0.28673	0.62334	0.87423	0.79115
w_{22j}^{σ}	0.85277	0.44287	0.70665	0.27346	0.53355	0.24499	0.36445	0.89143	0.75467	0.3448	0.4649	0.63987	0.30549	0.78761	0.69242	0.56929	0.52089
w_{23j}^{σ}	0.35212	0.74724	0.84092	0.48945	0.33867	0.55369	0.54172	0.52229	0.44351	0.43831	0.88758	0.79268	0.51741	0.59932	0.41293	0.27906	0.22506
w_{24j}^{σ}	0.79788	0.68731	0.31342	0.46256	0.20662	0.71341	0.65353	0.55258	0.75329	0.74653	0.76045	0.6211	0.5965	0.83381	0.87298	0.36568	0.36651
w_{25j}^{σ}	0.30048	0.22754	0.85341	0.49689	0.51239	0.77587	0.63644	0.86264	0.61695	0.60353	0.64484	0.82694	0.81029	0.64978	0.21027	0.2822	0.39519
w_{26j}^{σ}	0.77395	0.28914	0.75157	0.68337	0.30446	0.6919	0.79409	0.45676	0.44513	0.68383	0.53099	0.83479	0.59834	0.60406	0.48155	0.79957	0.39644
w_{27j}^{σ}	0.25288	0.86478	0.80717	0.31188	0.52161	0.68805	0.56163	0.54182	0.28683	0.50517	0.6484	0.33608	0.45892	0.68531	0.41619	0.85612	0.76556
w_{28j}^{σ}	0.38907	0.46008	0.71952	0.36324	0.60058	0.79449	0.59741	0.40413	0.31358	0.41254	0.79231	0.74169	0.8152	0.70774	0.53096	0.83668	0.25444
w_{29j}^{σ}	0.48542	0.86032	0.6247	0.22163	0.71484	0.43576	0.3288	0.40214	0.76784	0.63797	0.7436	0.801	0.45726	0.44947	0.38356	0.46524	0.52712
w_{30j}^{σ}	0.26306	0.37508	0.25841	0.25755	0.28266	0.48135	0.36554	0.73	0.8119	0.28033	0.82786	0.35306	0.30819	0.46892	0.42468	0.87544	0.70813
w_{31j}^{σ}	0.54149	0.23417	0.61127	0.30658	0.88771	0.71686	0.77284	0.69361	0.47315	0.27383	0.24821	0.22329	0.80678	0.83712	0.57483	0.80473	0.2618
w_{32j}^{σ}	0.75201	0.74652	0.46857	0.60059	0.81004	0.24845	0.27248	0.6607	0.58767	0.64462	0.86612	0.48676	0.53815	0.54385	0.23966	0.74318	0.77411
w_{33j}^{σ}	0.44334	0.33095	0.51422	0.265	0.6944	0.26355	0.89182	0.55684	0.35178	0.23272	0.86675	0.50226	0.85723	0.74748	0.87653	0.76606	0.36854
w_{34j}^{σ}	0.57513	0.82241	0.74552	0.67816	0.51962	0.54908	0.41576	0.47009	0.69735	0.48387	0.67843	0.24005	0.65225	0.3568	0.85652	0.25259	0.27835

Πινάκας 3.3.1.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.22685	0.65138
2	0.14363	0.57961
3	0.85274	0.96495
4	0.1073	0.18704
5	0.84485	0.61712
6	0.2431	0.2305
7	0.16023	0.76713
8	0.52586	0.97526
9	0.3681	0.53785
10	0.89203	0.79547
11	0.48252	0.24796
12	0.44903	0.21492
13	0.41083	0.64904
14	0.22844	0.3817
15	0.50121	0.50608
16	0.69354	0.90509
17	0.63488	0.3602

Πινάκας 3.3.1.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.35879	0.65597
2	0.30054	0.60573
3	0.79692	0.87546
4	0.27511	0.33093
5	0.7914	0.63199
6	0.37017	0.36135
7	0.31216	0.73699
8	0.5681	0.88268
9	0.45767	0.5765
10	0.82442	0.75683
11	0.53777	0.37357
12	0.51432	0.35044
13	0.48758	0.65433
14	0.35991	0.46719
15	0.55085	0.55425
16	0.68548	0.83356
17	0.64442	0.45214

Πινάκας 3.3.1.15: Διασπορές των μετασυναπτικών βαρών

Π 3.3.2 Τιμές μετά την εκπαίδευση

για το σύνολο με 12 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
0.33512	1.0022	0.28616	0.79336	0.13904	0.087306	0.1747	0.37379	0.061541	0.54681	0.54141	0.95574	0.60343	0.32174	0.77063	0.69386	0.31756

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.40147	0.86525	0.29722	0.4842	1.2227	0.94226	0.39459	1.4269	0.29895	0.37966	1.8645	0.47057	1.2366	1.3898	0.35636	0.069182	0.50056

Πινάκας 3.3.2.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12
w_{1j}^c	1.3745	0.35059	0.15233	0.39047	0.85091	0.95222	0.061085	0.2493	1.3458	0.4538	1.1776	0.15206
w_{2j}^c	0	0	0	0	0	0	0	0	0	0	0	0
w_{3j}^c	1.1472	-0.07588	-0.53503	-0.43032	0.0018698	-0.045211	0.37203	0.093649	0.30674	-0.42261	0.74707	-0.4984
w_{4j}^c	0.3025	0.59861	0.31058	-0.65549	-0.76358	0.96133	0.20653	0.017991	0.013492	0.82258	0.31115	-0.85683
w_{5j}^c	0.052322	-0.93443	-0.60211	0.16888	-0.94526	0.51461	0.29338	-0.31973	0.74051	-0.89867	1.3929	1.0536
w_{6j}^c	-0.37765	-0.95163	-0.55565	1.0507	-0.72515	0.026322	-0.89689	-0.53493	0.18693	-0.082835	0.35517	0.40113
w_{7j}^c	0.79152	0.16086	0.0014454	0.25612	-0.69158	-0.52543	0.76118	-0.69128	0.48742	-0.43789	1.0869	-0.61696
w_{8j}^c	0.24383	0.25255	0.029272	-0.88421	-0.61312	-0.89446	0.47866	-0.96859	0.5128	-0.84262	0.1454	-0.43161
w_{9j}^c	1.3237	0.4261	0.86707	0.39182	0.35023	0.64416	-0.92329	0.11352	0.083493	0.31782	0.47186	0.17406
w_{10j}^c	0.40933	0.090169	0.31009	-1.0394	0.22662	-0.16225	-0.39152	-0.075852	0.37308	0.19396	0.65608	0.66718
w_{11j}^c	0.62344	0.033092	0.67194	0.6194	-0.78232	-0.48049	-0.71631	0.28298	-0.2033	-0.2052	0.31395	0.80944
w_{12j}^c	-0.74642	-0.76676	-0.76244	-0.55123	0.29331	0.97631	0.53587	-0.81185	0.007683	-0.78037	0.60009	-0.58649
w_{13j}^c	-0.29883	-0.50272	0.68019	1.1189	-0.30815	-0.88835	0.33734	0.080413	-0.95532	0.043021	-0.55322	-0.66406
w_{14j}^c	-0.29466	0.88798	0.030709	-0.83935	0.58855	-0.6718	-0.048068	0.71106	0.14345	0.20945	0.1441	0.31201
w_{15j}^c	0.94767	0.23971	-0.37306	0.092902	0.43665	-0.8949	0.69131	-0.1706	-0.41256	0.07961	0.73054	0.61247
w_{16j}^c	0.26254	0.6653	-0.74719	-0.70668	-0.84823	-0.57749	-0.19499	0.29643	-0.47539	0.091722	0.68933	-0.836

w_{17j}^{c}	0.63979	1.054	-0.34887	0.97478	-0.86506	-0.17641	0.20267	-0.55793	-0.71808	-0.14764	0.4775	-0.51517
w_{18j}^{c}	0.71573	0.087109	-0.36999	-0.4625	0.7771	-0.47106	-0.042614	-0.05128	-0.35323	0.81286	0.67737	0.15171
w_{19j}^{c}	0.34388	-0.30542	-0.87812	-0.025748	0.12391	0.46957	0.26378	0.38363	-0.53121	-0.62918	-0.13688	0.81965
w_{20j}^{c}	0.0086114	0.6122	-0.088003	0.65387	-0.76653	0.38679	-0.27433	0.57244	-0.0059119	-0.56121	0.21954	-0.5267
w_{21j}^{c}	1.0068	0.25807	0.53239	-0.14606	0.79862	0.159	-0.87256	-0.095518	0.46334	0.46816	-0.85819	0.49711
w_{22j}^{c}	-1.0314	0.92002	0.81846	0.17522	0.12341	-0.2722	0.78903	-0.46437	-1.047	-0.29131	-0.5505	-0.56936
w_{23j}^{c}	0.48399	1.2107	0.33459	0.36939	-0.79351	-0.33259	0.57708	-0.31451	-0.19698	-0.57756	0.27923	0.92956
w_{24j}^{c}	0.20023	0.43894	0.38058	0.74138	0.46153	-0.11937	-0.46185	0.6988	-0.090144	0.36159	-0.15008	0.020313
w_{25j}^{c}	0.37094	0.18563	0.13164	-0.45717	-0.18153	-0.59608	0.68885	0.053953	-0.64595	-0.73303	-0.12251	0.71676
w_{26j}^{c}	0.50597	-0.26094	-0.47635	0.31647	0.33103	0.56965	0.97393	0.81276	-0.0016063	0.27965	0.059198	1.0284
w_{27j}^{c}	0.17517	-0.12917	-0.85114	-0.27757	-0.17525	-0.11653	-0.82457	-0.40747	-0.02319	0.0029811	-0.67321	0.63397
w_{28j}^{c}	0.10961	-0.77312	0.64505	0.93176	0.047949	-0.78981	-0.73803	0.071486	-1.0609	0.34575	0.70586	0.54092
w_{29j}^{c}	1.0629	0.65107	0.095478	0.098822	0.7915	-0.27783	0.95372	-0.80938	0.44316	0.068425	0.49615	-0.69679
w_{30j}^{c}	-0.42482	-0.6559	-0.56961	-1.1877	0.38217	-0.69492	0.22357	0.8146	0.73489	0.70358	-0.17341	0.5016
w_{31j}^{c}	-0.76107	0.22775	-0.87469	-0.31881	0.97879	0.55794	-0.60888	-0.30105	0.044182	-0.46159	0.37485	0.45863
w_{32j}^{c}	-0.041675	-0.2142	-0.52728	0.47522	0.83012	-0.13823	-0.63577	-0.38681	0.049493	0.010894	-0.20825	-0.56594
w_{33j}^{c}	0.619	1.0825	-0.10034	1.5587	-0.37656	-0.78205	-0.95728	0.4471	0.37362	-0.030901	-0.40777	-1.1015
w_{34j}^{c}	-0.48422	1.0476	0.0014785	1.0679	0.56525	0.5762	0.20587	0.36827	0.28303	0.79452	-0.68508	-0.57573

Πινάκας 3.3.2.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12
w_{1j}^{σ}	0.42972	0.90886	0.28018	0.88614	0.8024	0.89308	0.20953	0.85994	0.81543	0.9287	0.57247	0.27695
w_{2j}^{σ}	0.51477	0.83301	0.55573	0.97192	0.44671	0.57609	0.83296	0.85455	0.54481	0.9639	0.60054	0.23988
w_{3j}^{σ}	0.49358	0.64979	0.3526	0.7528	0.54137	0.56966	0.68292	1.1048	0.28551	0.48514	0.79943	0.35629
w_{4j}^{σ}	0.56708	0.9821	0.65055	0.52894	0.2675	0.88766	0.61273	0.77558	0.54556	1.0665	0.48927	0.049509
w_{5j}^{σ}	3.2519	1.9983	0.30125	1.1993	0.21459	0.77432	0.65411	0.46972	0.56591	0.58082	18.1614	0.80338
w_{6j}^{σ}	0.4142	1.5116	0.3327	0.55147	0.28777	0.54723	0.21529	0.83355	17.1788	1.279	0.08924	0.67011
w_{7j}^{σ}	0.54546	0.98179	0.54013	1.0518	0.23014	0.31735	0.81044	0.49864	0.15172	0.51471	0.49272	0.29631
w_{8j}^{σ}	0.52578	1.297	0.55113	1.0653	0.31991	0.19246	0.72662	0.64229	0.34838	0.57098	0.71846	0.30569

w_{9j}^{σ}	0.24468	1.418	0.84224	0.47428	0.67601	0.78164	0.13505	0.73561	1.0656	1.0733	1.1385	0.58375
w_{10j}^{σ}	0.61358	1.2764	0.64295	0.74124	0.62319	0.51948	0.41007	0.71689	0.52946	1.2885	0.59196	0.78168
w_{11j}^{σ}	0.27283	1.2923	0.78147	0.96357	0.25893	0.3478	0.26417	0.51221	1.0468	0.69567	0.90615	0.83657
w_{12j}^{σ}	0.33103	0.97642	0.26824	0.5226	0.64609	0.92803	0.73127	0.80589	0.49898	0.80508	0.76013	0.27755
w_{13j}^{σ}	0.99764	1.0817	0.77856	0.87537	0.43948	0.22189	0.67995	0.82923	0.65907	0.70457	0.72459	0.38019
w_{14j}^{σ}	0.7236	0.94327	0.54531	0.73945	0.7481	0.32123	0.51745	0.52166	0.37302	1.1026	0.32131	0.60594
w_{15j}^{σ}	4.751	0.93493	0.39935	1.1415	0.7037	0.25804	0.80196	0.56664	0.61694	0.78728	0.8144	0.6145
w_{16j}^{σ}	0.52763	1.2049	0.27937	1.3392	0.23069	0.31929	0.47523	0.70375	0.36642	0.89264	0.57739	0.13403
w_{17j}^{σ}	0.63103	0.32489	0.42338	0.77478	0.23162	0.48793	0.63453	0.57293	1.1472	0.8147	1.1283	0.56683
w_{18j}^{σ}	0.65065	1.2755	0.40958	0.87771	0.81562	0.41412	0.5224	1.1529	0.24467	0.96129	0.55633	0.65529
w_{19j}^{σ}	0.39941	1.0176	0.18865	1.1387	0.58903	0.5739	0.65632	1.1806	0.74105	1.2378	0.74576	0.85464
w_{20j}^{σ}	0.30128	1.3699	0.49764	0.8791	0.27528	0.63187	0.43588	0.52773	0.25096	1.2525	0.56789	0.35202
w_{21j}^{σ}	0.40423	0.88102	0.73812	1.1634	0.81146	0.48467	0.1684	1.8653	0.47686	1.2787	3.1582	0.41577
w_{22j}^{σ}	0.81308	0.64423	0.82144	0.77296	0.58937	0.40362	0.81426	0.66593	0.46751	0.9187	0.75294	0.17306
w_{23j}^{σ}	0.39664	0.6402	0.65869	0.58506	0.2609	0.4085	0.74121	0.70946	0.6225	0.63638	0.47941	0.84763
w_{24j}^{σ}	0.26063	1.4751	0.70318	1.1667	0.6996	0.44494	0.38481	0.78147	0.40173	0.70634	0.78724	0.55664
w_{25j}^{σ}	1.0685	0.70744	0.58407	0.64737	0.47758	0.3607	0.78409	0.82349	0.50731	0.89739	0.53403	0.79536
w_{26j}^{σ}	0.84013	0.29547	0.36832	0.6025	0.66092	0.72675	0.87816	0.89027	0.31065	1.1173	0.76545	0.91412
w_{27j}^{σ}	0.46088	1.5516	0.15119	0.82214	0.47743	0.53555	0.2171	1.3309	0.35049	1.252	0.8834	0.71578
w_{28j}^{σ}	0.50356	0.83111	0.73361	0.92033	0.55438	0.29442	0.27997	0.94603	2.5569	1.2615	0.93109	0.7486
w_{29j}^{σ}	0.33272	1.207	0.58126	0.34199	0.81591	0.46755	0.87636	0.83168	0.50708	1.0166	0.89868	0.1739
w_{30j}^{σ}	0.38912	0.89937	0.33967	0.51301	0.67663	0.30334	0.62079	1.1018	0.92013	1.1992	0.48311	0.66481
w_{31j}^{σ}	0.47181	0.67687	0.22509	0.5778	0.88833	0.75477	0.32034	0.80394	0.61811	0.88726	0.98628	0.66699
w_{32j}^{σ}	0.95622	0.74489	0.32055	0.73715	0.84871	0.4946	0.32292	0.92339	0.34699	1.0893	0.72752	0.16134
w_{33j}^{σ}	1.1514	0.76378	0.50658	0.001	0.41196	0.2038	0.18355	0.94511	0.8784	0.95673	0.67952	0.0011053
w_{34j}^{σ}	0.63988	0.57049	0.56277	0.66221	0.75217	0.7916	0.63484	0.53678	1.4302	0.76962	0.44714	0.23667

Πινάκας 3.3.2.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.96756	0.033
2	-0.012424	1.0124
3	0.77121	0.99979
4	0.041937	0.96334
5	0.65014	0.70717
6	0.092798	0.60794
7	0.16877	0.74663
8	-0.016587	1.015
9	0.9937	0.0054719
10	-0.0048589	1.0055
11	1.0021	-0.0022362
12	0.81033	0.12622

Πινάκας 3.3.2.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.41276	0.4847
2	0.59148	0.68325
3	0.773	0.88886
4	0.38468	0.43562
5	0.78296	0.63462
6	0.41734	0.30647
7	0.30038	0.73061
8	0.73057	0.95346
9	0.40971	0.46221
10	0.97265	0.988
11	0.47902	0.51687
12	0.46546	0.31132

Πινάκας 3.3.2.5: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την εκπαίδευση ,
για το σύνολο με 7 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
239.8863	0.48646	0.21188	1.416	0.0006401	0.47981	0.19045	0.22632	0.69829	0.72676	0.89367	2.0813	0.13647	0.16834	0.89989	0.40583	0.070637

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.72446	0.44127	0.94302	1.4581	0.12919	0.89055	0.20628	0.46604	0.82118	0.47519	0.0061385	0.88843	1.4886	0.94564	1.2633	0.95628	0.84499

Πινάκας 3.3.2.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^c	0.27667	0.7569	-0.018165	1.5306	1.4776	0.91854	81.5753
w_{2j}^c	0	0	0	0	0	0	0
w_{3j}^c	-0.55484	1.4908	-0.29226	0.61953	0.63199	0.1041	-0.024884
w_{4j}^c	-0.99953	0.30118	-0.55827	-0.22813	0.093087	-0.88879	-0.2817
w_{5j}^c	-0.80001	0.23281	-0.27754	0.48392	0.25782	-0.33109	-0.45766
w_{6j}^c	-0.23216	0.6895	-0.033687	-0.31516	-0.47705	-1.2191	0.97425
w_{7j}^c	-0.39417	0.71291	0.20356	0.9624	0.21583	-0.70875	0.65041
w_{8j}^c	-1.3257	-0.33712	-0.76711	0.46773	0.38621	-0.17194	0.19211
w_{9j}^c	0.00017458	-0.016191	0.60883	1.1107	-0.24914	-0.13319	-0.59691
w_{10j}^c	1.026	-0.76312	0.066163	0.45689	-0.85556	-1.0503	-0.07941
w_{11j}^c	0.0019727	0.61122	-0.058942	0.56781	0.35349	0.19455	-0.59887
w_{12j}^c	-0.8415	0.73589	-0.87383	-0.036876	-0.42952	0.063741	-0.63088
w_{13j}^c	0.74076	0.68486	-0.070407	-0.88378	-0.77922	0.24515	0.68477
w_{14j}^c	-0.91573	0.36606	0.18098	-0.035231	-0.64311	0.46811	-0.31542
w_{15j}^c	1.0418	0.090267	0.053554	0.60958	-0.62398	0.089492	-0.5757
w_{16j}^c	0.57384	0.21316	0.2173	0.24603	0.35781	-0.074963	-1.0086
w_{17j}^c	0.6893	-0.23087	-0.40772	-0.4983	-0.51121	0.22757	1.0212

w_{18j}^{c}	0.77413	0.76322	0.036397	-0.0089302	0.17069	-1.0697	-0.8527
w_{19j}^{c}	-0.67961	0.14322	-0.59809	0.0010396	0.63281	1.0181	-0.74032
w_{20j}^{c}	-0.089918	-0.74642	-0.30887	-0.053651	0.87346	0.24903	-0.89695
w_{21j}^{c}	-0.61458	-0.68453	0.83181	-0.77472	-0.6793	-0.81935	0.72619
w_{22j}^{c}	0.75884	-0.44806	-0.85144	-0.72351	-0.12472	0.14093	0.13643
w_{23j}^{c}	0.011742	-0.48986	-0.10062	0.29968	0.26503	-0.54295	0.72628
w_{24j}^{c}	-0.62317	-0.25483	0.64085	-0.13332	0.57822	0.54227	0.39354
w_{25j}^{c}	-0.06158	0.61919	-0.22523	0.39583	0.6868	-0.45389	0.83214
w_{26j}^{c}	0.3571	-0.38936	0.23403	0.33634	0.8185	-0.63169	0.56603
w_{27j}^{c}	0.41593	-0.90262	-0.65643	-0.85143	0.080771	0.78292	0.23356
w_{28j}^{c}	0.77809	0.0018602	-1.1084	0.15338	0.66949	-0.85928	0.9646
w_{29j}^{c}	-0.79003	-0.060249	0.156	0.69313	-0.81739	-0.26615	0.81208
w_{30j}^{c}	-0.37334	0.59205	0.63741	-0.27623	-0.44448	0.49704	-0.59655
w_{31j}^{c}	0.52943	-0.14072	-0.82954	0.51265	0.13283	-0.017969	0.82825
w_{32j}^{c}	-0.10417	-0.8402	-0.19218	-0.66591	0.19108	-0.02511	-0.099012
w_{33j}^{c}	0.9225	-0.7008	-0.77433	0.33174	1.1165	0.88307	-0.19546
w_{34j}^{c}	0.87685	-0.81627	0.42482	-0.94417	0.2552	0.28794	-0.5452

Πινάκας 3.3.2.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{σ}	1.2154	0.52967	2.7081	0.81413	3.5402e-005	1.0847	269.4827
w_{2j}^{σ}	0.60183	0.86232	0.87646	1.0851	0.29238	0.66489	0.85496
w_{3j}^{σ}	0.54439	0.00037334	0.28354	0.167	0.74061	0.97511	0.68435
w_{4j}^{σ}	1.3652	0.40733	0.70889	25.4521	0.78911	1.3889	0.4846
w_{5j}^{σ}	0.49264	0.64516	0.51505	0.20978	0.76371	1.4605	0.49032
w_{6j}^{σ}	1.8196	0.63863	0.63893	1.1717	0.30365	0.615	1.0627
w_{7j}^{σ}	0.36576	0.28342	0.99261	0.39634	0.16951	0.48669	0.85967
w_{8j}^{σ}	0.88655	0.37532	0.24486	0.24762	0.81006	1.1395	0.59069
w_{9j}^{σ}	0.69851	0.41735	1.0268	0.82318	0.54483	0.27872	0.21045

w_{10j}^{σ}	0.73182	0.36035	0.67123	0.74964	0.2625	0.79512	0.45751
w_{11j}^{σ}	0.89218	0.783	0.52827	0.9028	0.45698	0.89367	0.38739
w_{12j}^{σ}	1.0558	0.85788	0.38107	2.0813	0.48336	1.044	0.18064
w_{13j}^{σ}	1.0135	0.72851	0.42219	0.95795	10.3661	0.13648	0.59428
w_{14j}^{σ}	0.89808	0.62488	0.65764	0.19558	0.35062	0.16167	0.33862
w_{15j}^{σ}	0.94946	0.37512	0.59328	0.9225	0.89209	0.89989	0.17289
w_{16j}^{σ}	1.1768	0.55573	0.54539	1.1173	1.0303	0.8807	0.24924
w_{17j}^{σ}	0.77947	0.4654	0.43904	0.71718	0.69836	1.531	0.6729
w_{18j}^{σ}	0.79427	0.63283	0.49864	0.72446	0.66008	0.90009	0.11897
w_{19j}^{σ}	0.83159	0.78023	0.75952	0.83099	0.53441	0.45318	0.49902
w_{20j}^{σ}	0.4951	0.33995	0.41905	0.90748	1.1824	0.94302	0.010508
w_{21j}^{σ}	0.97195	0.30989	0.94583	0.89275	0.37461	0.62005	0.73587
w_{22j}^{σ}	0.85806	0.51985	0.62175	1.2386	0.45504	0.13932	0.46806
w_{23j}^{σ}	0.89302	0.34027	0.63566	0.89937	0.45853	0.9341	0.68885
w_{24j}^{σ}	1.1625	0.51907	0.55339	0.24611	1.0429	0.40775	0.67069
w_{25j}^{σ}	0.56635	0.54358	0.45068	0.46604	0.54186	1.4096	0.78116
w_{26j}^{σ}	0.77297	0.48163	0.72275	0.41746	0.63203	0.73605	0.74037
w_{27j}^{σ}	1.4016	0.33324	0.76706	0.39961	0.41624	1.0044	0.54729
w_{28j}^{σ}	0.73267	0.44884	0.38884	0.42293	0.63934	0.97509	0.59804
w_{29j}^{σ}	0.69527	0.4455	0.65634	0.78876	0.16022	0.83317	0.80847
w_{30j}^{σ}	1.1209	0.66163	0.73807	0.75677	0.24853	1.1236	0.23856
w_{31j}^{σ}	1.2374	0.40672	0.4815	0.87897	0.44421	1.0697	0.79499
w_{32j}^{σ}	0.99321	0.39926	0.53317	0.85558	0.68485	0.7858	0.40818
w_{33j}^{σ}	0.85656	0.46474	0.29145	0.95827	0.54157	0.84874	0.46321
w_{34j}^{σ}	0.77467	0.37935	0.56885	0.58226	0.69109	0.84499	0.00033412

Πινάκας 3.3.2.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	-0.0015955	1.0016
2	0.87579	0.09548
3	-0.0033415	1.0029
4	0.9861	0.013895
5	0.4866	0.49623
6	-0.012233	1.0122
7	0.14324	0.88272

Πινάκας 3.3.2.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.70947	0.75065
2	0.31646	0.58449
3	0.79374	0.89223
4	0.57946	0.61229
5	0.74524	0.61286
6	0.6848	0.72401
7	0.25154	0.68402

Πινάκας 3.3.2.10: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την εκπαίδευση ,
για το σύνολο με 17 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
0.58476	0.97318	0.00029978	0.00033362	0.045303	0.25946	0.23488	0.45432	0.40518	1.4733	0.74134	0.42718	0.55645	0.21955	0.2251	0.05107	0.57307

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.17227	0.83964	0.40109	0.80688	0.032481	1.1402	0.57366	0.6208	0.65495	0.38956	1.2151	0.5881	1.2858	1.5209	0.19663	0.83293	0.22555

Πινάκας 3.3.2.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
w_{1j}^c	0.36219	-0.019863	0.24263	0.58192	0.84818	0.97669	0.023594	0.22797	0.85078	0.61729	1.1296	0.25743	0.18292	0.76727	0.51223	0.88651	0.43243
w_{2j}^c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w_{3j}^c	-0.29261	-0.36622	1.046	0.18938	0.36185	-0.15191	-0.87235	0.94013	0.20136	0.16268	0.66907	0.68071	-0.17498	-0.85554	0.6358	-0.6332	-0.4841
w_{4j}^c	-0.34982	-0.92998	0.45605	0.31063	0.26643	0.83074	-0.11133	0.97892	0.97607	-0.59131	0.21718	-0.55364	0.21985	-0.71573	0.012739	-0.72137	-0.6039
w_{5j}^c	-0.6053	-0.21109	0.30585	0.40956	-0.10527	0.33884	-0.030621	0.19264	-0.62859	-0.51098	0.49789	-0.91012	0.30661	-0.10797	0.18819	-0.39675	0.76519
w_{6j}^c	0.20221	-0.4368	-0.75709	-0.59352	-0.94706	0.4628	-0.77557	0.2745	0.23881	-0.9344	0.32179	1.0107	-0.16456	0.86023	-0.10653	0.067505	0.65138
w_{7j}^c	-0.61443	-0.19279	-0.22714	0.53683	-0.20377	0.14427	0.36617	-0.36997	0.29785	-0.83027	0.8956	0.067111	-0.029394	-0.35596	-0.82469	0.15237	0.77879
w_{8j}^c	0.35353	0.78062	-0.34922	0.66871	0.39661	-0.81512	-0.5584	-1.1724	0.10503	-0.75839	0.50495	0.19685	0.81944	-0.48089	-0.14563	-0.90471	-0.42946
w_{9j}^c	-0.038174	0.83377	0.92631	-0.95755	-1.0232	-0.74234	0.89142	-0.11677	-0.99716	-0.91252	1.0616	0.86733	-0.3108	-0.92168	0.33957	-0.088803	-0.52541
w_{10j}^c	-0.035576	-0.85496	-0.79422	-0.058348	0.90496	0.022214	-0.71515	0.21327	-0.6446	-0.051826	1.2956	-0.010434	0.70646	-0.055806	0.2902	0.11578	0.021102
w_{11j}^c	-0.1659	0.19455	0.50096	-0.84794	0.71657	-0.19416	-0.36723	0.25782	0.71035	0.52648	-0.44542	0.55526	-0.84456	-0.85643	-0.86132	-0.0070249	-0.22528
w_{12j}^c	-0.2751	-0.40216	-0.072265	0.17498	-0.79383	-0.21588	0.66882	-0.2002	0.70238	-0.844	0.07265	0.25197	-0.25934	-0.92536	-0.33098	0.68245	-0.51837
w_{13j}^c	0.1832	0.24515	-0.46129	-0.35901	0.76628	-0.461	-0.1119	-0.026327	-0.78648	0.97565	-0.87454	0.31766	0.65787	-0.70792	-0.86832	-0.019922	0.18684
w_{14j}^c	0.46961	0.37635	-0.30796	0.16072	-0.88282	-0.76581	0.83793	0.75566	0.89247	-0.11478	-0.29986	-0.69699	0.10441	-0.27432	0.268	-0.094465	-0.40548
w_{15j}^c	0.23278	-0.071027	1.3429	0.11406	0.57384	-0.56355	0.78602	0.11054	-0.7795	-0.56225	0.14242	0.9967	-0.30729	0.40259	-0.89082	0.76562	0.85525
w_{16j}^c	0.54	0.15834	0.25299	0.77168	-0.75672	-0.74321	-0.61249	-0.81983	-0.36904	0.10309	-0.04282	0.39606	0.2204	-0.77719	-0.40458	0.36494	0.18799

w_{17j}^{c}	-0.60168	-0.40054	0.25243	0.92419	-0.066526	0.56532	0.35767	1.0201	0.46881	-0.27272	-0.50007	0.86555	-0.28996	0.024353	-0.43537	0.093986	0.49936
w_{18j}^{c}	0.31599	-0.027218	0.0083564	-0.18255	-0.50019	0.70299	0.3891	-0.80716	-0.87578	-0.35087	0.40316	0.31632	-0.27345	-0.48564	0.18951	0.7797	0.55855
w_{19j}^{c}	0.8678	0.91406	-0.47164	0.45139	-0.22045	0.98724	0.54326	-0.38913	-0.837	-0.61538	-0.42578	-0.24644	-0.78363	-0.92955	0.26249	-0.48547	-0.85664
w_{20j}^{c}	0.47242	0.26587	-0.75959	0.61973	0.54996	0.049469	-0.77622	-0.6386	0.17326	0.36143	-0.10634	0.83455	0.44525	0.23686	0.92412	0.68044	0.10145
w_{21j}^{c}	0.72428	0.24903	0.74422	-0.87273	0.056094	0.26039	-0.68037	-0.42724	-0.99157	-0.9644	-0.66383	-0.87219	0.47469	-0.75233	0.22576	1.0208	0.66304
w_{22j}^{c}	0.6777	-0.33448	-0.23854	-0.78895	-0.041185	-0.84918	-0.4842	0.95865	0.60424	-0.58673	-0.84303	0.17738	-0.50498	0.6785	0.42528	0.26177	-0.038077
w_{23j}^{c}	-0.51909	-0.089442	0.36236	-0.17489	-0.62116	0.0093595	-0.063116	0.55094	-0.30399	-0.31896	0.9444	0.67982	0.0015949	0.14092	-0.38185	-0.64907	-0.9929
w_{24j}^{c}	0.58323	0.29961	-0.51895	-0.25071	-1.0132	0.48037	0.35119	0.64978	0.60432	0.56131	-0.0089017	0.20097	-0.19323	0.81056	0.93686	-0.72052	-0.51192
w_{25j}^{c}	-0.47238	-0.46545	0.68503	-0.15035	-0.1296	0.63318	0.21815	1.0289	0.18025	0.15305	0.43367	0.79695	0.69701	0.28507	-0.97295	-0.87153	-0.46975
w_{26j}^{c}	0.51316	-0.97362	0.46658	0.37967	-0.71315	0.42142	0.73399	-0.28616	-0.28547	0.38222	0.19125	0.84154	0.2561	0.15455	-0.19728	0.85725	-0.46564
w_{27j}^{c}	-0.37435	0.63672	0.033183	-0.68245	-0.059941	0.41118	-0.010114	0.63492	-0.74548	-0.12772	-0.30337	-0.74313	-0.28259	0.38671	-0.3978	1.0798	0.57771
w_{28j}^{c}	-0.41175	-0.22372	0.49019	-0.53017	0.18653	0.70832	0.1797	-0.23733	-0.6719	-0.39237	0.43427	0.47943	0.77968	0.45069	-0.041789	0.8574	-0.8389
w_{29j}^{c}	-0.1497	0.13463	0.54455	-0.93445	0.47487	-0.34321	-0.68625	-0.061935	0.62621	0.25116	0.88401	0.73427	-0.22498	-0.28702	-0.47137	-0.45857	-0.11686
w_{30j}^{c}	-0.68238	-0.43292	-1.0998	-0.83499	-0.77585	-0.19696	-0.52874	0.45871	0.75347	-0.77017	0.55601	-0.5936	-0.63575	-0.23151	-0.3557	0.80333	0.45755
w_{31j}^{c}	-0.004107	-0.85468	0.12333	-0.69578	0.98602	0.46893	0.63667	0.5431	-0.21375	-0.78883	-0.50924	-0.92785	0.72015	0.82015	0.076842	0.67152	-0.81918
w_{32j}^{c}	0.68098	0.81001	-0.45571	0.14243	0.77823	-0.82349	-0.80496	0.31021	0.11659	0.27024	0.55312	-0.20715	-0.15071	-0.017526	-0.87671	0.44723	0.61767
w_{33j}^{c}	-0.26274	-0.19066	-0.15799	-0.81156	0.429	-0.83283	1.0111	0.23197	-0.57179	-0.90626	0.0063014	-0.12272	0.88667	0.56397	0.94391	0.81457	-0.57523
w_{34j}^{c}	0.38282	1.0476	0.33268	0.36696	-0.14052	-0.0020331	-0.33331	-0.30811	0.4138	-0.1886	-0.64343	-0.88854	-0.15902	-0.55149	0.88321	-0.34946	-0.79389

Πινάκας 3.3.2.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
w_{1j}^{σ}	0.54864	0.59358	0.32412	0.60621	0.82262	0.86814	0.15295	0.81661	0.78403	0.63196	0.72974	0.32317	0.40033	0.73689	0.56251	0.75437	0.43118
w_{2j}^{σ}	0.67424	0.93513	0.55632	0.53984	0.31178	0.4998	0.28801	0.82918	0.63895	0.36021	0.6838	0.51684	0.55146	0.68541	0.84109	0.90599	0.39982
w_{3j}^{σ}	0.54839	0.88865	0.57996	0.61853	0.63082	0.50173	0.048102	0.98042	0.60414	0.6073	0.22795	0.73642	0.0047553	0.25104	0.76842	0.63089	0.18687
w_{4j}^{σ}	0.69377	0.98051	0.21468	0.65884	0.625	0.82848	0.46724	0.96919	0.89409	0.34337	0.3064	0.3086	0.64968	0.29975	0.54379	0.88176	0.12422
w_{5j}^{σ}	0.64952	0.99423	0.45111	0.69581	0.44383	0.6233	0.52945	1.2898	0.29502	0.37116	0.26209	0.43474	0.21922	0.51219	0.59816	0.26331	0.81489
w_{6j}^{σ}	0.44503	1.0503	0.38125	0.34322	0.14565	0.71114	0.2154	1.4602	0.62227	0.22328	0.28147	0.8458	0.35027	0.85134	0.50728	0.35371	0.77599
w_{7j}^{σ}	0.38879	0.85203	1.0577	0.73853	0.45823	0.59309	0.6791	0.45527	0.65539	0.25988	0.25067	0.51776	0.40011	0.42566	0.23261	1.1338	0.8163
w_{8j}^{σ}	0.60429	0.66957	1.7577	0.78278	0.68627	0.23936	0.28498	0.62002	0.57911	0.2854	0.47112	0.22149	0.74586	0.38189	0.45988	0.54306	0.34336
w_{9j}^{σ}	0.61241	0.85619	0.57031	0.2203	0.11357	0.265	0.8841	0.2556	0.14517	0.23098	0.91264	0.81676	0.51868	0.22805	0.66935	0.4178	0.30131

w_{10j}^{σ}	0.54766	1.2491	0.81173	0.52917	0.86161	0.53467	0.27924	1.0192	0.29658	0.53206	2.0113	0.60555	0.79125	0.53053	0.6527	0.85651	0.52495
w_{11j}^{σ}	0.53898	0.74134	0.60095	0.25748	0.74662	0.47544	0.38563	0.80507	0.77796	0.73447	0.8415	0.70633	0.40293	0.25066	0.22567	0.74777	0.43176
w_{12j}^{σ}	0.58056	1.0915	0.60961	0.61298	0.18851	0.46618	0.75499	0.54786	0.77461	0.25483	0.5507	0.59148	0.31662	0.22664	0.4217	1.2571	0.30918
w_{13j}^{σ}	0.75608	0.55646	0.4731	0.4281	0.82805	0.3714	0.50107	0.5488	0.23427	0.89175	0.93539	0.54824	0.92428	0.30272	0.22294	0.60508	0.57368
w_{14j}^{σ}	0.68396	0.22563	0.53935	0.60669	0.14172	0.25489	0.80659	0.79201	0.84339	0.51004	0.27748	0.2476	0.57619	0.4542	0.63414	0.76377	0.34907
w_{15j}^{σ}	0.18501	0.45113	0.50435	0.59713	0.65537	0.31986	0.84451	1.0116	0.23905	0.35354	1.4143	0.85434	0.24359	0.69098	0.21945	0.70685	0.80721
w_{16j}^{σ}	0.00077908	1.3183	5.3827e-005	0.8226	0.19077	0.26134	0.35277	0.64933	0.42568	0.58579	0.46452	0.58292	0.82023	0.27965	0.36274	0.81112	0.63896
w_{17j}^{σ}	0.42652	0.9593	0.34555	0.87511	0.52567	0.72231	0.68655	0.58192	0.69116	0.45452	1.3848	0.8234	0.4663	0.55849	0.39079	0.67371	0.72661
w_{18j}^{σ}	0.19538	0.88172	0.42227	0.49001	0.32274	0.74354	0.63899	0.62551	0.20261	0.42718	1.0663	0.57758	0.29995	0.38016	0.61844	0.57953	0.72101
w_{19j}^{σ}	0.99283	1.0458	0.62285	0.70843	0.43778	0.90446	0.76534	1.0617	0.21256	0.33511	0.70706	0.52326	0.36162	0.2268	0.6248	0.75921	0.14888
w_{20j}^{σ}	0.77108	0.41255	0.2754	0.7668	0.73253	0.55133	0.2229	0.73669	0.59623	0.6766	0.41099	0.80196	0.68251	0.63303	0.86277	0.67492	0.54051
w_{21j}^{σ}	0.8265	0.80688	0.77965	0.24566	0.54091	0.63985	0.2295	0.53807	0.14951	0.21303	1.0345	0.35609	0.71076	0.2871	0.60818	0.8969	0.79294
w_{22j}^{σ}	0.98249	0.27187	0.91959	0.28148	0.49215	0.22928	0.28045	0.42527	0.71736	0.34508	1.4076	0.56609	0.42163	0.788	0.67143	0.88499	0.37644
w_{23j}^{σ}	0.53029	0.9435	0.76726	0.49077	0.2679	0.54048	0.52385	0.63989	0.42407	0.43851	0.72325	0.79124	0.51442	0.59957	0.40215	0.83674	0.073861
w_{24j}^{σ}	0.67254	0.57366	0.43957	0.46516	0.077495	0.70952	0.62076	0.6981	0.73016	0.74662	0.57146	0.50927	0.63551	0.83401	0.85793	0.74681	0.32195
w_{25j}^{σ}	0.40995	0.92233	0.71392	0.49853	0.47272	0.77266	0.62144	0.61702	0.60525	0.60369	0.63183	0.82902	0.79532	0.64996	0.1782	0.71547	0.32296
w_{26j}^{σ}	0.76341	0.74388	0.78422	0.68508	0.2358	0.67512	0.77225	0.64882	0.4291	0.6839	1.1614	0.81204	0.75857	0.60429	0.47148	0.63373	0.32773
w_{27j}^{σ}	0.75849	0.87531	0.28395	0.31468	0.4741	0.66492	0.54402	1.1726	0.24023	0.50554	0.66071	0.22404	0.30647	0.68545	0.39446	0.50885	0.75592
w_{28j}^{σ}	0.5403	0.65804	0.46972	0.3649	0.57932	0.78599	0.58284	0.7057	0.28348	0.41285	0.69751	0.73216	0.80933	0.70768	0.52114	1.1085	0.14797
w_{29j}^{σ}	0.60945	0.96473	0.61227	0.22958	0.70149	0.42839	0.26822	0.79435	0.75803	0.638	0.58957	0.79856	0.41623	0.44972	0.37194	0.75494	0.48047
w_{30j}^{σ}	0.45457	0.70841	0.31409	0.26151	0.20605	0.46673	0.31135	0.71128	0.80345	0.28059	0.88074	0.30008	0.39847	0.46921	0.41028	1.0876	0.68414
w_{31j}^{σ}	0.67919	0.77237	0.60588	0.31013	0.85848	0.70986	0.76842	0.75252	0.45712	0.27413	0.63798	0.1613	0.76346	0.83725	0.56703	0.98203	0.15818
w_{32j}^{σ}	0.60838	0.47827	0.41821	0.60044	0.79284	0.25035	0.1612	1.0622	0.56189	0.64451	1.4527	0.31217	0.51066	0.54356	0.23149	0.80265	0.79298
w_{33j}^{σ}	0.71155	0.83293	0.33549	0.26859	0.67977	0.23779	0.87619	0.89601	0.33017	0.23308	0.83381	0.45003	0.89883	0.74759	0.86638	0.7918	0.29978
w_{34j}^{σ}	0.58427	0.25491	0.93389	0.67995	0.47945	0.54042	0.40455	0.34101	0.69147	0.4839	0.52694	0.1454	0.67835	0.35728	0.85525	0.68816	0.17303

Πινάκας 3.3.2.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.015503	0.98062
2	-0.0049165	1.0049
3	0.93464	0.064049
4	0.10548	0.18919
5	0.74153	0.59847
6	-0.0058952	1.0099
7	0.17463	0.71444
8	-0.0030825	1.0031
9	0.28364	0.64043
10	0.89129	0.79618
11	0.99276	0.0072359
12	0.82261	0.12396
13	1.0694	-0.064296
14	0.22816	0.38207
15	0.45731	0.54843
16	-0.0015366	1.0015
17	0.40559	0.53959

Πινάκας 3.3.2.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.47705	0.66346
2	0.65036	0.76059
3	0.68591	0.81044
4	0.28297	0.32792
5	0.76668	0.62794
6	0.36761	0.34315
7	0.28026	0.72181
8	0.78383	0.92765
9	0.45379	0.56214
10	0.82418	0.75723
11	0.46154	0.53835
12	0.49431	0.32701
13	0.49003	0.62886
14	0.36029	0.46725
15	0.54457	0.54879
16	0.81172	0.94543
17	0.63168	0.4194

Πινάκας 3.3.2.15: Διασπορές των μετασυναπτικών βαρών

Π 3.3.3 Τιμές μετά την επανεκπαίδευση

για το σύνολο με 12 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
0.33512	1.0022	0.28616	0.79336	0.13904	0.087306	0.1747	0.37379	0.061541	0.54681	0.54141	0.95574	0.60343	0.32174	0.77063	0.69386	0.31756

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.40147	0.86525	0.29722	0.4842	1.2227	0.94226	0.39459	1.4269	0.29895	0.37966	1.8645	0.47057	1.2366	1.3898	0.35636	0.069182	0.50056

Πινάκας 3.3.3.1: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12
w_{1j}^c	1.3745	0.35059	0.15233	0.39047	0.85091	0.95222	0.061085	0.2493	1.3458	0.4538	1.1776	0.15206
w_{2j}^c	0	0	0	0	0	0	0	0	0	0	0	0
w_{3j}^c	1.1472	-0.07588	-0.53503	-0.43032	0.0018698	-0.045211	0.37203	0.093649	0.30674	-0.42261	0.74707	-0.4984
w_{4j}^c	0.3025	0.59861	0.31058	-0.65549	-0.76358	0.96133	0.20653	0.017991	0.013492	0.82258	0.31115	-0.85683
w_{5j}^c	0.052322	-0.93443	-0.60211	0.16888	-0.94526	0.51461	0.29338	-0.31973	0.74051	-0.89867	1.3929	1.0536
w_{6j}^c	-0.37765	-0.95163	-0.55565	1.0507	-0.72515	0.026322	-0.89689	-0.53493	0.18693	-0.082835	0.35517	0.40113
w_{7j}^c	0.79152	0.16086	0.0014454	0.25612	-0.69158	-0.52543	0.76118	-0.69128	0.48742	-0.43789	1.0869	-0.61696
w_{8j}^c	0.24383	0.25255	0.029272	-0.88421	-0.61312	-0.89446	0.47866	-0.96859	0.5128	-0.84262	0.1454	-0.43161
w_{9j}^c	1.3237	0.4261	0.86707	0.39182	0.35023	0.64416	-0.92329	0.11352	0.083493	0.31782	0.47186	0.17406
w_{10j}^c	0.40933	0.090169	0.31009	-1.0394	0.22662	-0.16225	-0.39152	-0.075852	0.37308	0.19396	0.65608	0.66718
w_{11j}^c	0.62344	0.033092	0.67194	0.6194	-0.78232	-0.48049	-0.71631	0.28298	-0.2033	-0.2052	0.31395	0.80944
w_{12j}^c	-0.74642	-0.76676	-0.76244	-0.55123	0.29331	0.97631	0.53587	-0.81185	0.007683	-0.78037	0.60009	-0.58649
w_{13j}^c	-0.29883	-0.50272	0.68019	1.1189	-0.30815	-0.88835	0.33734	0.080413	-0.95532	0.043021	-0.55322	-0.66406
w_{14j}^c	-0.29466	0.88798	0.030709	-0.83935	0.58855	-0.6718	-0.048068	0.71106	0.14345	0.20945	0.1441	0.31201
w_{15j}^c	0.94767	0.23971	-0.37306	0.092902	0.43665	-0.8949	0.69131	-0.1706	-0.41256	0.07961	0.73054	0.61247
w_{16j}^c	0.26254	0.6653	-0.74719	-0.70668	-0.84823	-0.57749	-0.19499	0.29643	-0.47539	0.091722	0.68933	-0.836

w_{17j}^{c}	0.63979	1.054	-0.34887	0.97478	-0.86506	-0.17641	0.20267	-0.55793	-0.71808	-0.14764	0.4775	-0.51517
w_{18j}^{c}	0.71573	0.087109	-0.36999	-0.4625	0.7771	-0.47106	-0.042614	-0.05128	-0.35323	0.81286	0.67737	0.15171
w_{19j}^{c}	0.34388	-0.30542	-0.87812	-0.025748	0.12391	0.46957	0.26378	0.38363	-0.53121	-0.62918	-0.13688	0.81965
w_{20j}^{c}	0.0086114	0.6122	-0.088003	0.65387	-0.76653	0.38679	-0.27433	0.57244	-0.0059119	-0.56121	0.21954	-0.5267
w_{21j}^{c}	1.0068	0.25807	0.53239	-0.14606	0.79862	0.159	-0.87256	-0.095518	0.46334	0.46816	-0.85819	0.49711
w_{22j}^{c}	-1.0314	0.92002	0.81846	0.17522	0.12341	-0.2722	0.78903	-0.46437	-1.047	-0.29131	-0.5505	-0.56936
w_{23j}^{c}	0.48399	1.2107	0.33459	0.36939	-0.79351	-0.33259	0.57708	-0.31451	-0.19698	-0.57756	0.27923	0.92956
w_{24j}^{c}	0.20023	0.43894	0.38058	0.74138	0.46153	-0.11937	-0.46185	0.6988	-0.090144	0.36159	-0.15008	0.020313
w_{25j}^{c}	0.37094	0.18563	0.13164	-0.45717	-0.18153	-0.59608	0.68885	0.053953	-0.64595	-0.73303	-0.12251	0.71676
w_{26j}^{c}	0.50597	-0.26094	-0.47635	0.31647	0.33103	0.56965	0.97393	0.81276	-0.0016063	0.27965	0.059198	1.0284
w_{27j}^{c}	0.17517	-0.12917	-0.85114	-0.27757	-0.17525	-0.11653	-0.82457	-0.40747	-0.02319	0.0029811	-0.67321	0.63397
w_{28j}^{c}	0.10961	-0.77312	0.64505	0.93176	0.047949	-0.78981	-0.73803	0.071486	-1.0609	0.34575	0.70586	0.54092
w_{29j}^{c}	1.0629	0.65107	0.095478	0.098822	0.7915	-0.27783	0.95372	-0.80938	0.44316	0.068425	0.49615	-0.69679
w_{30j}^{c}	-0.42482	-0.6559	-0.56961	-1.1877	0.38217	-0.69492	0.22357	0.8146	0.73489	0.70358	-0.17341	0.5016
w_{31j}^{c}	-0.76107	0.22775	-0.87469	-0.31881	0.97879	0.55794	-0.60888	-0.30105	0.044182	-0.46159	0.37485	0.45863
w_{32j}^{c}	-0.041675	-0.2142	-0.52728	0.47522	0.83012	-0.13823	-0.63577	-0.38681	0.049493	0.010894	-0.20825	-0.56594
w_{33j}^{c}	0.619	1.0825	-0.10034	1.5587	-0.37656	-0.78205	-0.95728	0.4471	0.37362	-0.030901	-0.40777	-1.1015
w_{34j}^{c}	-0.48422	1.0476	0.0014785	1.0679	0.56525	0.5762	0.20587	0.36827	0.28303	0.79452	-0.68508	-0.57573

Πινάκας 3.3.3.2: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12
w_{1j}^{σ}	0.42972	0.90886	0.28018	0.88614	0.8024	0.89308	0.20953	0.85994	0.81543	0.9287	0.57247	0.27695
w_{2j}^{σ}	0.51477	0.83301	0.55573	0.97192	0.44671	0.57609	0.83296	0.85455	0.54481	0.9639	0.60054	0.23988
w_{3j}^{σ}	0.49358	0.64979	0.3526	0.7528	0.54137	0.56966	0.68292	1.1048	0.28551	0.48514	0.79943	0.35629
w_{4j}^{σ}	0.56708	0.9821	0.65055	0.52894	0.2675	0.88766	0.61273	0.77558	0.54556	1.0665	0.48927	0.049509
w_{5j}^{σ}	3.2519	1.9983	0.30125	1.1993	0.21459	0.77432	0.65411	0.46972	0.56591	0.58082	18.1614	0.80338
w_{6j}^{σ}	0.4142	1.5116	0.3327	0.55147	0.28777	0.54723	0.21529	0.83355	17.1788	1.279	0.08924	0.67011
w_{7j}^{σ}	0.54546	0.98179	0.54013	1.0518	0.23014	0.31735	0.81044	0.49864	0.15172	0.51471	0.49272	0.29631
w_{8j}^{σ}	0.52578	1.297	0.55113	1.0653	0.31991	0.19246	0.72662	0.64229	0.34838	0.57098	0.71846	0.30569

w_{9j}^{σ}	0.24468	1.418	0.84224	0.47428	0.67601	0.78164	0.13505	0.73561	1.0656	1.0733	1.1385	0.58375
w_{10j}^{σ}	0.61358	1.2764	0.64295	0.74124	0.62319	0.51948	0.41007	0.71689	0.52946	1.2885	0.59196	0.78168
w_{11j}^{σ}	0.27283	1.2923	0.78147	0.96357	0.25893	0.3478	0.26417	0.51221	1.0468	0.69567	0.90615	0.83657
w_{12j}^{σ}	0.33103	0.97642	0.26824	0.5226	0.64609	0.92803	0.73127	0.80589	0.49898	0.80508	0.76013	0.27755
w_{13j}^{σ}	0.99764	1.0817	0.77856	0.87537	0.43948	0.22189	0.67995	0.82923	0.65907	0.70457	0.72459	0.38019
w_{14j}^{σ}	0.7236	0.94327	0.54531	0.73945	0.7481	0.32123	0.51745	0.52166	0.37302	1.1026	0.32131	0.60594
w_{15j}^{σ}	4.751	0.93493	0.39935	1.1415	0.7037	0.25804	0.80196	0.56664	0.61694	0.78728	0.8144	0.6145
w_{16j}^{σ}	0.52763	1.2049	0.27937	1.3392	0.23069	0.31929	0.47523	0.70375	0.36642	0.89264	0.57739	0.13403
w_{17j}^{σ}	0.63103	0.32489	0.42338	0.77478	0.23162	0.48793	0.63453	0.57293	1.1472	0.8147	1.1283	0.56683
w_{18j}^{σ}	0.65065	1.2755	0.40958	0.87771	0.81562	0.41412	0.5224	1.1529	0.24467	0.96129	0.55633	0.65529
w_{19j}^{σ}	0.39941	1.0176	0.18865	1.1387	0.58903	0.5739	0.65632	1.1806	0.74105	1.2378	0.74576	0.85464
w_{20j}^{σ}	0.30128	1.3699	0.49764	0.8791	0.27528	0.63187	0.43588	0.52773	0.25096	1.2525	0.56789	0.35202
w_{21j}^{σ}	0.40423	0.88102	0.73812	1.1634	0.81146	0.48467	0.1684	1.8653	0.47686	1.2787	3.1582	0.41577
w_{22j}^{σ}	0.81308	0.64423	0.82144	0.77296	0.58937	0.40362	0.81426	0.66593	0.46751	0.9187	0.75294	0.17306
w_{23j}^{σ}	0.39664	0.6402	0.65869	0.58506	0.2609	0.4085	0.74121	0.70946	0.6225	0.63638	0.47941	0.84763
w_{24j}^{σ}	0.26063	1.4751	0.70318	1.1667	0.6996	0.44494	0.38481	0.78147	0.40173	0.70634	0.78724	0.55664
w_{25j}^{σ}	1.0685	0.70744	0.58407	0.64737	0.47758	0.3607	0.78409	0.82349	0.50731	0.89739	0.53403	0.79536
w_{26j}^{σ}	0.84013	0.29547	0.36832	0.6025	0.66092	0.72675	0.87816	0.89027	0.31065	1.1173	0.76545	0.91412
w_{27j}^{σ}	0.46088	1.5516	0.15119	0.82214	0.47743	0.53555	0.2171	1.3309	0.35049	1.252	0.8834	0.71578
w_{28j}^{σ}	0.50356	0.83111	0.73361	0.92033	0.55438	0.29442	0.27997	0.94603	2.5569	1.2615	0.93109	0.7486
w_{29j}^{σ}	0.33272	1.207	0.58126	0.34199	0.81591	0.46755	0.87636	0.83168	0.50708	1.0166	0.89868	0.1739
w_{30j}^{σ}	0.38912	0.89937	0.33967	0.51301	0.67663	0.30334	0.62079	1.1018	0.92013	1.1992	0.48311	0.66481
w_{31j}^{σ}	0.47181	0.67687	0.22509	0.5778	0.88833	0.75477	0.32034	0.80394	0.61811	0.88726	0.98628	0.66699
w_{32j}^{σ}	0.95622	0.74489	0.32055	0.73715	0.84871	0.4946	0.32292	0.92339	0.34699	1.0893	0.72752	0.16134
w_{33j}^{σ}	1.1514	0.76378	0.50658	0.001	0.41196	0.2038	0.18355	0.94511	0.8784	0.95673	0.67952	0.0011053
w_{34j}^{σ}	0.63988	0.57049	0.56277	0.66221	0.75217	0.7916	0.63484	0.53678	1.4302	0.76962	0.44714	0.23667

Πινάκας 3.3.3.3: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.96756	0.033
2	-0.012424	1.0124
3	0.77121	0.99979
4	0.041937	0.96334
5	0.65014	0.70717
6	0.092798	0.60794
7	0.16877	0.74663
8	-0.016587	1.015
9	0.9937	0.0054719
10	-0.0048589	1.0055
11	1.0021	-0.0022362
12	0.81033	0.12622

Πινάκας 3.3.3.4: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.41276	0.4847
2	0.59148	0.68325
3	0.773	0.88886
4	0.38468	0.43562
5	0.78296	0.63462
6	0.41734	0.30647
7	0.30038	0.73061
8	0.73057	0.95346
9	0.40971	0.46221
10	0.97265	0.988
11	0.47902	0.51687
12	0.46546	0.31132

Πινάκας 3.3.3.5: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την επανεκπαίδευση ,
για το σύνολο με 7 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
239.8863	0.48646	0.21188	1.416	0.0006401	0.47981	0.19045	0.22632	0.69829	0.72676	0.89367	2.0813	0.13647	0.16834	0.89989	0.40583	0.070637

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.72446	0.44127	0.94302	1.4581	0.12919	0.89055	0.20628	0.46604	0.82118	0.47519	0.0061385	0.88843	1.4886	0.94564	1.2633	0.95628	0.84499

Πινάκας 3.3.3.6: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^c	0.27667	0.7569	-0.018165	1.5306	1.4776	0.91854	81.5753
w_{2j}^c	0	0	0	0	0	0	0
w_{3j}^c	-0.55484	1.4908	-0.29226	0.61953	0.63199	0.1041	-0.024884
w_{4j}^c	-0.99953	0.30118	-0.55827	-0.22813	0.093087	-0.88879	-0.2817
w_{5j}^c	-0.80001	0.23281	-0.27754	0.48392	0.25782	-0.33109	-0.45766
w_{6j}^c	-0.23216	0.6895	-0.033687	-0.31516	-0.47705	-1.2191	0.97425
w_{7j}^c	-0.39417	0.71291	0.20356	0.9624	0.21583	-0.70875	0.65041
w_{8j}^c	-1.3257	-0.33712	-0.76711	0.46773	0.38621	-0.17194	0.19211
w_{9j}^c	0.00017458	-0.016191	0.60883	1.1107	-0.24914	-0.13319	-0.59691
w_{10j}^c	1.026	-0.76312	0.066163	0.45689	-0.85556	-1.0503	-0.07941
w_{11j}^c	0.0019727	0.61122	-0.058942	0.56781	0.35349	0.19455	-0.59887
w_{12j}^c	-0.8415	0.73589	-0.87383	-0.036876	-0.42952	0.063741	-0.63088
w_{13j}^c	0.74076	0.68486	-0.070407	-0.88378	-0.77922	0.24515	0.68477
w_{14j}^c	-0.91573	0.36606	0.18098	-0.035231	-0.64311	0.46811	-0.31542
w_{15j}^c	1.0418	0.090267	0.053554	0.60958	-0.62398	0.089492	-0.5757
w_{16j}^c	0.57384	0.21316	0.2173	0.24603	0.35781	-0.074963	-1.0086
w_{17j}^c	0.6893	-0.23087	-0.40772	-0.4983	-0.51121	0.22757	1.0212

w_{18j}^{c}	0.77413	0.76322	0.036397	-0.0089302	0.17069	-1.0697	-0.8527
w_{19j}^{c}	-0.67961	0.14322	-0.59809	0.0010396	0.63281	1.0181	-0.74032
w_{20j}^{c}	-0.089918	-0.74642	-0.30887	-0.053651	0.87346	0.24903	-0.89695
w_{21j}^{c}	-0.61458	-0.68453	0.83181	-0.77472	-0.6793	-0.81935	0.72619
w_{22j}^{c}	0.75884	-0.44806	-0.85144	-0.72351	-0.12472	0.14093	0.13643
w_{23j}^{c}	0.011742	-0.48986	-0.10062	0.29968	0.26503	-0.54295	0.72628
w_{24j}^{c}	-0.62317	-0.25483	0.64085	-0.13332	0.57822	0.54227	0.39354
w_{25j}^{c}	-0.06158	0.61919	-0.22523	0.39583	0.6868	-0.45389	0.83214
w_{26j}^{c}	0.3571	-0.38936	0.23403	0.33634	0.8185	-0.63169	0.56603
w_{27j}^{c}	0.41593	-0.90262	-0.65643	-0.85143	0.080771	0.78292	0.23356
w_{28j}^{c}	0.77809	0.0018602	-1.1084	0.15338	0.66949	-0.85928	0.9646
w_{29j}^{c}	-0.79003	-0.060249	0.156	0.69313	-0.81739	-0.26615	0.81208
w_{30j}^{c}	-0.37334	0.59205	0.63741	-0.27623	-0.44448	0.49704	-0.59655
w_{31j}^{c}	0.52943	-0.14072	-0.82954	0.51265	0.13283	-0.017969	0.82825
w_{32j}^{c}	-0.10417	-0.8402	-0.19218	-0.66591	0.19108	-0.02511	-0.099012
w_{33j}^{c}	0.9225	-0.7008	-0.77433	0.33174	1.1165	0.88307	-0.19546
w_{34j}^{c}	0.87685	-0.81627	0.42482	-0.94417	0.2552	0.28794	-0.5452

Πινάκας 3.3.3.7: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7
w_{1j}^{σ}	1.2154	0.52967	2.7081	0.81413	3.5402e-005	1.0847	269.4827
w_{2j}^{σ}	0.60183	0.86232	0.87646	1.0851	0.29238	0.66489	0.85496
w_{3j}^{σ}	0.54439	0.00037334	0.28354	0.167	0.74061	0.97511	0.68435
w_{4j}^{σ}	1.3652	0.40733	0.70889	25.4521	0.78911	1.3889	0.4846
w_{5j}^{σ}	0.49264	0.64516	0.51505	0.20978	0.76371	1.4605	0.49032
w_{6j}^{σ}	1.8196	0.63863	0.63893	1.1717	0.30365	0.615	1.0627
w_{7j}^{σ}	0.36576	0.28342	0.99261	0.39634	0.16951	0.48669	0.85967
w_{8j}^{σ}	0.88655	0.37532	0.24486	0.24762	0.81006	1.1395	0.59069
w_{9j}^{σ}	0.69851	0.41735	1.0268	0.82318	0.54483	0.27872	0.21045

w_{10j}^{σ}	0.73182	0.36035	0.67123	0.74964	0.2625	0.79512	0.45751
w_{11j}^{σ}	0.89218	0.783	0.52827	0.9028	0.45698	0.89367	0.38739
w_{12j}^{σ}	1.0558	0.85788	0.38107	2.0813	0.48336	1.044	0.18064
w_{13j}^{σ}	1.0135	0.72851	0.42219	0.95795	10.3661	0.13648	0.59428
w_{14j}^{σ}	0.89808	0.62488	0.65764	0.19558	0.35062	0.16167	0.33862
w_{15j}^{σ}	0.94946	0.37512	0.59328	0.9225	0.89209	0.89989	0.17289
w_{16j}^{σ}	1.1768	0.55573	0.54539	1.1173	1.0303	0.8807	0.24924
w_{17j}^{σ}	0.77947	0.4654	0.43904	0.71718	0.69836	1.531	0.6729
w_{18j}^{σ}	0.79427	0.63283	0.49864	0.72446	0.66008	0.90009	0.11897
w_{19j}^{σ}	0.83159	0.78023	0.75952	0.83099	0.53441	0.45318	0.49902
w_{20j}^{σ}	0.4951	0.33995	0.41905	0.90748	1.1824	0.94302	0.010508
w_{21j}^{σ}	0.97195	0.30989	0.94583	0.89275	0.37461	0.62005	0.73587
w_{22j}^{σ}	0.85806	0.51985	0.62175	1.2386	0.45504	0.13932	0.46806
w_{23j}^{σ}	0.89302	0.34027	0.63566	0.89937	0.45853	0.9341	0.68885
w_{24j}^{σ}	1.1625	0.51907	0.55339	0.24611	1.0429	0.40775	0.67069
w_{25j}^{σ}	0.56635	0.54358	0.45068	0.46604	0.54186	1.4096	0.78116
w_{26j}^{σ}	0.77297	0.48163	0.72275	0.41746	0.63203	0.73605	0.74037
w_{27j}^{σ}	1.4016	0.33324	0.76706	0.39961	0.41624	1.0044	0.54729
w_{28j}^{σ}	0.73267	0.44884	0.38884	0.42293	0.63934	0.97509	0.59804
w_{29j}^{σ}	0.69527	0.4455	0.65634	0.78876	0.16022	0.83317	0.80847
w_{30j}^{σ}	1.1209	0.66163	0.73807	0.75677	0.24853	1.1236	0.23856
w_{31j}^{σ}	1.2374	0.40672	0.4815	0.87897	0.44421	1.0697	0.79499
w_{32j}^{σ}	0.99321	0.39926	0.53317	0.85558	0.68485	0.7858	0.40818
w_{33j}^{σ}	0.85656	0.46474	0.29145	0.95827	0.54157	0.84874	0.46321
w_{34j}^{σ}	0.77467	0.37935	0.56885	0.58226	0.69109	0.84499	0.00033412

Πινάκας 3.3.3.8: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	-0.0015955	1.0016
2	0.87579	0.09548
3	-0.0033415	1.0029
4	0.9861	0.013895
5	0.4866	0.49623
6	-0.012233	1.0122
7	0.14324	0.88272

Πινάκας 3.3.3.9: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.70947	0.75065
2	0.31646	0.58449
3	0.79374	0.89223
4	0.57946	0.61229
5	0.74524	0.61286
6	0.6848	0.72401
7	0.25154	0.68402

Πινάκας 3.3.3.10: Διασπορές των μετασυναπτικών βαρών

Τιμές μετά την επανεκπαίδευση ,
για το σύνολο με 17 κανόνες

x_1^σ	x_2^σ	x_3^σ	x_4^σ	x_5^σ	x_6^σ	x_7^σ	x_8^σ	x_9^σ	x_{10}^σ	x_{11}^σ	x_{12}^σ	x_{13}^σ	x_{14}^σ	x_{15}^σ	x_{16}^σ	x_{17}^σ
0.58476	0.97318	0.00029978	0.00033362	0.045303	0.25946	0.23488	0.45432	0.40518	1.4733	0.74134	0.42718	0.55645	0.21955	0.2251	0.05107	0.57307

x_{18}^σ	x_{19}^σ	x_{20}^σ	x_{21}^σ	x_{22}^σ	x_{23}^σ	x_{24}^σ	x_{25}^σ	x_{26}^σ	x_{27}^σ	x_{28}^σ	x_{29}^σ	x_{30}^σ	x_{31}^σ	x_{32}^σ	x_{33}^σ	x_{34}^σ
0.17227	0.83964	0.40109	0.80688	0.032481	1.1402	0.57366	0.6208	0.65495	0.38956	1.2151	0.5881	1.2858	1.5209	0.19663	0.83293	0.22555

Πινάκας 3.3.3.11: Διασπορές των ασαφοποιητών εισόδου

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
w_{1j}^c	0.36219	-0.019863	0.24263	0.58192	0.84818	0.97669	0.023594	0.22797	0.85078	0.61729	1.1296	0.25743	0.18292	0.76727	0.51223	0.88651	0.43243
w_{2j}^c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w_{3j}^c	-0.29261	-0.36622	1.046	0.18938	0.36185	-0.15191	-0.87235	0.94013	0.20136	0.16268	0.66907	0.68071	-0.17498	-0.85554	0.6358	-0.6332	-0.4841
w_{4j}^c	-0.34982	-0.92998	0.45605	0.31063	0.26643	0.83074	-0.11133	0.97892	0.97607	-0.59131	0.21718	-0.55364	0.21985	-0.71573	0.012739	-0.72137	-0.6039
w_{5j}^c	-0.6053	-0.21109	0.30585	0.40956	-0.10527	0.33884	-0.030621	0.19264	-0.62859	-0.51098	0.49789	-0.91012	0.30661	-0.10797	0.18819	-0.39675	0.76519
w_{6j}^c	0.20221	-0.4368	-0.75709	-0.59352	-0.94706	0.4628	-0.77557	0.2745	0.23881	-0.9344	0.32179	1.0107	-0.16456	0.86023	-0.10653	0.067505	0.65138
w_{7j}^c	-0.61443	-0.19279	-0.22714	0.53683	-0.20377	0.14427	0.36617	-0.36997	0.29785	-0.83027	0.8956	0.067111	-0.029394	-0.35596	-0.82469	0.15237	0.77879
w_{8j}^c	0.35353	0.78062	-0.34922	0.66871	0.39661	-0.81512	-0.5584	-1.1724	0.10503	-0.75839	0.50495	0.19685	0.81944	-0.48089	-0.14563	-0.90471	-0.42946
w_{9j}^c	-0.038174	0.83377	0.92631	-0.95755	-1.0232	-0.74234	0.89142	-0.11677	-0.99716	-0.91252	1.0616	0.86733	-0.3108	-0.92168	0.33957	-0.088803	-0.52541
w_{10j}^c	-0.035576	-0.85496	-0.79422	-0.058348	0.90496	0.022214	-0.71515	0.21327	-0.6446	-0.051826	1.2956	-0.010434	0.70646	-0.055806	0.2902	0.11578	0.021102
w_{11j}^c	-0.1659	0.19455	0.50096	-0.84794	0.71657	-0.19416	-0.36723	0.25782	0.71035	0.52648	-0.44542	0.55526	-0.84456	-0.85643	-0.86132	-0.0070249	-0.22528
w_{12j}^c	-0.2751	-0.40216	-0.072265	0.17498	-0.79383	-0.21588	0.66882	-0.2002	0.70238	-0.844	0.07265	0.25197	-0.25934	-0.92536	-0.33098	0.68245	-0.51837
w_{13j}^c	0.1832	0.24515	-0.46129	-0.35901	0.76628	-0.461	-0.1119	-0.026327	-0.78648	0.97565	-0.87454	0.31766	0.65787	-0.70792	-0.86832	-0.019922	0.18684
w_{14j}^c	0.46961	0.37635	-0.30796	0.16072	-0.88282	-0.76581	0.83793	0.75566	0.89247	-0.11478	-0.29986	-0.69699	0.10441	-0.27432	0.268	-0.094465	-0.40548
w_{15j}^c	0.23278	-0.071027	1.3429	0.11406	0.57384	-0.56355	0.78602	0.11054	-0.7795	-0.56225	0.14242	0.9967	-0.30729	0.40259	-0.89082	0.76562	0.85525
w_{16j}^c	0.54	0.15834	0.25299	0.77168	-0.75672	-0.74321	-0.61249	-0.81983	-0.36904	0.10309	-0.04282	0.39606	0.2204	-0.77719	-0.40458	0.36494	0.18799

w_{17j}^{c}	-0.60168	-0.40054	0.25243	0.92419	-0.066526	0.56532	0.35767	1.0201	0.46881	-0.27272	-0.50007	0.86555	-0.28996	0.024353	-0.43537	0.093986	0.49936
w_{18j}^{c}	0.31599	-0.027218	0.0083564	-0.18255	-0.50019	0.70299	0.3891	-0.80716	-0.87578	-0.35087	0.40316	0.31632	-0.27345	-0.48564	0.18951	0.7797	0.55855
w_{19j}^{c}	0.8678	0.91406	-0.47164	0.45139	-0.22045	0.98724	0.54326	-0.38913	-0.837	-0.61538	-0.42578	-0.24644	-0.78363	-0.92955	0.26249	-0.48547	-0.85664
w_{20j}^{c}	0.47242	0.26587	-0.75959	0.61973	0.54996	0.049469	-0.77622	-0.6386	0.17326	0.36143	-0.10634	0.83455	0.44525	0.23686	0.92412	0.68044	0.10145
w_{21j}^{c}	0.72428	0.24903	0.74422	-0.87273	0.056094	0.26039	-0.68037	-0.42724	-0.99157	-0.9644	-0.66383	-0.87219	0.47469	-0.75233	0.22576	1.0208	0.66304
w_{22j}^{c}	0.6777	-0.33448	-0.23854	-0.78895	-0.041185	-0.84918	-0.4842	0.95865	0.60424	-0.58673	-0.84303	0.17738	-0.50498	0.6785	0.42528	0.26177	-0.038077
w_{23j}^{c}	-0.51909	-0.089442	0.36236	-0.17489	-0.62116	0.0093595	-0.063116	0.55094	-0.30399	-0.31896	0.9444	0.67982	0.0015949	0.14092	-0.38185	-0.64907	-0.9929
w_{24j}^{c}	0.58323	0.29961	-0.51895	-0.25071	-1.0132	0.48037	0.35119	0.64978	0.60432	0.56131	-0.0089017	0.20097	-0.19323	0.81056	0.93686	-0.72052	-0.51192
w_{25j}^{c}	-0.47238	-0.46545	0.68503	-0.15035	-0.1296	0.63318	0.21815	1.0289	0.18025	0.15305	0.43367	0.79695	0.69701	0.28507	-0.97295	-0.87153	-0.46975
w_{26j}^{c}	0.51316	-0.97362	0.46658	0.37967	-0.71315	0.42142	0.73399	-0.28616	-0.28547	0.38222	0.19125	0.84154	0.2561	0.15455	-0.19728	0.85725	-0.46564
w_{27j}^{c}	-0.37435	0.63672	0.033183	-0.68245	-0.059941	0.41118	-0.010114	0.63492	-0.74548	-0.12772	-0.30337	-0.74313	-0.28259	0.38671	-0.3978	1.0798	0.57771
w_{28j}^{c}	-0.41175	-0.22372	0.49019	-0.53017	0.18653	0.70832	0.1797	-0.23733	-0.6719	-0.39237	0.43427	0.47943	0.77968	0.45069	-0.041789	0.8574	-0.8389
w_{29j}^{c}	-0.1497	0.13463	0.54455	-0.93445	0.47487	-0.34321	-0.68625	-0.061935	0.62621	0.25116	0.88401	0.73427	-0.22498	-0.28702	-0.47137	-0.45857	-0.11686
w_{30j}^{c}	-0.68238	-0.43292	-1.0998	-0.83499	-0.77585	-0.19696	-0.52874	0.45871	0.75347	-0.77017	0.55601	-0.5936	-0.63575	-0.23151	-0.3557	0.80333	0.45755
w_{31j}^{c}	-0.004107	-0.85468	0.12333	-0.69578	0.98602	0.46893	0.63667	0.5431	-0.21375	-0.78883	-0.50924	-0.92785	0.72015	0.82015	0.076842	0.67152	-0.81918
w_{32j}^{c}	0.68098	0.81001	-0.45571	0.14243	0.77823	-0.82349	-0.80496	0.31021	0.11659	0.27024	0.55312	-0.20715	-0.15071	-0.017526	-0.87671	0.44723	0.61767
w_{33j}^{c}	-0.26274	-0.19066	-0.15799	-0.81156	0.429	-0.83283	1.0111	0.23197	-0.57179	-0.90626	0.0063014	-0.12272	0.88667	0.56397	0.94391	0.81457	-0.57523
w_{34j}^{c}	0.38282	1.0476	0.33268	0.36696	-0.14052	-0.0020331	-0.33331	-0.30811	0.4138	-0.1886	-0.64343	-0.88854	-0.15902	-0.55149	0.88321	-0.34946	-0.79389

Πινάκας 3.3.3.12: Κέντρα των προσυναπτικών βαρών

Rule #(j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
w_{1j}^{σ}	0.54864	0.59358	0.32412	0.60621	0.82262	0.86814	0.15295	0.81661	0.78403	0.63196	0.72974	0.32317	0.40033	0.73689	0.56251	0.75437	0.43118
w_{2j}^{σ}	0.67424	0.93513	0.55632	0.53984	0.31178	0.4998	0.28801	0.82918	0.63895	0.36021	0.6838	0.51684	0.55146	0.68541	0.84109	0.90599	0.39982
w_{3j}^{σ}	0.54839	0.88865	0.57996	0.61853	0.63082	0.50173	0.048102	0.98042	0.60414	0.6073	0.22795	0.73642	0.0047553	0.25104	0.76842	0.63089	0.18687
w_{4j}^{σ}	0.69377	0.98051	0.21468	0.65884	0.625	0.82848	0.46724	0.96919	0.89409	0.34337	0.3064	0.3086	0.64968	0.29975	0.54379	0.88176	0.12422
w_{5j}^{σ}	0.64952	0.99423	0.45111	0.69581	0.44383	0.6233	0.52945	1.2898	0.29502	0.37116	0.26209	0.43474	0.21922	0.51219	0.59816	0.26331	0.81489
w_{6j}^{σ}	0.44503	1.0503	0.38125	0.34322	0.14565	0.71114	0.2154	1.4602	0.62227	0.22328	0.28147	0.8458	0.35027	0.85134	0.50728	0.35371	0.77599
w_{7j}^{σ}	0.38879	0.85203	1.0577	0.73853	0.45823	0.59309	0.6791	0.45527	0.65539	0.25988	0.25067	0.51776	0.40011	0.42566	0.23261	1.1338	0.8163
w_{8j}^{σ}	0.60429	0.66957	1.7577	0.78278	0.68627	0.23936	0.28498	0.62002	0.57911	0.2854	0.47112	0.22149	0.74586	0.38189	0.45988	0.54306	0.34336
w_{9j}^{σ}	0.61241	0.85619	0.57031	0.2203	0.11357	0.265	0.8841	0.2556	0.14517	0.23098	0.91264	0.81676	0.51868	0.22805	0.66935	0.4178	0.30131

w_{10j}^{σ}	0.54766	1.2491	0.81173	0.52917	0.86161	0.53467	0.27924	1.0192	0.29658	0.53206	2.0113	0.60555	0.79125	0.53053	0.6527	0.85651	0.52495
w_{11j}^{σ}	0.53898	0.74134	0.60095	0.25748	0.74662	0.47544	0.38563	0.80507	0.77796	0.73447	0.8415	0.70633	0.40293	0.25066	0.22567	0.74777	0.43176
w_{12j}^{σ}	0.58056	1.0915	0.60961	0.61298	0.18851	0.46618	0.75499	0.54786	0.77461	0.25483	0.5507	0.59148	0.31662	0.22664	0.4217	1.2571	0.30918
w_{13j}^{σ}	0.75608	0.55646	0.4731	0.4281	0.82805	0.3714	0.50107	0.5488	0.23427	0.89175	0.93539	0.54824	0.92428	0.30272	0.22294	0.60508	0.57368
w_{14j}^{σ}	0.68396	0.22563	0.53935	0.60669	0.14172	0.25489	0.80659	0.79201	0.84339	0.51004	0.27748	0.2476	0.57619	0.4542	0.63414	0.76377	0.34907
w_{15j}^{σ}	0.18501	0.45113	0.50435	0.59713	0.65537	0.31986	0.84451	1.0116	0.23905	0.35354	1.4143	0.85434	0.24359	0.69098	0.21945	0.70685	0.80721
w_{16j}^{σ}	0.00077908	1.3183	5.3827e-005	0.8226	0.19077	0.26134	0.35277	0.64933	0.42568	0.58579	0.46452	0.58292	0.82023	0.27965	0.36274	0.81112	0.63896
w_{17j}^{σ}	0.42652	0.9593	0.34555	0.87511	0.52567	0.72231	0.68655	0.58192	0.69116	0.45452	1.3848	0.8234	0.4663	0.55849	0.39079	0.67371	0.72661
w_{18j}^{σ}	0.19538	0.88172	0.42227	0.49001	0.32274	0.74354	0.63899	0.62551	0.20261	0.42718	1.0663	0.57758	0.29995	0.38016	0.61844	0.57953	0.72101
w_{19j}^{σ}	0.99283	1.0458	0.62285	0.70843	0.43778	0.90446	0.76534	1.0617	0.21256	0.33511	0.70706	0.52326	0.36162	0.2268	0.6248	0.75921	0.14888
w_{20j}^{σ}	0.77108	0.41255	0.2754	0.7668	0.73253	0.55133	0.2229	0.73669	0.59623	0.6766	0.41099	0.80196	0.68251	0.63303	0.86277	0.67492	0.54051
w_{21j}^{σ}	0.8265	0.80688	0.77965	0.24566	0.54091	0.63985	0.2295	0.53807	0.14951	0.21303	1.0345	0.35609	0.71076	0.2871	0.60818	0.8969	0.79294
w_{22j}^{σ}	0.98249	0.27187	0.91959	0.28148	0.49215	0.22928	0.28045	0.42527	0.71736	0.34508	1.4076	0.56609	0.42163	0.788	0.67143	0.88499	0.37644
w_{23j}^{σ}	0.53029	0.9435	0.76726	0.49077	0.2679	0.54048	0.52385	0.63989	0.42407	0.43851	0.72325	0.79124	0.51442	0.59957	0.40215	0.83674	0.073861
w_{24j}^{σ}	0.67254	0.57366	0.43957	0.46516	0.077495	0.70952	0.62076	0.6981	0.73016	0.74662	0.57146	0.50927	0.63551	0.83401	0.85793	0.74681	0.32195
w_{25j}^{σ}	0.40995	0.92233	0.71392	0.49853	0.47272	0.77266	0.62144	0.61702	0.60525	0.60369	0.63183	0.82902	0.79532	0.64996	0.1782	0.71547	0.32296
w_{26j}^{σ}	0.76341	0.74388	0.78422	0.68508	0.2358	0.67512	0.77225	0.64882	0.4291	0.6839	1.1614	0.81204	0.75857	0.60429	0.47148	0.63373	0.32773
w_{27j}^{σ}	0.75849	0.87531	0.28395	0.31468	0.4741	0.66492	0.54402	1.1726	0.24023	0.50554	0.66071	0.22404	0.30647	0.68545	0.39446	0.50885	0.75592
w_{28j}^{σ}	0.5403	0.65804	0.46972	0.3649	0.57932	0.78599	0.58284	0.7057	0.28348	0.41285	0.69751	0.73216	0.80933	0.70768	0.52114	1.1085	0.14797
w_{29j}^{σ}	0.60945	0.96473	0.61227	0.22958	0.70149	0.42839	0.26822	0.79435	0.75803	0.638	0.58957	0.79856	0.41623	0.44972	0.37194	0.75494	0.48047
w_{30j}^{σ}	0.45457	0.70841	0.31409	0.26151	0.20605	0.46673	0.31135	0.71128	0.80345	0.28059	0.88074	0.30008	0.39847	0.46921	0.41028	1.0876	0.68414
w_{31j}^{σ}	0.67919	0.77237	0.60588	0.31013	0.85848	0.70986	0.76842	0.75252	0.45712	0.27413	0.63798	0.1613	0.76346	0.83725	0.56703	0.98203	0.15818
w_{32j}^{σ}	0.60838	0.47827	0.41821	0.60044	0.79284	0.25035	0.1612	1.0622	0.56189	0.64451	1.4527	0.31217	0.51066	0.54356	0.23149	0.80265	0.79298
w_{33j}^{σ}	0.71155	0.83293	0.33549	0.26859	0.67977	0.23779	0.87619	0.89601	0.33017	0.23308	0.83381	0.45003	0.89883	0.74759	0.86638	0.7918	0.29978
w_{34j}^{σ}	0.58427	0.25491	0.93389	0.67995	0.47945	0.54042	0.40455	0.34101	0.69147	0.4839	0.52694	0.1454	0.67835	0.35728	0.85525	0.68816	0.17303

Πινάκας 3.3.3.13: Διασπορές των προσυναπτικών βαρών

Rule #(j)	v_{j1}^{c}	v_{j2}^{c}
1	0.015503	0.98062
2	-0.0049165	1.0049
3	0.93464	0.064049
4	0.10548	0.18919
5	0.74153	0.59847
6	-0.0058952	1.0099
7	0.17463	0.71444
8	-0.0030825	1.0031
9	0.28364	0.64043
10	0.89129	0.79618
11	0.99276	0.0072359
12	0.82261	0.12396
13	1.0694	-0.064296
14	0.22816	0.38207
15	0.45731	0.54843
16	-0.0015366	1.0015
17	0.40559	0.53959

Πινάκας 3.3.3.14: Κέντρα των μετασυναπτικών βαρών

Rule #(j)	v_{j1}^{σ}	v_{j2}^{σ}
1	0.47705	0.66346
2	0.65036	0.76059
3	0.68591	0.81044
4	0.28297	0.32792
5	0.76668	0.62794
6	0.36761	0.34315
7	0.28026	0.72181
8	0.78383	0.92765
9	0.45379	0.56214
10	0.82418	0.75723
11	0.46154	0.53835
12	0.49431	0.32701
13	0.49003	0.62886
14	0.36029	0.46725
15	0.54457	0.54879
16	0.81172	0.94543
17	0.63168	0.4194

Πινάκας 3.3.3.15: Διασπορές των μετασυναπτικών βαρών

Παράρτημα 4

Τεχνικά Χαρακτηριστικά Συσκευών

Παράρτημα 4

Τεχνικά Χαρακτηριστικά Συσκευών

Π 4.1 PDA - Φορητή Συσκευή

Η φορητή συσκευή είναι Hewlett Packard μοντέλο i-Paq σειρά H5450. Η οθόνη είναι υγρών κρυστάλλων τύπου TFT 65.536 χρωμάτων με διαστάσεις εικονοστοιχείου 0.24mm. Έχει μνήμη 64MB και φέρει υποδοχές USB για να συνδέεται είτε στην βάση της είτε απευθείας σε PC. Το εγκατεστημένο λειτουργικό σύστημα είναι το Microsoft Windows for PocketPC έκδοση 2002. Επιπλέον απαιτεί παροχή ρεύματος 1250 mAh και διαθέτει μπαταρία πολυμερών Λιθίου (επαναφορτιζόμενη και αποσπώμενη). Ο επεξεργαστής του είναι Intel XSCALE σε συχνότητα 400 MHz, με εσωτερική μνήμη 48MB τύπου flash ROM. Υπάρχει η δυνατότητα εγκατάστασης της JVM Jeode. Η JVM είναι μια εικονική μηχανή πλήρως συμβατή με την Personal Java 1.2 και μπορεί χρησιμοποιηθεί είτε σαν πρόσθετο πρόγραμμα στο i-Paq για να εκτελούνται applets είτε σαν ξεχωριστό πρόγραμμα για να εκτελούνται εφαρμογές Java.

Π 4.2 Αισθητήρες - Εξυπηρετητής

Οι αισθητήρες εκτελούνται σε ένα φορητό υπολογιστή (laptop) Pentium III 500 MHz με μνήμη RAM 128MB ενώ ο εξυπηρετητής εκτελείται σε PC Pentium IV 3000 MHz με μνήμη RAM 1GB

Π 4.3 Ασύρματες Συνδέσεις

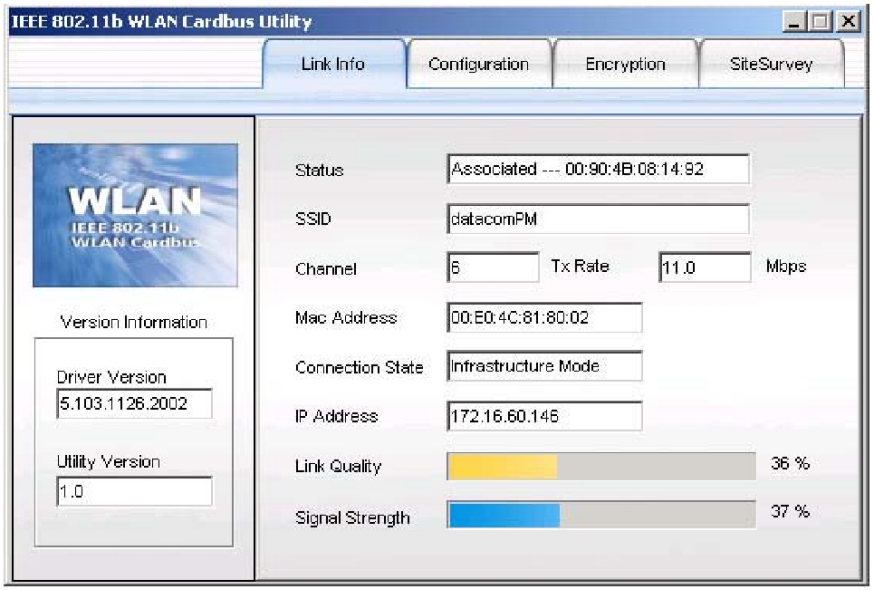
Το πρότυπο πάνω στο οποίο βασίστηκαν οι ασύρματες συνδέσεις είναι το IEEE 802.11. Επιπλέον χρησιμοποιήθηκαν:

- ένας προσαρμογέας PCMCIA Texas Instrument PCI-1225 CardBus για να επιτευχθεί η ασύρματη σύνδεση PDA και φορητού υπολογιστή.
- ένα access point για την σύνδεση του PDA με το δίκτυο κορμού.

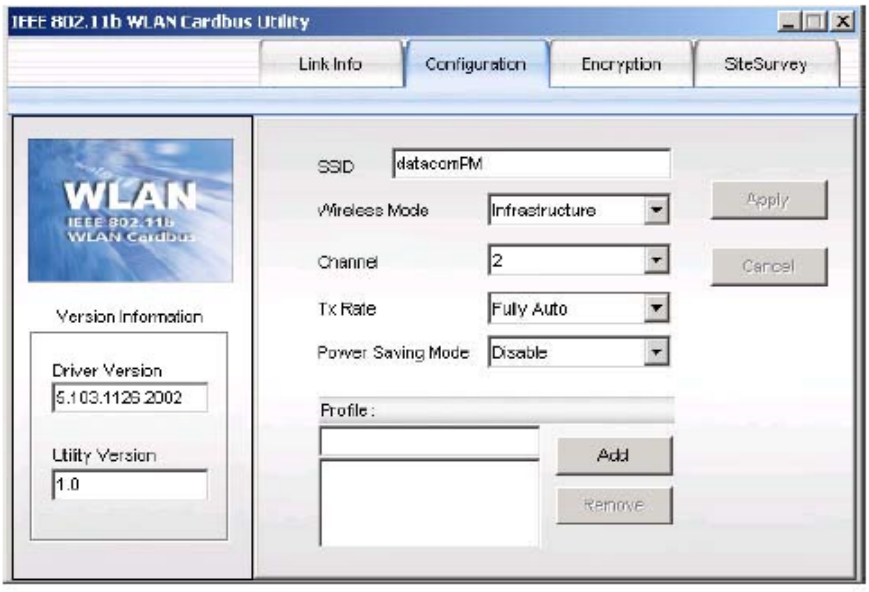
Π 4.3.1 Ασύρματος Προσαρμογέας

Ο προσαρμογέας είναι μια ασύρματη κάρτα τύπου PCMCIA που υλοποιεί το πρωτόκολλο IEEE 802.11 και είναι συμβατή κατά Wi-Fi (επιτροπή προτυποποίησης προϊόντων ασύρματων τοπικών δικτύων που βασίζονται στο πρωτόκολλο αυτό). Το εύρος συχνοτήτων είναι από 2.4 GHz έως 2.4835 GHz . Η τεχνική μετάδοσης είναι DSSS (Direct Sequence Spread Spectrum). Η διεπαφή είναι 32-bit CardBus. Ο ρυθμός μετάδοσης δεδομένων είναι δυναμικός και διαβαθμίζεται στα 11/5.5/2/1 Mbps. Η εμβέλεια λειτουργίας σε εσωτερικό χώρο είναι από 35 έως 100 μέτρα, ενώ σε εξωτερικό περιβάλλον είναι από 200 έως 350 μέτρα. Τέλος το χρησιμοποιούμενο MAC πρωτόκολλο είναι το CSMA/CA με ACK.

Οι ρυθμίσεις του προσαρμογέα επιτυγχάνονται μέσω γραφικού περιβάλλοντος.

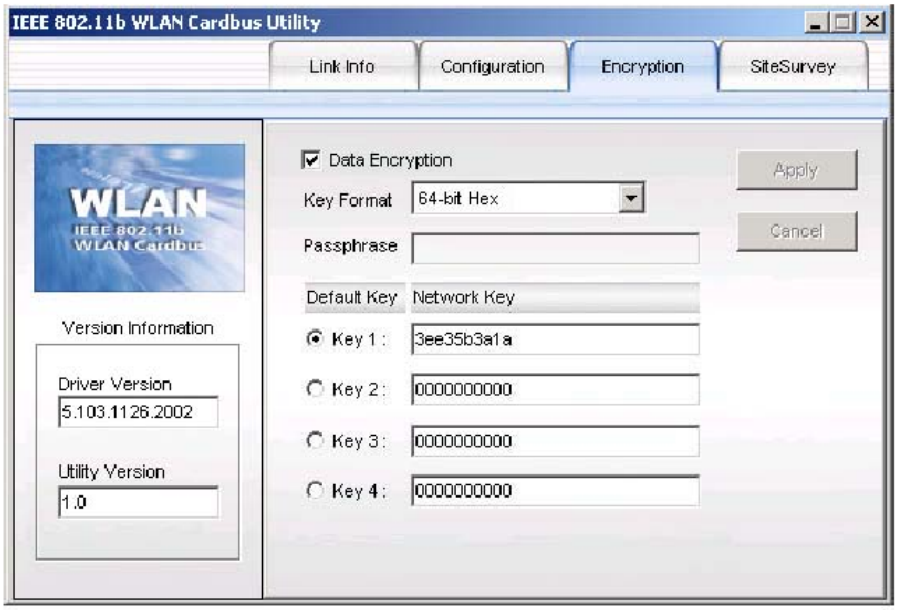


– Status: η κατάσταση στην οποία βρίσκεται η σύνδεση ("Associated" ή "Ad-Hoc" ή ανεπιτυχής)
– SSID: το αναγνωριστικό του εκάστοτε ασύρματου τοπικού δικτύου
– Channel: δείχνει το κανάλι το οποίο χρησιμοποιείται είτε από Infrastructure είτε από Ad Hoc δίκτυα
– Mac Address: δείχνει την Mac διεύθυνση του σταθμού
– Connection State: δείχνει τον τρόπο ασύρματης σύνδεσης
– IP Address: δείχνει την IP διεύθυνση
– Link Quality / Signal Strength: δείκτης της ποιότητας του σήματος και της ποιότητας της σύνδεσης
– Tx Rate: δείχνει τον ρυθμό μετάδοσης δεδομένων



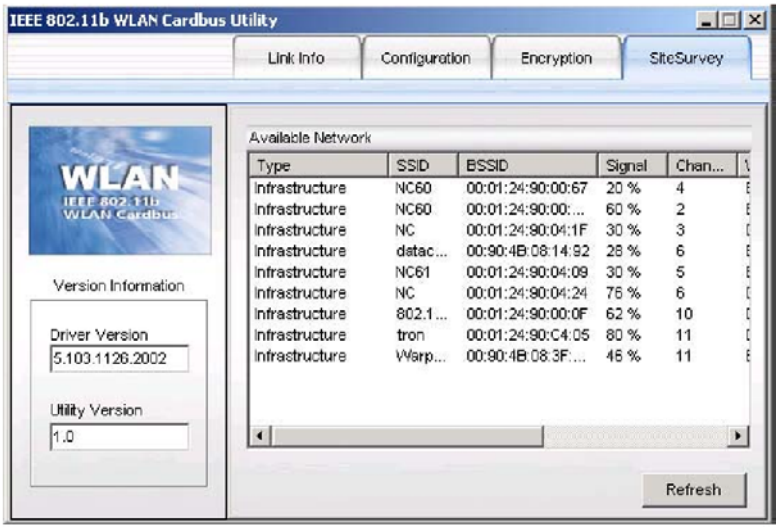
Στο ταμπλό "Configuration" υπάρχει η δυνατότητα εισαγωγής και τροποποίησης των ρυθμίσεων ενός δικτύου.

Στο ταμπλό "Encryption" δίνεται η δυνατότητα καθορισμού της ασφάλειας ενός δικτύου. Κάθε σταθμός στο δίκτυο θα πρέπει να έχει ενεργοποιημένη την συνάρτηση Encryption και οι τιμές του πεδίου Network Key θα πρέπει να είναι κοινές.



Από το μενού του Key Format παρέχονται οι επόμενες επιλογές:

- **Manual 64-bit ASCII:** ορίζεται κλειδί κρυπτογράφησης από 8 χαρακτήρες
- **Manual 128-bit ASCII:** ορίζεται κλειδί κρυπτογράφησης από 16 χαρακτήρες
- **Manual 64-bit Hex:** επιτρέπει τιμές 8 χαρακτήρων από τα πεδία 0~9 και a/A~f/F (για τον καθορισμό του κλειδιού κρυπτογράφησης)
- **Manual 128-bit Hex:** επιτρέπει τιμές 16 χαρακτήρων από τα πεδία 0~9 και a/A~f/F (για τον καθορισμό του κλειδιού κρυπτογράφησης)
- **64-bit Passphrase:** σε αυτή την επιλογή μόλις πληκτρολογηθεί η λέξη pass, δημιουργείται αυτόματα κλειδί κρυπτογράφησης 8 byte στο Key 1 πεδίο
- **128-bit Passphrase:** σε αυτή την επιλογή μόλις πληκτρολογηθεί η λέξη pass, δημιουργείται αυτόματα κλειδί κρυπτογράφησης 16 byte στο Key 1 πεδίο



Στο ταμπλό "SiteSurvey" δίνεται η δυνατότητα να αποκτήσουμε πληροφορίες για όλα τα δίκτυα που βρίσκονται εντός της εμβέλειας λειτουργίας.

Π 4.3.2 Σημείο Πρόσβασης (Access Point)

Το σημείο πρόσβασης είναι το μοντέλο DWL-1000 AP της εταιρείας D-Link με περιγραφή Air Premier Enterprise 2.4GHz (IEEE 802.11) Wireless Access Point. Ο ρυθμός μετάδοσης μπορεί να διαβαθμιστεί στα 22/11/5.5/2/1 Mbps. Υποστηρίζει κρυπτογράφηση 64, 128, 256 bit WEP (Wired Equivalent Privacy). Η πρόσβαση στο μέσο γίνεται βάση του αλγορίθμου CSMA/CK με ACK. Το εύρος συχνοτήτων είναι από 2.4 GHz έως 2.4835 GHz και η εμβέλεια για εσωτερικό χώρο κυμαίνεται στα 100 μέτρα ενώ για εξωτερικό χώρο κυμαίνεται στα 400 μέτρα.

Παράρτημα 5

Πηγαίοι Κώδικες σε Γλώσσα Προγραμματισμού Java

Παράρτημα 5

Πηγαίοι Κώδικες σε Γλώσσα Προγραμματισμού Java

```
//************
// Supfunis.java
//************
class Supfunis
{
   static final double SQRTPI = Math.sqrt(Math.PI);

   static double erf(double x)          //υπολογισμός της συνάρτησης λάθους.
   {
      double t,z,value;

      z = Math.abs(x);
      t = 1.0/(1.0+0.5*z);
      value = t*Math.exp(-z*z-1.26551223+t*(1.00002368+t*(0.37409196+
            t*(0.09678418+t*(-0.18628806+t*(0.27886807+t*(-1.13520398+
            t*(1.48851587+t*(-0.82215223+t*0.17087277)))))))));
      if(x<0.0)
         value = 2.0-value;
      value = 1.0-value;
      return value;
   }
   //υπολογισμός του μέτρου της τομής δυο Gauss συναρτήσεων συμμετοχής.
   static double intersection(double xc,double xs,double wc,double ws)
   {
      double value,h1,h2;

      if(xc==wc)              //περίπτωση 1: c1=c2 με τα σ1, σ2 να έχουν οποιεσδήποτε τιμές.
      {
         if(xs<ws)
            value = SQRTPI*xs;
         else
            value = SQRTPI*ws;
      }
      else
      {
         h1 = (xc+xs*wc/ws)/(1.0+xs/ws);
         if(xs==ws)           //περίπτωση 2: c1≠c2, σ1=σ2.
         {
            if(wc>xc)
               value = SQRTPI*ws*(0.5+0.5*erf((h1-wc)/ws))+
                     SQRTPI*xs*(0.5-0.5*erf((h1-xc)/xs));
            else
               value = SQRTPI*xs*(0.5+0.5*erf((h1-xc)/xs))+
                     SQRTPI*ws*(0.5-0.5*erf((h1-wc)/ws));
         }
         else
         {
            h2 = (xc-xs*wc/ws)/(1.0-xs/ws);
            if(xs<ws)         //περίπτωση 3: c1≠c2, σ1>σ2.
            {
               if(xc>wc)
                  value = SQRTPI*xs*(0.5+0.5*erf((h1-xc)/xs))+
                        SQRTPI*xs*(0.5-0.5*erf((h2-xc)/xs))+
```

```
                    SQRTPI*ws*(0.5*erf((h2-wc)/ws)-0.5*erf((h1-wc)/ws));
            else
               value = SQRTPI*xs*(0.5+0.5*erf((h2-xc)/xs))+
                    SQRTPI*xs*(0.5-0.5*erf((h1-xc)/xs))+
                    SQRTPI*ws*(0.5*erf((h1-wc)/ws)-0.5*erf((h2-wc)/ws));
         }
         else                    //περίπτωση 4: c1≠c2, σ1<σ2.
         {
            if(wc>xc)
               value = SQRTPI*ws*(0.5+0.5*erf((h1-wc)/ws))+
                    SQRTPI*ws*(0.5-0.5*erf((h2-wc)/ws))+
                    SQRTPI*xs*(0.5*erf((h2-xc)/xs)-0.5*erf((h1-xc)/xs));
            else
               value = SQRTPI*ws*(0.5+0.5*erf((h2-wc)/ws))+
                    SQRTPI*ws*(0.5-0.5*erf((h1-wc)/ws))+
                    SQRTPI*xs*(0.5*erf((h1-xc)/xs)-0.5*erf((h2-xc)/xs));
         }
      }
   }
   return value;
}
//υπολογισμός του μέτρου ομοιότητας δυο ασαφών συνόλων.
static double mutualSubsethood(double xc,double xs,double wc,double ws)
{
   double conjuction;

   conjuction = intersection(xc,xs,wc,ws);
   return conjuction/(SQRTPI*xs+SQRTPI*ws-conjuction);
}
//υπολογισμός του σήματος στρώματος εξόδου.
static double[] testing(double xc[],double xs[],double wc[][],double ws[][],
                        double vc[][],double vs[][])
{
   int n = xs.length;
   int q = ws[0].length;
   int p = vs[0].length;
   double z[] = new double[q];
   double y[] = new double[p];
   int i,j,k;

   for(j = 0;j<q;j++)      //υπολογισμός του ασαφούς εσωτερικού γινομένου.
   {
      z[j] = 1.0;
      for(i = 0;i<n;i++)
         z[j] *= mutualSubsethood(xc[i],xs[i],wc[i][j],ws[i][j]);
   }
   for(k = 0;k<p;k++)    //η τροποποιημένη μέθοδος απο-ασαφοποίησης κέντρου βάρους.
   {
      y[k] = 0.0;
      double sum = 0.0;
      for(j = 0;j<q;j++)
      {
         y[k] += z[j]*vc[j][k]*vs[j][k];
         sum += z[j]*vs[j][k];
      }
      y[k] /= sum;
   }
   return y;
}
static int getMaxIndex(double pin[])        //εύρεση του στοιχείου πίνακα με την μεγαλύτερη τιμή.
```

```
    {
        int value;

        value = 0;
        for(int i = 1;i<pin.length;i++)
            if(pin[i]>pin[value])
                value = i;
        return value;
    }
    //υπολογισμός του πλήθους λανθασμένων ταξινομήσεων.
    static int countResubstitutions(double data[][],double xs[],double wc[][],
                                    double ws[][],double vc[][],double vs[][])
    {
        double xc[] = new double[data[0].length-1];
        int value = 0;
        int store;

        for(int rows = 0;rows<data.length;rows++)
        {
            copy(data[rows],xc);
            store = getMaxIndex(testing(xc,xs,wc,ws,vc,vs));        //εύρεση του αποτελέσματος με την
                                                                    //λογική winner takes all.
            if(((double)(store+1))!=data[rows][data[0].length-1])   //έλεγχος για το κατά πόσο η
                value++;                                            //ταξινόμηση είναι αποτυχημένη.
        }
        return value;
    }
    static void copy(double source[],double destin[])    //αντιγραφή πινάκων στοιχείο προς στοιχείο.
    {
        for(int i = 0;i<destin.length;i++)
            destin[i] = source[i];
    }
}



//************
// Learning.java
//************
class Learning     //αλγόριθμος backpropagation με την μέθοδο κλήσης ⊕ υβριδική μάθηση.
{
    double SQRTPI = Math.sqrt(Math.PI);
    int n;
    int q;
    int p;
    double xc[];              //πίνακες που περιέχουν τα διανύσματα βαρών του νευρο-ασαφούς
    double xs[];              //συστήματος κατά την t-οστή εκτέλεση του αλγορίθμου.
    double wc[][];
    double ws[][];
    double vc[][];
    double vs[][];
    double y[];               //πίνακας που περιέχει το σήμα εξόδου.
    double d[];               //πίνακας που περιέχει το ιδανικό πρότυπο εξόδου.
    double z[];               //πίνακας με τα ασαφή εσωτερικά γινόμενα.
    double E[][];             //πίνακας με τα μέτρα ομοιότητας.
    double pdcwc[][];         //πίνακες που περιέχουν τις μερικές παραγώγους της τομής
    double pdcws[][];         //δυο ασαφών συνόλων.
    double pdcxs[][];
```

```
//εκτέλεση ενός βήματος του backpropagation αλγορίθμου ή ενός βήματος του αλγορίθμου
//υβριδικής μάθησης αναλόγως του ορίσματος d[].
void gradientDescent(double h,double a,double d[],double xc[],double xs[],double wc[][],
                     double ws[][],double vc[][],double vs[][],double pxs[],
                     double pwc[][],double pws[][],double pvc[][],double pvs[][])
{
   int i,j,k;
   double fxs[];          //πίνακες που περιέχουν τα διανύσματα βαρών του νευρο-ασαφούς
   double fwc[][];        //συστήματος κατά την (t+1)-οστή εκτέλεση του αλγορίθμου.
   double fws[][];
   double fvc[][];
   double fvs[][];

   this.xc = xc;
   this.xs = xs;
   this.wc = wc;
   this.ws = ws;
   this.vc = vc;
   this.vs = vs;
   n = xs.length;
   q = ws[0].length;
   p = vs[0].length;
   fxs = new double[n];
   fwc = new double[n][q];
   fws = new double[n][q];
   fvc = new double[q][p];
   fvs = new double[q][p];
   E = new double[n][q];
   z = new double[q];
   y = new double[p];
   for(j = 0;j<q;j++)                //υπολογισμός του ασαφούς εσωτερικού γινομένου.
   {
      z[j] = 1.0;
      for(i = 0;i<n;i++)
      {
         E[i][j] = Supfunis.mutualSubsethood(xc[i],xs[i],wc[i][j],ws[i][j]);
         z[j] *= E[i][j];
      }
   }
   for(k = 0;k<p;k++)    //η τροποποιημένη μέθοδος απο-ασαφοποίησης κέντρου βάρους.
   {
      y[k] = 0.0;
      double sum = 0.0;
      for(j = 0;j<q;j++)
      {
         y[k] += z[j]*vc[j][k]*vs[j][k];
         sum += z[j]*vs[j][k];
      }
      y[k] /= sum;
   }
   if(d==null)              //εκτίμηση της κατηγορίας για την περίπτωση του αλγορίθμου
   {                        //υβριδικής μάθησης.
      d = new double[p];
      copy(y,d);
      int win = Supfunis.getMaxIndex(d);
      for(k = 0;k<p;k++)
         if(k==win)
            d[k] = 1.0;
         else
            d[k] = 0.0;
```

```
    }
    this.d = d;
    createPartialDerivatives();      //κατασκευή των πινάκων που περιέχουν την μερική παράγωγο
                                     //κάθε ασαφούς τομής ως προς τα αντίστοιχα κέντρα και
                                     //διασπορές.
    for(i = 0;i<n;i++)               //ανανέωση της διασποράς των κόμβων εισόδου.
    {
       fxs[i] = xs[i]-h*pdexs(i)+a*(xs[i]-pxs[i]);
       if(fxs[i]<=0.0)
          fxs[i] = 0.001;
    }
    for(i = 0;i<n;i++)       //ανανέωση του κέντρου και της διασποράς των προσυναπτικών βαρών.
       for(j = 0;j<q;j++)
       {
          fwc[i][j] = wc[i][j]-a*pdewc(i,j)+a*(wc[i][j]-pwc[i][j]);
          fws[i][j] = ws[i][j]-a*pdews(i,j)+a*(ws[i][j]-pws[i][j]);
          if(fws[i][j]<=0.0)
             fws[i][j] = 0.001;
       }
    for(j = 0;j<q;j++)       //ανανέωση του κέντρου και της διασποράς των μετασυναπτικών βαρών.
       for(k = 0;k<p;k++)
       {
          fvc[j][k] = vc[j][k]-a*pdevc(j,k)+a*(vc[j][k]-pvc[j][k]);
          fvs[j][k] = vs[j][k]-a*pdevs(j,k)+a*(vs[j][k]-pvs[j][k]);
          if(fvs[j][k]<=0.0)
             fvs[j][k] = 0.001;
       }
     copy(xs,pxs);           //ενημέρωση των παραμέτρων του συστήματος περί των
     copy(wc,pwc);           //καινούργιων ανανεωμένων τιμών.
     copy(ws,pws);
     copy(vc,pvc);
     copy(vs,pvs);
     copy(fxs,xs);
     copy(fwc,wc);
     copy(fws,ws);
     copy(fvc,vc);
     copy(fvs,vs);
}
void copy(double source[],double destin[])          //αντιγραφή πινάκων στοιχείο προς στοιχείο.
{
   for(int i = 0;i<source.length;i++)
      destin[i] = source[i];
}
void copy(double source[][],double destin[][])      //αντιγραφή δισδιάστατων πινάκων
{                                                   //στοιχείο προς στοιχείο.
   for(int i = 0;i<source.length;i++)
      for(int j = 0;j<source[0].length;j++)
         destin[i][j] = source[i][j];
}

double pdevc(int j,int k)          //υπολογισμός του  ∂e/∂v_jk^c .

{
   double value;

   value = 0.0;
   for(int s = 0;s<q;s++)
      value += z[s]*vs[s][k];
   value = -(d[k]-y[k])*z[j]*vs[j][k]/value;
   return value;
```

```
}
double pdevs(int j,int k)          //υπολογισμός του ∂e/∂v_jk^σ .
{
   double value;

   double sum = 0.0;
   double temp = 0.0;
   for(int s = 0;s<q;s++)
   {
      sum += z[s]*vs[s][k];
      temp += z[s]*vc[s][k]*vs[s][k];
   }
   value = z[j]*vc[j][k]*sum-z[j]*temp;
   value = -(d[k]-y[k])*value/Math.pow(sum,2.0);
   return value;
}
double pdewc(int i,int j)          //υπολογισμός του ∂e/∂w_ij^c .
{
   double value;

   value = 0.0;
   for(int k = 0;k<p;k++)
      value += -(d[k]-y[k])*pdyz(k,j)*pdzE(j,i)*pdEwc(i,j);
   return value;
}
double pdews(int i,int j)          //υπολογισμός του ∂e/∂w_ij^σ .
{
   double value;

   value = 0.0;
   for(int k = 0;k<p;k++)
      value += -(d[k]-y[k])*pdyz(k,j)*pdzE(j,i)*pdEws(i,j);
   return value;
}
double pdexs(int i)                //υπολογισμός του ∂e/∂x_i^σ .
{
   double value;

   value = 0.0;
   for(int j = 0;j<q;j++)
      for(int k = 0;k<p;k++)
         value += -(d[k]-y[k])*pdyz(k,j)*pdzE(j,i)*pdExs(i,j);
   return value;
}
double pdyz(int k,int j)           //υπολογισμός του ∂y_k/∂z_j .
{
   double value;

   value = 0.0;
   for(int s = 0;s<q;s++)
```

```
        value += z[s]*vs[s][k];
    value = vs[j][k]*(vc[j][k]-y[k])/value;
    return value;
}

double pdzE(int j,int i)              //υπολογισμός του ∂z_j/∂E_ij .

{
    double value;

    value = 1.0;
    for(int r = 0;r<n;r++)
        if(r!=i)
            value *= E[r][j];
    return value;
}

double pdEwc(int i,int j)                    //υπολογισμός του ∂E_ij/∂w_ij^c .

{
    double value;

    double c = E[i][j]*SQRTPI*(xs[i]+ws[i][j])/(1+E[i][j]);
    value = pdcwc[i][j]*(SQRTPI*(xs[i]+ws[i][j])-c)-(-pdcwc[i][j]*c);
    value /= Math.pow(SQRTPI*(xs[i]+ws[i][j])-c,2.0);
    return value;
}

double pdEws(int i,int j)             //υπολογισμός του ∂E_ij/∂w_ij^σ .

{
    double value;

    double c = E[i][j]*SQRTPI*(xs[i]+ws[i][j])/(1+E[i][j]);
    value = pdcws[i][j]*(SQRTPI*(xs[i]+ws[i][j])-c)-((SQRTPI-pdcws[i][j])*c);
    value /= Math.pow(SQRTPI*(xs[i]+ws[i][j])-c,2.0);
    return value;
}

double pdExs(int i,int j)             //υπολογισμός του ∂E_ij/∂x_i^σ .

{
    double value;

    double c = E[i][j]*SQRTPI*(xs[i]+ws[i][j])/(1+E[i][j]);
    value = pdcxs[i][j]*(SQRTPI*(xs[i]+ws[i][j])-c)-((SQRTPI-pdcxs[i][j])*c);
    value /= Math.pow(SQRTPI*(xs[i]+ws[i][j])-c,2.0);
    return value;
}

void createPartialDerivatives()    //υπολογισμός των ∂C(s_i ∩ w_ij)/∂w_ij^c , ∂C(s_i ∩ w_ij)/∂w_ij^σ ,
∂C(s_i ∩ w_ij)/∂x_i^σ .

{
    pdcwc = new double[n][q];
    pdcws = new double[n][q];
    pdcxs = new double[n][q];
```

```
for(int i = 0;i<n;i++)
   for(int j = 0;j<q;j++)
   {
   if(xc[i]==wc[i][j])              //περίπτωση 1: c1=c2 με τα σ1, σ2 να έχουν οποιεσδήποτε τιμές.
   {
      if(xs[i]<ws[i][j])
      {
         pdcwc[i][j] = 0;
         pdcws[i][j] = 0;
         pdcxs[i][j] = SQRTPI;
      }
      else if(xs[i]>ws[i][j])
      {
         pdcwc[i][j] = 0;
         pdcws[i][j] = SQRTPI;
         pdcxs[i][j] = 0;
      }
      else
      {
         pdcwc[i][j] = 0;
         pdcws[i][j] = SQRTPI;
         pdcxs[i][j] = SQRTPI;
      }
   }
   else
   {
      double h1 = (xc[i]+xs[i]*wc[i][j]/ws[i][j])/(1.0+xs[i]/ws[i][j]);
      if(xs[i]==ws[i][j])           //περίπτωση 2: c1≠c2, σ1=σ2.
      {
         if(wc[i][j]>xc[i])
         {
            pdcwc[i][j] = -1.0*Math.exp(-1.0*Math.pow((h1-wc[i][j])/ws[i][j],2.0));
            pdcws[i][j] = ((h1-wc[i][j])/ws[i][j])*pdcwc[i][j]+SQRTPI*0.5*
                                 (1+Supfunis.erf((h1-wc[i][j])/ws[i][j]));
            pdcxs[i][j] = ((h1-xc[i])/xs[i])*Math.exp(-1.0*Math.pow((h1-xc[i])/xs[i],2.0));
            pdcxs[i][j] += SQRTPI*0.5*(1-Supfunis.erf((h1-xc[i])/xs[i]));
         }
         else
         {
            pdcwc[i][j] = Math.exp(-1.0*Math.pow((h1-wc[i][j])/ws[i][j],2.0));
            pdcws[i][j] = ((h1-wc[i][j])/ws[i][j])*pdcwc[i][j]+SQRTPI*0.5*
                                 (1-Supfunis.erf((h1-wc[i][j])/ws[i][j]));
            pdcxs[i][j] = -((h1-xc[i])/xs[i])*Math.exp(-1.0*Math.pow((h1-xc[i])/xs[i],2.0));
            pdcxs[i][j] += SQRTPI*0.5*(1+Supfunis.erf((h1-xc[i])/xs[i]));
         }
      }
      else
      {
         double h2 = (xc[i]-xs[i]*wc[i][j]/ws[i][j])/(1.0-xs[i]/ws[i][j]);
         if(h1>h2)
         {
            double swap = h2;
            h2 = h1;
            h1 = swap;
         }
         if(xs[i]<ws[i][j])          //περίπτωση 3: c1≠c2, σ1>σ2.
         {
            pdcwc[i][j] = -Math.exp(-1.0*Math.pow((h2-wc[i][j])/ws[i][j],2.0));
            pdcwc[i][j] += Math.exp(-1.0*Math.pow((h1-wc[i][j])/ws[i][j],2.0));
```

```
                pdcws[i][j] = ((h1-wc[i][j])/ws[i][j])*
                                Math.exp(-1.0*Math.pow((h1-wc[i][j])/ws[i][j],2.0));
                pdcws[i][j] += -((h2-wc[i][j])/ws[i][j])*
                                Math.exp(-1.0*Math.pow((h2-wc[i][j])/ws[i][j],2.0));
                pdcws[i][j] += SQRTPI*0.5*(-Supfunis.erf((h1-wc[i][j])/ws[i][j])+
                                Supfunis.erf((h2-wc[i][j])/ws[i][j]));
                pdcxs[i][j] = -((h1-xc[i])/xs[i])*Math.exp(-1.0*Math.pow((h1-xc[i])/xs[i],2.0));
                pdcxs[i][j] += ((h2-xc[i])/xs[i])*Math.exp(-1.0*Math.pow((h2-xc[i])/xs[i],2.0));
                pdcxs[i][j] += SQRTPI*(1+0.5*Supfunis.erf((h1-xc[i])/xs[i])-0.5*
                                Supfunis.erf((h2-xc[i])/xs[i]));
            }
            else                    //περίπτωση 3: c1≠c2, σ1<σ2.
            {
                pdcwc[i][j] = -Math.exp(-1.0*Math.pow((h1-wc[i][j])/ws[i][j],2.0));
                pdcwc[i][j] += Math.exp(-1.0*Math.pow((h2-wc[i][j])/ws[i][j],2.0));
                pdcws[i][j] = -((h1-wc[i][j])/ws[i][j])*
                                Math.exp(-1.0*Math.pow((h1-wc[i][j])/ws[i][j],2.0));
                pdcws[i][j] += ((h2-wc[i][j])/ws[i][j])*
                                Math.exp(-1.0*Math.pow((h2-wc[i][j])/ws[i][j],2.0));
                pdcws[i][j] += SQRTPI*(1+0.5*Supfunis.erf((h1-wc[i][j])/ws[i][j])-0.5*
                                Supfunis.erf((h2-wc[i][j])/ws[i][j]));
                pdcxs[i][j] = ((h1-xc[i])/xs[i])*Math.exp(-1.0*Math.pow((h1-xc[i])/xs[i],2.0));
                pdcxs[i][j] += -((h2-xc[i])/xs[i])*Math.exp(-1.0*Math.pow((h2-xc[i])/xs[i],2.0));
                pdcxs[i][j] += SQRTPI*0.5*(-Supfunis.erf((h1-xc[i])/xs[i])+
                                Supfunis.erf((h2-xc[i])/xs[i]));
            }
          }
        }
        }
    }
}



//***********
// IOFiles.java
//***********
import java.io.*;

class IOFiles
{
   static boolean blank(int ascii)      //έλεγχος εάν η ascii τιμή αντιστοιχεί σε διαχωριστικό χαρακτήρα.
   {
      switch (ascii)
      {
         case (int)' ':
         case (int)'\t':
         case (int)'\r':
         case (int)'\n':
               return true;
         default:
               return false;
      }
   }
   //μετατροπή αρχείου που περιέχει αριθμούς χωρισμένους με (οποιοδήποτε αριθμό ή οποιαδήποτε
   //διάταξη από) διαχωριστικούς χαρακτήρες, σε byte αρχείο που περιέχει μόνο τους αριθμούς.
   static int fileNormalForm(String source,String destination)
   {
      int count;
```

```
    boolean isNum;
    String num;
    int ascii;

    count = 0;
    try
    {
       FileInputStream fis = new FileInputStream(source);
       FileOutputStream fos = new FileOutputStream(destination);
       BufferedOutputStream bos = new BufferedOutputStream(fos);
       DataOutputStream dos = new DataOutputStream(bos);

       isNum = false;
       num = "";
       while((ascii = fis.read())!=-1)
       {
          if(blank(ascii)&&!isNum);          //διαχωριστικός χαρακτήρας.
          else if(!blank(ascii)&&!isNum)     //αρχή αριθμού.
             {
             isNum = true;
             num += String.valueOf((char)ascii);
             }
          else if(!blank(ascii)&&isNum)              //χαρακτήρες που αντιστοιχούν στην String
             num += String.valueOf((char)ascii);     //αναπαράσταση του αριθμού.
          else if(blank(ascii)&&isNum)               //τέλος του αριθμού.
             {
                count++;
                dos.writeDouble(Double.parseDouble(num));
                num = "";
                isNum = false;
             }
       }
       if(isNum)
       {
          count++;
          dos.writeDouble(Double.parseDouble(num));
       }
       fis.close();
       dos.close();
    }
    catch(IOException e)
    {
       System.out.println("ERROR: "+e.toString());
    }
    return count;
}
//αποθήκευση δισδιάστατου πίνακα αριθμών double σε byte αρχείο.
static void arrayToFile(double array[][],String file)
{
    try
    {
       FileOutputStream fos = new FileOutputStream(file);
       BufferedOutputStream bos = new BufferedOutputStream(fos);
       DataOutputStream dos = new DataOutputStream(bos);

       for(int i = 0;i<array.length;i++)
          for(int j = 0;j<array[0].length;j++)
             dos.writeDouble(array[i][j]);
       dos.close();
    }
```

```
    catch(IOException e)
    {
       System.out.println("ERROR: "+e.toString());
    }
}
//αποθήκευση πίνακα αριθμών double σε byte αρχείο.
static void arrayToFile(double array[],String file)
{
    try
    {
       FileOutputStream fos = new FileOutputStream(file);
       BufferedOutputStream bos = new BufferedOutputStream(fos);
       DataOutputStream dos = new DataOutputStream(bos);

       for(int i = 0;i<array.length;i++)
          dos.writeDouble(array[i]);
       dos.close();
    }
    catch(IOException e)
    {
       System.out.println("ERROR: "+e.toString());
    }
}
//ανάκτηση των αριθμών double που είναι αποθηκευμένοι σε byte αρχείο
//και τοποθέτηση τους σε πίνακα.
static double[] fileToArray(String file,int elements)
{
    double value[] = new double[elements];
    try
    {
       FileInputStream fis = new FileInputStream(file);
       BufferedInputStream bis = new BufferedInputStream(fis);
       DataInputStream dis = new DataInputStream(bis);

       for(int i = 0;i<value.length;i++)
          value[i] = dis.readDouble();
       dis.close();
    }
    catch(IOException e)
    {
       System.out.println("ERROR: "+e.toString());
    }
    return value;
}
//ανάκτηση των αριθμών double που είναι αποθηκευμένοι σε byte αρχείο
//και τοποθέτηση τους σε δισδιάστατο πίνακα.
static double[][] fileToArray(String file,int rows,int columns)
{
    double value[][] = new double[rows][columns];
    try
    {
       FileInputStream fis = new FileInputStream(file);
       BufferedInputStream bis = new BufferedInputStream(fis);
       DataInputStream dis = new DataInputStream(bis);

       for(int i = 0;i<value.length;i++)
          for(int j = 0;j<value[0].length;j++)
             value[i][j] = dis.readDouble();
       dis.close();
    }
```

```
      catch(IOException e)
      {
         System.out.println("ERROR: "+e.toString());
      }
      return value;
   }

   //εμφάνιση των περιεχομένων double πίνακα.
   static void show(double pin[])
   {
      for(int i = 0;i<pin.length;i++)
         System.out.print(pin[i]+" ");
   }
   //εμφάνιση των περιεχομένων δισδιάστατου double πίνακα.
   static void show(double pin[][])
   {
      for(int i = 0;i<pin.length;i++)
      {
         for(int j = 0;j<pin[0].length;j++)
            System.out.print(pin[i][j]+" ");
         System.out.println();
      }
   }

   public static void main(String args[])
   {
      String from = args[0];
      String to = args[1];
   }

}



//***********
// Double.java
//***********
class Double       //wrapper κλάση του πρωταρχικού τύπου double.
{
   //μετατροπή της αλφαριθμητικής αναπαράστασης ενός οποιουδήποτε αριθμού σε double.
   static double parseDouble(String str)
   {
      double value;
      int decimalPoint,exponent;

      decimalPoint = str.indexOf('.');
      exponent = 1+str.indexOf('e')+str.indexOf('E');
      if((exponent==-1)&&(decimalPoint==-1))        //η αναπαράσταση του αριθμού δεν περιλαμβάνει
         value = (double)Long.parseLong(str);       //ούτε υποδιαστολή ούτε εκθέτη.
      else if(exponent==-1)                         //η αναπαράσταση του αριθμού περιλαμβάνει
      {                                             //μόνο υποδιαστολή.
         value = (double)Long.parseLong(str.substring(0,decimalPoint)+
                                 str.substring(decimalPoint+1,str.length()));
         value /= Math.pow(10.0,(double)str.substring(decimalPoint+1,str.length()).length());
      }
      else if(decimalPoint==-1)                     //η αναπαράσταση του αριθμού περιλαμβάνει
      {                                             //μόνο εκθέτη.
         value = (double)Long.parseLong(str.substring(0,exponent));
         value *= Math.pow(10.0,(double)Long.parseLong(str.substring(exponent+1,str.length())));
```

```
      }
      else                                          //η αναπαράσταση του αριθμού περιλαμβάνει
      {                                             //εκθέτη και υποδιαστολή.
         value = (double)Long.parseLong(str.substring(0,decimalPoint)+
                                 str.substring(decimalPoint+1,exponent));
         value *= Math.pow(10.0,(double)(-str.substring(decimalPoint+1,exponent).length()+
                           Long.parseLong(str.substring(exponent+1,str.length()))));
      }

      return value;
   }

//    public static void main(String args[])
//    {
//       System.out.println(Double.parseDouble(args[0]));
//    }

}



//***************
// Constraints.java
//***************
import java.awt.*;

class Constraints
{
   //συνάρτηση που διευκολύνει τον καθορισμό των παραμέτρων
   //κάθε κελιού στη διάταξη δομημένων πλεγμάτων.
   static void addConstraints(GridBagLayout gbl,Component c,GridBagConstraints gbc,
                              int gx,int gy,int gw,int gh,int wx,int wy,int ar)
   {
      gbc.gridx = gx;
      gbc.gridy = gy;
      gbc.gridwidth = gw;
      gbc.gridheight = gh;
      gbc.weightx = wx;
      gbc.weighty = wy;
      gbc.anchor = ar;
      gbl.setConstraints(c,gbc);
   }
}



//***************
// MainFrame.java
//***************
import java.awt.*;

class MainFrame extends Frame     //κλάση μου δημιουργεί το αρχικό παράθυρο
{                                 //της εφαρμογής στο PDA.
   Font f = new Font("TimesRoman",Font.BOLD,12);
   String title;                  //ο υπέρτιτλος του παραθύρου.
   String path;                   //η διαδρομή του φακέλου που περιέχει τα αρχεία της εφαρμογής.
   Choice operation;
   Button start;
```

```
Poster icon;                    //αντικείμενο για την εμφάνιση εικόνων.
TextField l[] = new TextField[10];

//δημιουργία του αρχικού παραθύρου της εφαρμογής χρησιμοποιώντας
//την διάταξη δομημένων πλεγμάτων.
MainFrame(String title,String path)
{
   super(title);
   this.title = title;
   this.path = path;
   GridBagLayout gbl = new GridBagLayout();
   GridBagConstraints gbc = new GridBagConstraints();
   setLayout(gbl);
   Label lbl;
   lbl = new Label("Operation Mode::");
   Constraints.addConstraints(gbl,lbl,gbc,0,0,1,1,0,1,GridBagConstraints.WEST);
   add(lbl);
   operation = new Choice();       //λίστα επιλογής λειτουργίας.
   operation.setFont(f);
   operation.add("");
   operation.add("Classification");
   operation.add("Training");
   Constraints.addConstraints(gbl,operation,gbc,1,0,1,1,1,0,GridBagConstraints.WEST);
   add(operation);
   start = new Button("Start");
   Constraints.addConstraints(gbl,start,gbc,1,1,1,1,0,0,GridBagConstraints.CENTER);
   add(start);
   start.setEnabled(false);
   icon = new Poster(path);        //καμβάς για την εμφάνιση εικόνων στο παράθυρο της εφαρμογής.
   gbc.fill = GridBagConstraints.BOTH;
   Constraints.addConstraints(gbl,icon,gbc,1,3,1,8,0,0,GridBagConstraints.NORTHWEST);
   add(icon);
   gbc.fill = GridBagConstraints.NONE;
   icon.setPoster(operation.getSelectedIndex());
   for(int i = 0;i<l.length;i++)     //δημιουργία πλέγματος από πεδία κειμένου τα οποία
   {                                 //εμφανίζουν πληροφορίες σχετικές με την
      l[i] = new TextField(11);      //λειτουργία που ενδέχεται να επιλέξει ο χρήστης.
      l[i].setFont(f);
      l[i].setBackground(Color.white);
      l[i].setEditable(false);
      Constraints.addConstraints(gbl,l[i],gbc,0,i+1,1,1,0,0,GridBagConstraints.WEST);
      add(l[i]);
   }
   setSize(239,293);
   setResizable(false);
   setMain();
   show();
}
void setMain() //εμφάνιση των κειμένων που αφορούν στο αρχικό υποπαράθυρο της εφαρμογής.
{
   icon.setPoster(operation.getSelectedIndex());
   l[0].setText(" ");
   l[1].setText("Wireless");
   l[2].setText("Implementation");
   l[3].setText("of");
   l[4].setText("Fuzzy");
   l[5].setText("Neural");
   l[6].setText("Classification");
   l[7].setText("System");
   l[8].setText(" ");
```

```
    l[9].setText(" ");
  }
  void setClassification()              //εμφάνιση των κειμένων που αφορούν στο παράθυρο
  {                                     //ταξινόμησης της εφαρμογής.
    icon.setPoster(operation.getSelectedIndex());
    l[0].setText("Fuzzy-Neural");
    l[1].setText("Network::");
    l[2].setText("Subsethood");
    l[3].setText("Product");
    l[4].setText("Fuzzy");
    l[5].setText("Neural");
    l[6].setText("Inference");
    l[7].setText("System");
    l[8].setText("Wireless LAN::");
    l[9].setText("Ad-Hoc 802.11b");
  }
  void setTraining()                    //εμφάνιση των κειμένων που αφορούν στο παράθυρο
  {                                     //ανανέωσης των παραμέτρων του συστήματος.
    icon.setPoster(operation.getSelectedIndex());
    l[0].setText(" ");
    l[1].setText("Learning::");
    l[2].setText("Supervised");
    l[3].setText("Gradient");
    l[4].setText("Descent");
    l[5].setText("Wireless LAN::");
    l[6].setText("Ad-Hoc or");
    l[7].setText("Infrastructure");
    l[8].setText("802.11b");
    l[9].setText(" ");
  }
  public boolean action(Event vnt,Object arg)       //χειρισμός συμβάντων που προκαλούνται από
  {                                                 //τον χρήστη.
    if(vnt.target instanceof Choice)
    {
      String str = (String)arg;
      if(str.equals(""))                    //εμφάνιση αρχικών πληροφοριών.
      {
        setMain();
        start.setEnabled(false);
      }
      else if(str.equals("Training"))       //εμφάνιση πληροφοριών σχετικών με την ταξινόμηση.
      {
        setTraining();
        start.setEnabled(true);
      }
      else                                  //εμφάνιση πληροφοριών σχετικών με την ανανέωση των βαρών.
      {
        setClassification();
        start.setEnabled(true);
      }
      return true;
    }
    else if(vnt.target instanceof Button)
    {
      String str = (String)arg;
      if(str.equals("Start"))
      {
        if(operation.getSelectedIndex()==1)         //εκκίνηση του παραθύρου ταξινόμησης,
        {
          ClassificationFrame cf = new ClassificationFrame(title,true,path);
```

```
            }
            else                                    //εκκίνηση του παραθύρου ανανέωσης των βαρών.
            {
               TrainingFrame tf = new TrainingFrame(title,path);
            }
            dispose();
         }
         return true;
      }
      else
         return false;
   }
   public boolean handleEvent(Event vnt)     //διακοπή της λειτουργίας, το παράθυρο
   {                                         //της εφαρμογής κλείνει.
      if(vnt.id==Event.WINDOW_DESTROY)
      {
         System.exit(0);
         return true;
      }
      else
         return super.handleEvent(vnt);
   }
   public static void main(String args[])       //σημείο εκκίνησης της όλης εφαρμογής.
   {
      MainFrame mf = new MainFrame("WIFuNeCS",args[0]);
   }
}

class Poster extends Canvas
{
   Image current;
   Image img[] = new Image[3];

   Poster(String path)        //καθορισμός των εικόνων που μπορούν να εμφανιστούν.
   {
      super();
      img[0] = Toolkit.getDefaultToolkit().getImage(path+"Machine.gif");
      img[1] = Toolkit.getDefaultToolkit().getImage(path+"Detestus.jpg");
      img[2] = Toolkit.getDefaultToolkit().getImage(path+"Plug.jpg");
   }
   void setPoster(int selection)        //επιλογή εικόνας προς εμφάνιση.
   {
      current = img[selection%img.length];
      repaint();
   }
   public void paint(Graphics g)      //εμφάνιση εικόνας.
   {
      if(current!=null)
         g.drawImage(current,0,0,this);
   }
}




//********************
// ClassificationFrame.java
//********************
import java.awt.*;
import java.awt.event.*;
```

```
import java.io.*;

class ClassificationFrame extends Frame      //δημιουργία του παραθύρου της ταξινόμησης.
{
   Font f = new Font("TimesRoman",Font.BOLD,12);
   String title;              //υπέρτιτλος του παραθύρου της εφαρμογής.
   boolean real;              //μεταβλητή που ελέγχει την λειτουργία συλλογής δεδομένων
                              //δηλαδή το εάν όλα τα δεδομένα αποστέλλονται από έναν αισθητήρα ή από
                              //διαφορετικούς.
   String path;               //η διαδρομή του φακέλου που περιέχει τα αρχεία της εφαρμογής.
   Choice typeOfData;         //λίστα επιλογής προβλήματος ταξινόμησης.
   int parameters[][];        //πίνακας με τις παραμέτρους κάθε προβλήματος.
   Choice ruleSet;            //λίστα επιλογής απλού ή πολυεπίπεδου ταξινομητή.
   Button reset;
   TextField category;        //πεδίο κειμένου στο οποίο εμφανίζεται το αποτέλεσμα της ταξινόμησης.
   Movie film;                //αντικείμενο για την δημιουργία κινούμενης εικόνας.
   TextField aSupfunis,bSupfunis,cSupfunis;          //τα αποτελέσματα των ταξινομητών.
   TextField numOfPatterns;                  //ο αριθμός των δεδομένων που έχουν συλλεγεί.
   TextField sensors;         //αριθμός αισθητήρων.
   Button quit;
   Weights heavy;             //αντικείμενο με τις παραμέτρους του συστήματος.
   int input;                 //ο αριθμός των χαρακτηριστικών κάθε δείγματος.
   Result rslt;               //αντικείμενο που χειρίζεται τα συλλεγόμενα πρότυπα εισόδου.
   Classification supfunis = new Classification();      //αντικείμενο που επιτελεί την ταξινόμηση.
   String xc[]=new String[100];              //το δείγμα αναπαριστώμενο από πίνακα.
   static Server tmp[]=new Server[100];      //αντικείμενο για να δέχεται το PDA δεδομένα από
                                             //διαφορετικούς αισθητήρες.
   static ServerII tmpII;                    //αντικείμενο για να δέχεται το PDA όλα τα δεδομένα από
                                             //τον ίδιο αισθητήρα.

   //δημιουργία του αρχικού παραθύρου της εφαρμογής χρησιμοποιώντας
   //την διάταξη δομημένων πλεγμάτων.
   ClassificationFrame(String title,boolean real,String path)
   {
      super(title+"/-/Classification");
      this.title = title;
      this.real = real;
      this.path = path;
      GridBagLayout gbl = new GridBagLayout();
      GridBagConstraints gbc = new GridBagConstraints();
      setLayout(gbl);
      Label l;
      l = new Label("Type of Data::");
      Constraints.addConstraints(gbl,l,gbc,0,0,1,1,0,0,GridBagConstraints.WEST);
      add(l);
      typeOfData = new Choice();    //λίστα επιλογής προβλήματος ταξινόμησης.
      typeOfData.setFont(f);
      typeOfData.add(" ");
      Constraints.addConstraints(gbl,typeOfData,gbc,1,0,2,1,0,0,GridBagConstraints.WEST);
      add(typeOfData);
      try                        //ανάκτηση των δεδομένων που αφορούν στα προβλήματα ταξινόμησης
      {                          //τα οποία έχουν φορτωθεί στο PDA.
         FileReader fr;
         BufferedReader br;
         String str;
         int end;
         fr = new FileReader(path+"info.txt");          //όλες οι σχετικές πληροφορίες βρίσκονται στο
                                                        //αρχείο info.txt.
         br = new BufferedReader(fr);
         while((str = br.readLine())!=null)
```

```
    {
       typeOfData.add(str.substring(0,str.indexOf(' ')));
    }
    br.close();
    fr = new FileReader(path+"info.txt");
    br = new BufferedReader(fr);
    parameters = new int[5][typeOfData.getItemCount()-1];
    for(int j = 0;j<parameters[0].length;j++)
    {
       str = br.readLine();
       int i;
       for(i = 0;i<5;i++)
       {
          end = str.indexOf(' ');
          if(i!=0)
             parameters[i-1][j] = Integer.parseInt(str.substring(0,end));
          str = str.substring(end+1,str.length());
       }
       parameters[i-1][j] = Integer.parseInt(str);
    }
    br.close();
}
catch(IOException e)    {}
l = new Label("Rule Set::");
Constraints.addConstraints(gbl,l,gbc,0,1,1,1,0,0,GridBagConstraints.WEST);
add(l);
ruleSet = new Choice();          //λίστα επιλογής απλού ή πολυεπίπεδου ταξινομητή.
ruleSet.setFont(f);
ruleSet.add("");
ruleSet.add("Single");
ruleSet.add("Triple");
Constraints.addConstraints(gbl,ruleSet,gbc,1,1,2,1,0,0,GridBagConstraints.WEST);
add(ruleSet);
ruleSet.setEnabled(false);
reset = new Button("Reset");
Constraints.addConstraints(gbl,reset,gbc,2,1,1,1,0,0,GridBagConstraints.CENTER);
add(reset);
reset.setEnabled(false);
film = new Movie(true,path);  //δημιουργία αντικειμένου για κινούμενα γραφικά.
gbc.fill = GridBagConstraints.BOTH;
Constraints.addConstraints(gbl,film,gbc,0,2,3,1,0,1,GridBagConstraints.NORTHWEST);
add(film);
gbc.fill = GridBagConstraints.NONE;
l = new Label("Result::");
Color c = new Color(205,208,209);
l.setBackground(c);
Constraints.addConstraints(gbl,l,gbc,0,3,1,1,0,0,GridBagConstraints.WEST);
add(l);
category = new TextField(2);  //καθορισμός του πεδίου που αφορά στα αποτελέσματα.
category.setFont(f);
category.setBackground(c);
category.setEditable(false);
Constraints.addConstraints(gbl,category,gbc,1,3,2,1,0,0,GridBagConstraints.WEST);
add(category);
l = new Label("SuPFuNIS::");
c = new Color(231,235,236);
l.setBackground(c);
Constraints.addConstraints(gbl,l,gbc,0,4,1,1,0,0,GridBagConstraints.WEST);
add(l);
aSupfunis = new TextField(16);          //καθορισμός του πεδίου που αφορά το πρώτο τμήμα
```

```
    aSupfunis.setBackground(c);              //του πολυεπίπεδου ταξινομητή.
    aSupfunis.setFont(f);
    aSupfunis.setEditable(false);
    Constraints.addConstraints(gbl,aSupfunis,gbc,1,4,2,1,0,0,GridBagConstraints.WEST);
    add(aSupfunis);
    bSupfunis = new TextField(16);
    bSupfunis.setBackground(c);  //καθορισμός του πεδίου που αφορά το δεύτερο τμήμα
    bSupfunis.setFont(f);            //του πολυεπίπεδου ταξινομητή.
    bSupfunis.setEditable(false);
    Constraints.addConstraints(gbl,bSupfunis,gbc,1,5,2,1,0,0,GridBagConstraints.WEST);
    add(bSupfunis);
    cSupfunis = new TextField(16);           //καθορισμός του πεδίου που αφορά το τρίτο τμήμα
    cSupfunis.setBackground(c);              //του πολυεπίπεδου ταξινομητή.
    cSupfunis.setFont(f);
    cSupfunis.setEditable(false);
    Constraints.addConstraints(gbl,cSupfunis,gbc,1,6,2,1,0,0,GridBagConstraints.WEST);
    add(cSupfunis);
    l = new Label("Patterns::");
    Constraints.addConstraints(gbl,l,gbc,0,7,1,1,0,0,GridBagConstraints.WEST);
    add(l);
    numOfPatterns = new TextField(7);      //καθορισμός του πεδίου που απεικονίζει τον αριθμό
    numOfPatterns.setFont(f);              //των δειγμάτων που έχουν συλλεγεί.
    numOfPatterns.setBackground(Color.white);
    numOfPatterns.setEditable(false);
    Constraints.addConstraints(gbl,numOfPatterns,gbc,1,7,2,1,0,0,GridBagConstraints.WEST);
    add(numOfPatterns);
    l = new Label("Sensors::");
    Constraints.addConstraints(gbl,l,gbc,0,8,1,1,0,0,GridBagConstraints.WEST);
    add(l);
    sensors = new TextField(2);              //καθορισμός του πεδίου που απεικονίζει τον αριθμό
    sensors.setFont(f);                      //των αισθητήρων.
    sensors.setBackground(Color.white);
    sensors.setEditable(false);
    Constraints.addConstraints(gbl,sensors,gbc,1,8,1,1,1,0,GridBagConstraints.WEST);
    add(sensors);
    quit = new Button("Quit");
    Constraints.addConstraints(gbl,quit,gbc,2,8,1,1,0,0,GridBagConstraints.CENTER);
    add(quit);
    setSize(239,293);
    setResizable(false);
    setZero();
    show();
}
void setZero()                       //αρχικοποίηση όλων των πεδίων κειμένου.
{
    category.setText("0");
    aSupfunis.setText(" ");
    bSupfunis.setText(" ");
    cSupfunis.setText(" ");
    numOfPatterns.setText("0");
    sensors.setText("0");
}
public boolean action(Event vnt,Object arg)          //χειρισμός συμβάντων που προκαλούνται από
{                                                    //τον χρήστη.
    if(vnt.target instanceof Button)
    {
       String str = (String)arg;
       if(str.equals("Quit"))          //διακοπή κάθε λειτουργίας και επιστροφή στο αρχικό παράθυρο.
       {
         System.out.println("Supfunis OFF, Sensors OFF");
```

```
            film.stop();
            supfunis.stopClassifing();
            if(!real)
            {
               stop_multi_sensor();
             }
            else
            {
               stop_simple_sensor();
            }
            MainFrame mf = new MainFrame(title,path);
            dispose();
         }
         else if(str.equals("Reset"))   //διακοπή των λειτουργιών της ταξινόμησης
         {                              //και της συλλογής δειγμάτων.
            System.out.println("Supfunis OFF, Sensors OFF,Screen CLEARED");
            film.stop();
            supfunis.stopClassifing();
            if(!real)
               stop_multi_sensor();
            else
               stop_simple_sensor();
            setZero();
            ruleSet.select(0);
            ruleSet.setEnabled(false);
            reset.setEnabled(false);
            typeOfData.setEnabled(true);
         }
         return true;
      }
      else if(vnt.target instanceof Choice)
      {
         String str = (String)arg;
         if(str.equals("")||str.equals("Single")||str.equals("Triple"))
         {
            String rs = ruleSet.getSelectedItem();
            if(rs.equals(""))                   //διακοπή της λειτουργίας της ταξινόμησης.
            {
               System.out.println("Supfunis OFF");
               film.stop();
               supfunis.stopClassifing();
               typeOfData.setEnabled(false);
               ruleSet.setEnabled(true);
               reset.setEnabled(true);
            }
            else if(rs.equals("Single"))        //εκκίνηση της λειτουργίας της κατηγοριοποίησης
            {                                   //χρησιμοποιώντας τον απλό ταξινομητή.
               System.out.println("Supfunis OFF, Supfunis(1) ON");
               film.stop();
               supfunis.stopClassifing();
               supfunis = new Classification();
               film.start();
               supfunis.startClassifing(true,this,heavy);
               typeOfData.setEnabled(false);
               ruleSet.setEnabled(true);
               reset.setEnabled(true);
            }
            else                                //εκκίνηση της λειτουργίας της κατηγοριοποίησης
            {                                   //χρησιμοποιώντας τον πολυεπίπεδο ταξινομητή.
               System.out.println("Supfunis OFF, Supfunis(3) ON");
```

```
                film.stop();
                supfunis.stopClassifing();
                supfunis = new Classification();
                film.start();
                supfunis.startClassifing(false,this,heavy);
                typeOfData.setEnabled(false);
                ruleSet.setEnabled(true);
                reset.setEnabled(true);
            }
        }
        else
        {
            String tod = typeOfData.getSelectedItem();
            if(tod.equals(" "))
            {
                ruleSet.setEnabled(false);
                reset.setEnabled(false);
                typeOfData.setEnabled(true);
            }
            else if(!tod.equals(" "))    //εκκίνηση της λειτουργίας λήψης δεδομένων
            {                            //από τους αισθητήρες.
                System.out.println("Initialization, Sensors ON");
                int index = typeOfData.getSelectedIndex()-1;
                heavy = new Weights(path,typeOfData.getSelectedItem(),parameters[0][index],
                    parameters[1][index],parameters[2][index],parameters[3][index],parameters[4][index]);
                input = parameters[0][index];
                if(!real)
                {
                    start_multi_sensor();
                    sensors.setText("1");
                }
                else
                {
                    start_simple_sensor();
                    sensors.setText(String.valueOf(input));
                }
                typeOfData.setEnabled(false);
                ruleSet.setEnabled(true);
                reset.setEnabled(true);
            }
        }
        return true;
    }
    else
        return false;
}
public boolean handleEvent(Event vnt)    //διακοπή της λειτουργίας, το παράθυρο
{                                        //της εφαρμογής κλείνει.
    if(vnt.id==Event.WINDOW_DESTROY)
    {
        System.exit(0);
        return true;
    }
    else
        return super.handleEvent(vnt);
}
public void stop_multi_sensor()      //παύση της διεργασίας λήψης δεδομένων από έναν αισθητήρα.
{
        if(tmpII!=null)
        {
```

```
                tmpII.requestStop();
        }
}
public void start_multi_sensor()      //εκκίνηση της διαδικασίας λήψης δεδομένων από έναν αισθητήρα.
{
        if(tmpII==null)               //εάν είναι η πρώτη εκκίνηση.
        {
                tmpII=new ServerII("0",path+typeOfData.getSelectedItem(),input,numOfPatterns);
                tmpII.start();
        }
        else                          //κάθε άλλη φορά.
        {
                tmpII.echo.check=true;
                tmpII.m_chk=true;
                tmpII.features=numOfPatterns;
                tmpII.inp=input;
                tmpII.item=path+typeOfData.getSelectedItem();
                tmpII.xc=new String[input+1];
                for(int j=0;j<tmpII.xc.length;j++)
                {
                        tmpII.xc[j]="0";
                }
                tmpII.tmp_access=new Result(tmpII.xc,numOfPatterns);
        }
}
public void stop_simple_sensor()    //παύση της διεργασίας λήψης δεδομένων από πολλούς αισθητήρες
{
        for(int k=1;k<=input;k++)
        {
        if(tmp[k]!=null)
        {
                tmp[k].requestStop();
        }
        }
}
public void start_simple_sensor()   //εκκίνηση της διαδικασίας λήψης δεδομένων
{                                    //από πολλούς αισθητήρες.
        if(tmp[1]==null)  //εάν είναι η πρώτη εκκίνηση.
        {
                rslt = new Result(input,numOfPatterns);
                for(int k=1;k<=input;k++)
                {
                tmp[k]=new Server(String.valueOf(k),path+typeOfData.getSelectedItem(),
                                        input,numOfPatterns,rslt);
                tmp[k].start();
                }
        }
        else                //κάθε άλλη φορά εκτός της πρώτης.
        {
                rslt = new Result(input,numOfPatterns);
                for(int k=1;k<=input;k++)
                {
                        tmp[k].echo.check=true;
                        tmp[k].m_chk=true;
                        tmp[k].features=numOfPatterns;
                        tmp[k].inp=input;
                        tmp[k].item=path+typeOfData.getSelectedItem();
                        tmp[k].xc=new String[input+1];
                        for(int j=0;j<tmp[k].xc.length;j++)
                        {
```

```
                                    tmp[k].xc[j]="0";
                                }
                                tmp[k].tmp_access=rslt;
                            }
                        }
}
}



//********** *****
// Classification.java
//****************
class Classification implements Runnable    //κλάση που εμπεριέχει το σύνολο της λειτουργικότητας
{                                           //του ταξινομητή.
   Thread runner;                   //για τον χειρισμό του νήματος.
   boolean single;                  //ορίζεται το εάν θα χρησιμοποιηθεί ο απλός ή
                                    //ο πολυεπίπεδος ταξινομητής.
   ClassificationFrame pipe;        //αναφορά στο καλών αντικείμενο με σκοπό να
                                    //χρησιμοποιηθούν μεταβλητές ομοτύπου που έχουν
                                    //οριστεί σε αυτό.
   Weights arg;                     //αντικείμενο που περιέχει τις παραμέτρους του ταξινομητή.

   Classification()
   {
      super();
   }
   //εύρεση των δυο μεγαλύτερων στοιχείων ενός πίνακα, στην συνέχεια επιστρέφονται οι τιμές των
   //δεικτών των εν λόγω στοιχείων.
   int[] getMaxIndex(double pin[])
   {
      int value[] = new int[2];

      if(pin[0]>pin[1])             //αρχικοποίηση.
      {
         value[0] = 0;
         value[1] = 1;
      }
      else
      {
         value[0] = 1;
         value[1] = 0;
      }
      for(int i = 2;i<pin.length;i++)          //διατρέχονται όλα τα στοιχεία.
      {
         if(pin[i]>pin[value[0]])
         {
            value[1] = value[0];
            value[0] = i;
         }
         else if(pin[i]>pin[value[1]])
            value[1] = i;
      }
      return value;
   }
   //σύγκριση δυο πινάκων στοιχείο προς στοιχείο.
   boolean isEqual(double a[],double b[])
   {
      boolean value;
```

```
    value = true;
    for(int i = 0;i<a.length;i++)
        if(a[i]!=b[i])
        {
            value = false;
            break;
        }
    return value;
}
//εκκίνηση διεργασίας ταξινόμησης.
void startClassifing(boolean single,ClassificationFrame pipe,Weights arg)
{
    if(runner==null)
    {
        this.single = single;
        this.pipe = pipe;
        this.arg = arg;
        runner = new Thread(this);
        runner.start();
    }
}
//παύση διεργασίας ταξινόμησης.
void stopClassifing()
{
    runner = null;
}
//διεργασία που εσωκλείει την λειτουργία της ταξινόμησης.
public void run()
{
    Thread thisTread = Thread.currentThread();
    Result output;          //αντικείμενο μέσω του οποίου λαμβάνεται το πιο πρόσφατο δείγμα.
    double previous[];      //προηγούμενο πρότυπο εισόδου.
    double present[];       //παρόν πρότυπο εισόδου.
    double result[];        //σήμα στρώματος εξόδου.
    int win[];              //πίνακας με τους δείκτες των κόμβων εξόδου με τις μεγαλύτερες τιμές.
    double reliability;     //τιμή κριτηρίου αξιοπιστίας.
    boolean firstTime;
    double view;            //τα πέντε σημαντικά ψηφία του μέτρου αξιοπιστίας.
    double confidence[] = new double[3];   //πίνακας τιμών αξιοπιστίας.
    int partialWin[] = new int[3];         //πίνακας των αποτελεσμάτων ταξινόμησης των
                                           //στοιχειωδών ταξινομητών.
    if(!pipe.real)          //τα δεδομένα λαμβάνονται από πολλούς αισθητήρες.
        output = pipe.tmpII.tmp_access;
    else                    //τα δεδομένα λαμβάνονται από έναν αισθητήρα.
        output = pipe.rslt;
    firstTime = true;
    previous = output.access(null,null);
    if(single)              //απλός ταξινομητής.
    {
        while(runner==thisTread)
        {
            present = output.access(null,null);          //τιμή του νέου δείγματος.
            if(firstTime||!isEqual(previous,present))    //έλεγχος εάν διαφέρει από το προηγούμενο.
            {
                firstTime = false;
                previous = present;
                result = Supfunis.testing(present,arg.axs,arg.awc,arg.aws,  //σήμα εξόδου του νευρο-.
                                arg.avc,arg.avs);                           //ασαφούς συστήματος.
                win = getMaxIndex(result);                          //winner takes all.
```

```
            reliability = (result[win[0]]-result[win[1]])*result[win[0]]; //υπολογισμός μέτρου
                                                                          //αξιοπιστίας.
            //εμφάνιση αποτελεσμάτων στο παράθυρο της εφαρμογής του PDA.
            pipe.category.setText(String.valueOf(win[0]+1));
            view = ((double)((int)(reliability*10000.0+0.5)))/10000.0;
            pipe.aSupfunis.setText(String.valueOf(win[0]+1)+"  {"+
                                   String.valueOf(arg.awc[0].length)+"}  "+
                                   String.valueOf(view));
            pipe.bSupfunis.setText(" ");
            pipe.cSupfunis.setText(" ");
         }
         pause(250);                  //καθυστέρηση 250 msec.
      }
   }
   else           //πολυεπίπεδος ταξινομητής.
   {
      while(runner==thisTread)
      {
         //σε κάθε στοιχειώδες νευρο-ασαφές σύστημα του πολυεπίπεδου ταξινομητή
         //ακολουθείται η ίδια διαδικασία που ακολουθείται και στον απλό ταξινομητή.
         present = output.access(null,null);
         if(firstTime||!isEqual(previous,present))
         {
            firstTime = false;
            previous = present;
            result = Supfunis.testing(present,arg.axs,arg.awc,arg.aws,
                                      arg.avc,arg.avs);
            win = getMaxIndex(result);
            reliability = (result[win[0]]-result[win[1]])*result[win[0]];
            partialWin[0] = win[0];           //αποθήκευση της αξιοπιστίας στον αντίστοιχο πίνακα.
            confidence[0] = reliability;
            view = ((double)((int)(reliability*10000.0+0.5)))/10000.0;
            pipe.aSupfunis.setText(String.valueOf(win[0]+1)+"  {"+
                                   String.valueOf(arg.awc[0].length)+"}  "+
                                   String.valueOf(view));
            result = Supfunis.testing(present,arg.bxs,arg.bwc,arg.bws,
                                      arg.bvc,arg.bvs);
            win = getMaxIndex(result);
            reliability = (result[win[0]]-result[win[1]])*result[win[0]];
            partialWin[1] = win[0];           //αποθήκευση της αξιοπιστίας στον αντίστοιχο πίνακα.
            confidence[1] = reliability;
            view = ((double)((int)(reliability*10000.0+0.5)))/10000.0;
            pipe.bSupfunis.setText(String.valueOf(win[0]+1)+"  {"+
                                   String.valueOf(arg.bwc[0].length)+"}  "+
                                   String.valueOf(view));
            result = Supfunis.testing(present,arg.cxs,arg.cwc,arg.cws,
                                      arg.cvc,arg.cvs);
            win = getMaxIndex(result);
            reliability = (result[win[0]]-result[win[1]])*result[win[0]];
            partialWin[2] = win[0];           //αποθήκευση της αξιοπιστίας στον αντίστοιχο πίνακα.
            confidence[2] = reliability;
            view = ((double)((int)(reliability*10000.0+0.5)))/10000.0;
            pipe.cSupfunis.setText(String.valueOf(win[0]+1)+"  {"+
                                   String.valueOf(arg.cwc[0].length)+"}  "+
                                   String.valueOf(view));
            if((partialWin[0]==partialWin[1])&&(partialWin[1]==partialWin[2]))   //πλειοψηφία.
               pipe.category.setText(String.valueOf(partialWin[0]+1));
            else if((partialWin[0]==partialWin[1]))                              //πλειοψηφία.
               pipe.category.setText(String.valueOf(partialWin[0]+1));
            else if((partialWin[1]==partialWin[2]))                              //πλειοψηφία.
```

```
                pipe.category.setText(String.valueOf(partialWin[1]+1));
            else if((partialWin[0]==partialWin[2]))                    //πλειοψηφία.
                pipe.category.setText(String.valueOf(partialWin[2]+1));
            else                                    //επιλέγεται το αποτέλεσμα του ταξινομητή με το
            {                                       //μεγαλύτερο μέτρο αξιοπιστίας.
                win = getMaxIndex(confidence);
                pipe.category.setText(String.valueOf(partialWin[win[0]]+1));
            }
          }
          pause(250);
        }
      }
   }
   //συνάρτηση που προκαλεί καθυστέρηση για τον συγκεκριμένο αριθμό msec.
   public void pause(int msec)
   {
      try
      {
         Thread.sleep(msec);
      }
      catch(InterruptedException e)  {}
   }
}




//***** ****
// Result.java
//**********
import java.io.*;
import java.awt.*;

class Result                    //κλάση μέσω της οποίας γίνεται ο χειρισμός των συλλεγομένων δειγμάτων.
{
   double xc[];
   TextField patterns;
   int count;                   //μετρητής του πλήθους αποθηκευμένων προτύπων εισόδου.

   Result(String features[],TextField patterns)          //αρχικοποιήσεις επιμέρους μεταβλητών.
   {
      count = 0;
      this.patterns = patterns;
      xc = new double[features.length-1];
      for(int i = 0;i<xc.length;i++)
         xc[i] = Double.parseDouble(features[i+1]);
   }
   Result(int features,TextField patterns)                //αρχικοποιήσεις επιμέρους μεταβλητών.
   {
      count = 0;
      this.patterns = patterns;
      xc = new double[features];
      for(int i = 0;i<xc.length;i++)
         xc[i] = 0.0;
   }
   //΄σύγκριση δυο πινάκων στοιχείο προς στοιχείο.
   synchronized boolean isEqual(double a[],double b[])
   {
      boolean value;
```

```
        value = true;
        for(int i = 0;i<a.length;i++)
           if(a[i]!=b[i])
           {
              value = false;
              break;
           }
        return value;
     }
//συνάρτηση που χρησιμοποιείται για την αποθήκευση ή την ανάκτηση δειγμάτων εισόδου,
   //αυτές οι δυο λειτουργίες ελέγχονται μέσω της τιμής της μεταβλητής features[].
   synchronized double[] access(String features[],String name)
   {
      double value[] = null;

      if(features==null)      //ανάκτηση του πιο πρόσφατου δείγματος.
      {
         value = new double[xc.length];
         for(int i = 0;i<xc.length;i++)
            value[i] = xc[i];
      }
      else                    //αποθήκευση ενός καινούργιου δείγματος το οποίο
      {                       //συλλέχθηκε από αισθητήρα.
         value = new double[xc.length];
         for(int i = 0;i<xc.length;i++)
            value[i] = Double.parseDouble(features[i+1]);
         if(!isEqual(xc,value))        //έλεγχος για το εάν διαφέρει από το προηγούμενο.
         {
            for(int i = 0;i<xc.length;i++)
            {
               xc[i] = value[i];
            }
            count++;                                    //αύξηση του μετρητή.
            patterns.setText(String.valueOf(count));
            try
            {
               FileOutputStream fos = new FileOutputStream(name+".dat",true);   //αποθήκευση των
               BufferedOutputStream bos = new BufferedOutputStream(fos);        //χαρακτηριστικών
               DataOutputStream dos = new DataOutputStream(bos);                //του δείγματος
                                                                                //στο αρχείο
               for(int i = 0;i<xc.length;i++)                                   //[name].dat.
                  dos.writeDouble(xc[i]);
               dos.close();
            }
            catch(IOException e)    {}
         }
         value = null;
      }
      return value;
   }
}



//***** ******
// Weights.java
//************
import java.io.*;
```

```
class Weights      //κλάση που περιέχει τις παραμέτρους του όλου συστήματος.
{
   double axs[],bxs[],cxs[];                    //τα διανύσματα βαρών του απλού και των
   double awc[][],bwc[][],cwc[][];              //στοιχειωδών νευρο-ασαφών συστημάτων
   double aws[][],bws[][],cws[][];              //αποθηκεύονται στους αντίστοιχους double
   double avc[][],bvc[][],cvc[][];              //πίνακες.
   double avs[][],bvs[][],cvs[][];

   //ανάκτηση των παραμέτρων του συστήματος και αποθήκευσή τους στους παραπάνω πίνακες.
   Weights(String path,String name,int inputs,int aRules,int bRules,int cRules,int outputs)
   {
      super();
      axs = transfer(path+name,".axs",inputs);
      bxs = transfer(path+name,".bxs",inputs);
      cxs = transfer(path+name,".cxs",inputs);
      awc = transfer(path+name,".awc",inputs,aRules);
      bwc = transfer(path+name,".bwc",inputs,bRules);
      cwc = transfer(path+name,".cwc",inputs,cRules);
      aws = transfer(path+name,".aws",inputs,aRules);
      bws = transfer(path+name,".bws",inputs,bRules);
      cws = transfer(path+name,".cws",inputs,cRules);
      avc = transfer(path+name,".avc",aRules,outputs);
      bvc = transfer(path+name,".bvc",bRules,outputs);
      cvc = transfer(path+name,".cvc",cRules,outputs);
      avs = transfer(path+name,".avs",aRules,outputs);
      bvs = transfer(path+name,".bvs",bRules,outputs);
      cvs = transfer(path+name,".cvs",cRules,outputs);
   }
   //επιστρέφει double πίνακα με τους αριθμούς που βρίσκονται στο byte αρχείο.
   double[] transfer(String name,String extension,int elements)
   {
      double value[] = new double[elements];

      try
      {
         FileInputStream fis = new FileInputStream(name+extension);
         BufferedInputStream bis = new BufferedInputStream(fis);
         DataInputStream dis = new DataInputStream(bis);

         for(int i = 0;i<elements;i++)
         {
            value[i] = dis.readDouble();
         }
         dis.close();
      }
      catch(IOException e)    {}
      return value;
   }
   //επιστρέφει δισδιάστατο double πίνακα με τους αριθμούς που βρίσκονται στο byte αρχείο.
   double[][] transfer(String name,String extension,int rows,int columns)
   {
      double value[][] = new double[rows][columns];

      try
      {
         FileInputStream fis = new FileInputStream(name+extension);
         BufferedInputStream bis = new BufferedInputStream(fis);
         DataInputStream dis = new DataInputStream(bis);

         for(int i = 0;i<rows*columns;i++)
```

```
            {
               value[i/columns][i%columns] = dis.readDouble();
            }
            dis.close();
         }
         catch(IOException e)    {}
         return value;
      }
}



//***** ****
// Movie.java
//**********
import java.awt.*;

class Movie extends Canvas implements Runnable     //δημιουργία κινούμενης εικόνας.
{
   Image img[] = new Image[9];      //καρέ της κινούμενης εικόνας.
   int index;                       //δείκτης προς τα καρέ της εικόνας.
   Thread runner;                   //για τον χειρισμό του νήματος.

   Movie(boolean clockwise,String path)
   {
      super();
      if(clockwise)            //δεξιόστροφη κίνηση της εικόνας.
      {
         for(int i = 0;i<img.length;i++)
            img[i] = Toolkit.getDefaultToolkit().getImage(path+"machine"+
                                    String.valueOf(i)+".gif");
      }
      else                     //αριστερόστροφη κίνηση της εικόνας.
      {
         for(int i = 0;i<img.length;i++)
            img[8-i] = Toolkit.getDefaultToolkit().getImage(path+"machine"+
                                    String.valueOf(i)+".gif");
      }
      index = 5;
   }
   void start()      //εκκίνηση κινούμενης εικόνας.
   {
      if(runner==null)
      {
         runner = new Thread(this);
         runner.start();
      }
   }
   void stop()       //σταμάτημα κινούμενης εικόνας.
   {
      runner = null;
   }
   public void run()
   {
      Thread thisThread = Thread.currentThread();

      while(runner==thisThread)     //η κίνηση της εικόνας επιτυγχάνεται εναλλάσσοντας
      {                             //τον δείκτη προς τα καρέ της εικόνας με συγκεκριμένο τρόπο
         index++;                   //αλλά και κατά συγκεκριμένα χρονικά διαστήματα.
```

```
            if(index>=img.length)
               index = 0;
            repaint();
            try
            {
               Thread.sleep(33);
            }
            catch(InterruptedException e)   {}
         }
      }
      public void update(Graphics g)    //υπέρβαση της update() για να αποφευχθεί το flickering.
      {
         paint(g);
      }
      public void paint(Graphics g)      //απεικόνιση του επιλεγμένου καρέ.
      {
         if(img[index]!=null)
            g.drawImage(img[index],0,0,this);
      }
}



//***** ***********
// TrainingFrame.java
//****************
import java.awt.*;
import java.awt.event.*;
import java.io.*;

class TrainingFrame extends Frame//το παράθυρο της εφαρμογής που σχετίζεται με την
{                                //αρχικοποίηση-ανανέωση των παραμέτρων του συστήματος.
   Font f = new Font("TimesRoman",Font.BOLD,12);
   String title;              //ο υπέρτιτλος του παραθύρου.
   String path;               //η διαδρομή του φακέλου στον οποία βρίσκονται τα αρχεία της εφαρμογής.
   Choice typeOfData;         //λίστα επιλογής προβλήματος ταξινόμησης.
   Choice operation;          //λίστα επιλογής ανανέωσης ή αρχικοποίησης των διανυσμάτων βαρών.
   TextField ip;
   Movie film;
   TextField status;          //πεδίο κειμένου που εμφανίζονται πληροφορίες σχετικές με την πρόοδο
                              //της επιτελούμενης λειτουργίας.
   Button quit;
   client_initial cl = new client_initial();              //αντικείμενο που δέχεται τις παραμέτρους του
                                                          //συστήματος για ένα καινούργιο πρόβλημα.
   client_checking cc = new client_checking();            //αντικείμενο που δέχεται τις ανανεωμένες
                                                          //παραμέτρους του συστήματος, ενώ παράλληλα
                                                          //αποστέλλει τα δείγματα που έχει συλλέξει.
   //δημιουργία του αρχικού παραθύρου της εφαρμογής χρησιμοποιώντας
   //την διάταξη δομημένων πλεγμάτων.
   TrainingFrame(String title,String path)
   {
      super(title+"/-/Training");
      this.title = title;
      this.path = path;
      GridBagLayout gbl = new GridBagLayout();
      GridBagConstraints gbc = new GridBagConstraints();
      setLayout(gbl);
      Label l;
      l = new Label("Data Set::");
```

```
Constraints.addConstraints(gbl,l,gbc,0,0,1,1,0,0,GridBagConstraints.WEST);
add(l);
typeOfData = new Choice();    //δημιουργία της λίστας επιλογής προβλήματος ταξινόμησης.
typeOfData.setFont(f);
typeOfData.add(" ");
Constraints.addConstraints(gbl,typeOfData,gbc,1,0,1,1,0,0,GridBagConstraints.WEST);
add(typeOfData);
try
{
   FileReader fr;
   BufferedReader br;
   String str;

   fr = new FileReader(path+"info.txt");
   br = new BufferedReader(fr);
   while((str = br.readLine())!=null)
   {
      typeOfData.add(str.substring(0,str.indexOf(' ')));
   }
   br.close();
}
catch(IOException e)    {}
l = new Label("Operation::");
Constraints.addConstraints(gbl,l,gbc,0,1,1,1,0,0,GridBagConstraints.WEST);
add(l);
operation = new Choice();       //δημιουργία της λίστας επιλογής λειτουργίας.
operation.setFont(f);
operation.add("");
operation.add("Update");
operation.add("Initialize");
Constraints.addConstraints(gbl,operation,gbc,1,1,1,1,0,0,GridBagConstraints.WEST);
add(operation);
l = new Label("IP Address::");
Constraints.addConstraints(gbl,l,gbc,0,2,1,1,0,0,GridBagConstraints.WEST);
add(l);
ip = new TextField(11);         //καθορισμός του πεδίου που αφορά στη IP διεύθυνση
ip.setFont(f);                  //όπου βρίσκεται ο server για την αντίστοιχη λειτουργία.
Constraints.addConstraints(gbl,ip,gbc,1,2,1,1,0,0,GridBagConstraints.WEST);
add(ip);
film = new Movie(false,path);             //δημιουργία αντικειμένου για κινούμενη εικόνα.
gbc.fill = GridBagConstraints.BOTH;
Constraints.addConstraints(gbl,film,gbc,0,3,3,1,0,1,GridBagConstraints.NORTHWEST);
add(film);
gbc.fill = GridBagConstraints.NONE;
l = new Label("Status::");
Constraints.addConstraints(gbl,l,gbc,0,4,1,1,0,0,GridBagConstraints.WEST);
add(l);
status = new TextField(18);     //καθορισμός πεδίου κειμένου που εμφανίζονται πληροφορίες
status.setFont(f);              //σχετικές με την πρόοδο της επιτελούμενης λειτουργίας.
status.setBackground(Color.white);
status.setEditable(false);
Constraints.addConstraints(gbl,status,gbc,1,4,2,1,0,0,GridBagConstraints.WEST);
add(status);
l = new Label("");
Constraints.addConstraints(gbl,l,gbc,0,5,1,1,0,0,GridBagConstraints.WEST);
add(l);
l = new Label("");
Constraints.addConstraints(gbl,l,gbc,0,6,1,1,0,0,GridBagConstraints.WEST);
add(l);
quit = new Button("Quit");
```

```
    Constraints.addConstraints(gbl,quit,gbc,2,7,1,1,1,0,GridBagConstraints.CENTER);
    add(quit);
    setSize(239,293);
    setResizable(false);
    show();
    addWindowListener                //διακοπή της λειτουργίας, το παράθυρο της εφαρμογής κλείνει.
    (
        new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        }
    );
}
//ελέγχεται το εάν το αλφαριθμητικό αναπαριστά έγκυρη IP διεύθυνση, δηλαδή το εάν είναι ένας
//αριθμός μεταξύ 0.0.0.0 και 255.255.255.255.
boolean isValidIP(String str)
{
    int dot1,dot2,dot3;

    if(str.equals("localhost"))
        return true;
    if(str.length()>15)
        return false;
    dot1 = str.indexOf('.');
    if(dot1==-1)
        return false;
    dot2 = str.indexOf('.',(dot1+1));
    if(dot2==-1)
        return false;
    dot3 = str.indexOf('.',(dot2+1));
    if(dot3==-1)
        return false;
    String temp;
    temp = str.substring(0,dot1);
    try
    {
    if((Integer.parseInt(temp)<0)||(Integer.parseInt(temp)>255))
        return false;
    }
    catch(NumberFormatException e)
    {
        return false;
    }
    temp = str.substring(dot1+1,dot2);
    try
    {
    if((Integer.parseInt(temp)<0)||(Integer.parseInt(temp)>255))
        return false;
    }
    catch(NumberFormatException e)
    {
        return false;
    }
    temp = str.substring(dot2+1,dot3);
    try
    {
    if((Integer.parseInt(temp)<0)||(Integer.parseInt(temp)>255))
```

```
            return false;
        }
        catch(NumberFormatException e)
        {
            return false;
        }
        temp = str.substring(dot3+1,str.length());
        try
        {
        if((Integer.parseInt(temp)<0)||(Integer.parseInt(temp)>255))
            return false;
        }
        catch(NumberFormatException e)
        {
            return false;
        }
        return true;
    }
    public boolean action(Event vnt,Object arg)          //χειρισμός συμβάντων που προκαλούνται από
    {                                                    //τον χρήστη.
        if(vnt.target instanceof Button)          //διακοπή κάθε λειτουργίας.
        {
            film.stop();
            cl.stop();
            cc.stop();
            System.out.println("STOP ALL");
            String str = (String)arg;
            if(str.equals("Quit"))
            {
                MainFrame mf = new MainFrame(title,path);
                dispose();
            }
            return true;
        }
        else if(vnt.target instanceof Choice)
        {
            film.stop();                          //αρχικά διακόπτεται κάθε λειτουργία.
            cl.stop();
            cc.stop();
            System.out.println("STOP ALL");
            String tod,op;
            tod = typeOfData.getSelectedItem();
            op = operation.getSelectedItem();
            //εμφανίζονται κατάλληλα μηνύματα στο πεδίο κατάστασης αναλόγως του εάν η λειτουργία
            //ζητάται παρέχεται ή είναι λανθασμένη.
            if(op.equals(""))
            {
                status.setText(" ");
            }
            else if(tod.equals(" ")&&op.equals("Update"))
            {
                status.setText("Data Set Not Selected");
            }
            else if((!tod.equals(" "))&&op.equals("Initialize"))
            {
                status.setText("Already Initialized");
            }
            else if(tod.equals(" ")&&op.equals("Initialize"))
            {
                if(!isValidIP(ip.getText()))
```

```
            {
               status.setText("Invalid IP Address");
            }
            else                                          //έναρξη διεργασίας προσθήκης στο PDA
            {                                             //ενός καινούργιου προβλήματος ταξινόμησης.
               film.start();
               status.setText("Connecting ...");
               cl = new client_initial();
               cl.path=path;
               cl.port_number = 6000;
               cl.features = status;
                    cl.ip_adress = ip.getText();
               cl.start();
               System.out.println("Initialization");
            }
         }
         else if((!tod.equals(" "))&&op.equals("Update"))
         {
            if(!isValidIP(ip.getText()))
            {
               status.setText("Invalid IP Address");
            }
            else                                          //έναρξη διεργασίας ανανέωσης των
            {                                             //παραμέτρων του ταξινομητή επί ενός
               film.start();                              //ήδη γνωστού προβλήματος ταξινόμησης.
               status.setText("Connecting ...");
                    client_checking cc = new client_checking();
                    cc.fil = path+typeOfData.getSelectedItem();
                    cc.port_number = 6000;
                    cc.features = status;
                    cc.ip_adress = ip.getText();
                    cc.start();
               System.out.println("Updating");
            }
         }
         return true;
      }
      else
         return false;
   }
}
```

```
//**********
// Inintial.java
//**********
import java.io.*;
import javax.swing.*;
import javax.swing.text.*;
import java.util.Random;

//κλάση που ενθυλακώνει όλη την απαραίτητη λειτουργικότητα που απαιτείται για την αρχικοποίηση
//του συστήματος (από την πλευρά του server) αναφορικά με ένα συγκεκριμένο πρόβλημα
//ταξινόμησης, ο αλγόριθμος που χρησιμοποιείται είναι ο backpropagation με την μέθοδο κλίσης.
class Initial implements Runnable
{
   String name;    //το αναγνωριστικό του προβλήματος ταξινόμησης.
   int inputs;     //ο αριθμός των χαρακτηριστικών κάθε προτύπου εισόδου.
```

```
int aRules;      //ο βέλτιστος αριθμός κανόνων, που χρησιμοποιείται τόσο στον απλό όσο
                 //και στον πολυεπίπεδο ταξινομητή.
int bRules;      //αριθμός κανόνων μικρότερος του βέλτιστου.
int cRules;      //αριθμός κανόνων μεγαλύτερος του βέλτιστου.
int outputs;     //αριθμός των κόμβων εξόδου.
int patterns;    //ο αριθμός των προτύπων που περιέχονται στο σύνολο εκπαίδευσης.
int epochs;              //ο αριθμός των εποχών εκπαίδευσης.
double accuracy;         //η τιμή της επιθυμητής ακρίβειας υποκατάστασης.
double training[][];     //πίνακας στον οποίο είναι αποθηκευμένα τα πρότυπα του συνόλου
                         //εκπαίδευσης, καθώς και η κατηγορία που ανήκει το καθένα από αυτά.
double d[][];            //πίνακας με τις κατηγορίες των προτύπων του συνόλου
                         //εκπαίδευσης σε κωδικοποίηση 1 - out of - K.
double axs[],paxs[];             //πίνακες που περιέχουν όλες τις παραμέτρους του
double awc[][],pawc[][];         //ταξινομητή (τόσο του απλού όσο και του πολυεπίπεδου)
double aws[][],paws[][];         //κατά τα t-οστό και (t-1)-οστό βήματα εκτέλεσης του
double avc[][],pavc[][];         //αλγορίθμου εκπαίδευσης.
double avs[][],pavs[][];
double bxs[],pbxs[];
double bwc[][],pbwc[][];
double bws[][],pbws[][];
double bvc[][],pbvc[][];
double bvs[][],pbvs[][];
double cxs[],pcxs[];
double cwc[][],pcwc[][];
double cws[][],pcws[][];
double cvc[][],pcvc[][];
double cvs[][],pcvs[][];
Thread runner;           //μεταβλητή για τον έλεγχο των νημάτων.
JTextField write;        //αντικείμενο για την εμφάνιση αποτελεσμάτων στην οθόνη της εφαρμογής.
JButton btn;             //μεταβλητές για τον χειρισμό των κουμπιών
JButton btnSwitch;       //του παραθύρου της εφαρμογής.

//μέθοδος εγκατάστασης μέσω της οποίας ορίζονται οι αρχικές παράμετροι του προβλήματος
//ταξινόμησης, ενώ παράλληλα αρχικοποιούνται οι παράμετροι του αλγορίθμου εκπαίδευσης.
Initial(String name,int inputs,String file,int aRules,int epochs,double accuracy,
        JTextField arg,JButton btn,JButton btnSwitch)
{
   write = arg;
   this.btn = btn;
   this.btnSwitch=btnSwitch;
   this.name = name;
   this.inputs = inputs;
   this.aRules = aRules;                    //βέλτιστος αριθμός κανόνων.
   this.epochs = epochs;
   this.accuracy = accuracy/100.0;          //ποσοστό επιθυμητής ακρίβειας υποκατάστασης.
   bRules = (int)(aRules*0.6+0.5);          //προσδιορισμός του υποβέλτιστου πλήθους κανόνων.
   cRules = (int)(aRules*1.4+0.5);          //υπολογισμός του συνόλου κανόνων με πλήθος
                                            //μεγαλύτερο του βέλτιστου.
   IOFiles.fileNormalForm(file,name+".res");       //ανάκτηση των προτύπων εκπαίδευσης
                                                   //και αποθήκευση τους στο byte
                                                   //αρχείο [name].res.
   //υπολογισμός του πλήθους των προτύπων εκπαίδευσης
   //και αποθήκευση τους στους αντίστοιχους πίνακες.
   int count = 0;
   try
   {
      FileInputStream fis = new FileInputStream(name+".res");
      BufferedInputStream bis = new BufferedInputStream(fis);
      DataInputStream dis = new DataInputStream(bis);
```

```
            count = 0;
            try
            {
               while(true)
               {
                  dis.readDouble();
                  count++;
               }
            }
            catch(EOFException e)   {}
            dis.close();
         }
         catch(IOException e)    {}
         patterns = count/(inputs+1);
         training = IOFiles.fileToArray(name+".res",patterns,(inputs+1));
         outputs = (int)getMaxValue(training,training[0].length-1);
         d = new double[training.length][outputs];
         for(int i = 0;i<d.length;i++)                  //κατασκευή του κωδικοποιημένου πίνακα κατηγοριών.
            for(int j = 0;j<d[0].length;j++)
            {
               if(j==(training[i][inputs]-1))
                  d[i][j] = 1.0;
               else
                  d[i][j] = 0.0;
            }
         createSpecNfo();                 //κατασκευή του αρχείου βασικών παραμέτρων spec.nfo.
         constructNewWeights();           //αρχικοποίηση των διανυσμάτων βαρών του συστήματος.
         storeWeights();                  //αποθήκευση των παραμέτρων
      }
      //αποθήκευση στο αρχείο spec.nfo των βασικών παραμέτρων του προβλήματος ταξινόμησης.
      void createSpecNfo()
      {
         String str;

         try
         {
            FileWriter fw = new FileWriter("spec.nfo");
            BufferedWriter bw = new BufferedWriter(fw);

            try
            {
               str = name;                            //αναγνωριστικό προβλήματος.
               bw.write(str,0,str.length());
               bw.newLine();
               str = String.valueOf(inputs);          //πλήθος των χαρακτηριστικών κάθε δείγματος.
               bw.write(str,0,str.length());
               bw.newLine();
               //πλήθος των κανόνων που χρησιμοποιεί ο κάθε στοιχειώδης ταξινομητής.
               str = String.valueOf(aRules);
               bw.write(str,0,str.length());
               bw.newLine();
               str = String.valueOf(bRules);
               bw.write(str,0,str.length());
               bw.newLine();
               str = String.valueOf(cRules);
               bw.write(str,0,str.length());
               bw.newLine();
               str = String.valueOf(outputs);         //πλήθος των κόμβων εξόδου.
               bw.write(str,0,str.length());
               bw.newLine();
```

```
            str = String.valueOf(patterns);       //πλήθος των προτύπων εκπαίδευσης.
            bw.write(str,0,str.length());
            bw.newLine();
         }
         catch(EOFException e)   {}
         bw.close();
      }
      catch(IOException e)    {}
   }
   //επιστρέφει την μέγιστη τιμή του στοιχείου της συγκεκριμένης στήλης του δισδιάστατου πίνακα.
   double getMaxValue(double pin[][],int column)
   {
      double value;

      value = pin[0][column];
      for(int i = 1;i<pin.length;i++)
         if(pin[i][column]>value)
            value = pin[i][column];
      return value;
   }
   //επιστρέφει την ελάχιστη τιμή του στοιχείου της συγκεκριμένης στήλης του δισδιάστατου πίνακα.
   double getMinValue(double pin[][],int column)
   {
      double value;

      value = pin[0][column];
      for(int i = 1;i<pin.length;i++)
         if(pin[i][column]<value)
            value = pin[i][column];
      return value;
   }
   //δημιουργία πίνακα με τιμές στοιχείων ομοιόμορφα κατανεμημένες στο διάστημα [min,max].
   double[] createRandomArray(int dim,double min,double max)
   {
      Random rand = new Random();
      double value[] = new double[dim];

      for(int i = 0;i<value.length;i++)
         value[i] = min+(max-min)*rand.nextDouble();
      return value;
   }
   //δημιουργία δισδιάστατου πίνακα με τιμές στοιχείων
   //ομοιόμορφα κατανεμημένες στο διάστημα [min,max].
   double[][] createRandomArray(int rows,int columns,double min,double max)
   {
      Random rand = new Random();
      double value[][] = new double[rows][columns];

      for(int i = 0;i<value.length;i++)
         for(int j = 0;j<value[0].length;j++)
            value[i][j] = min+(max-min)*rand.nextDouble();
      return value;
   }
   //δημιουργία δισδιάστατου πίνακα με τιμές στοιχείων των οποίων οι τιμές ανά γραμμή
   //είναι ομοιόμορφα κατανεμημένες στο διάστημα [min[x],max[x]], όπου x είναι η γραμμή.
   double[][] createRandomArray(int rows,int columns,double min[],double max[])
   {
      Random rand = new Random();
      double value[][] = new double[rows][columns];
```

```
    for(int i = 0;i<value.length;i++)
        for(int j = 0;j<value[0].length;j++)
            value[i][j] = min[i]+(max[i]-min[i])*rand.nextDouble();
    return value;
}
//δημιουργία των πινάκων που περιέχουν τις αρχικές τιμές (τυχαίες) των παραμέτρων
//του νευρο-ασαφούς συστήματος.
void constructNewWeights()
{
    axs = createRandomArray(inputs,0.2,0.9);
    paxs = createRandomArray(inputs,0.2,0.9);
    bxs = createRandomArray(inputs,0.2,0.9);
    pbxs = createRandomArray(inputs,0.2,0.9);
    cxs = createRandomArray(inputs,0.2,0.9);
    pcxs = createRandomArray(inputs,0.2,0.9);
    double max[] = new double[inputs];
    double min[] = new double[inputs];
    for(int i = 0;i<inputs;i++)
    {
        max[i] = getMaxValue(training,i);
        min[i] = getMinValue(training,i);
    }
    awc = createRandomArray(inputs,aRules,min,max);
    pawc = createRandomArray(inputs,aRules,min,max);
    bwc = createRandomArray(inputs,bRules,min,max);
    pbwc = createRandomArray(inputs,bRules,min,max);
    cwc = createRandomArray(inputs,cRules,min,max);
    pcwc = createRandomArray(inputs,cRules,min,max);
    aws = createRandomArray(inputs,aRules,0.2,0.9);
    paws = createRandomArray(inputs,aRules,0.2,0.9);
    bws = createRandomArray(inputs,bRules,0.2,0.9);
    pbws = createRandomArray(inputs,bRules,0.2,0.9);
    cws = createRandomArray(inputs,cRules,0.2,0.9);
    pcws = createRandomArray(inputs,cRules,0.2,0.9);
    avc = createRandomArray(aRules,outputs,0.0,1.0);
    pavc = createRandomArray(aRules,outputs,0.0,1.0);
    bvc = createRandomArray(bRules,outputs,0.0,1.0);
    pbvc = createRandomArray(bRules,outputs,0.0,1.0);
    cvc = createRandomArray(cRules,outputs,0.0,1.0);
    pcvc = createRandomArray(cRules,outputs,0.0,1.0);
    avs = createRandomArray(aRules,outputs,0.2,0.9);
    pavs = createRandomArray(aRules,outputs,0.2,0.9);
    bvs = createRandomArray(bRules,outputs,0.2,0.9);
    pbvs = createRandomArray(bRules,outputs,0.2,0.9);
    cvs = createRandomArray(cRules,outputs,0.2,0.9);
    pcvs = createRandomArray(cRules,outputs,0.2,0.9);
}
//αποθήκευση των παραμέτρων του συστήματος στα αντίστοιχα byte αρχεία.
void storeWeights()
{
    IOFiles.arrayToFile(axs,name+".axs");    IOFiles.arrayToFile(paxs,"p"+name+".axs");
    IOFiles.arrayToFile(awc,name+".awc");    IOFiles.arrayToFile(pawc,"p"+name+".awc");
    IOFiles.arrayToFile(aws,name+".aws");    IOFiles.arrayToFile(paws,"p"+name+".aws");
    IOFiles.arrayToFile(avc,name+".avc");    IOFiles.arrayToFile(pavc,"p"+name+".avc");
    IOFiles.arrayToFile(avs,name+".avs");    IOFiles.arrayToFile(pavs,"p"+name+".avs");
    IOFiles.arrayToFile(bxs,name+".bxs");    IOFiles.arrayToFile(pbxs,"p"+name+".bxs");
    IOFiles.arrayToFile(bwc,name+".bwc");    IOFiles.arrayToFile(pbwc,"p"+name+".bwc");
    IOFiles.arrayToFile(bws,name+".bws");    IOFiles.arrayToFile(pbws,"p"+name+".bws");
    IOFiles.arrayToFile(bvc,name+".bvc");    IOFiles.arrayToFile(pbvc,"p"+name+".bvc");
    IOFiles.arrayToFile(bvs,name+".bvs");    IOFiles.arrayToFile(pbvs,"p"+name+".bvs");
```

```
    IOFiles.arrayToFile(cxs,name+".cxs");    IOFiles.arrayToFile(pcxs,"p"+name+".cxs");
    IOFiles.arrayToFile(cwc,name+".cwc");    IOFiles.arrayToFile(pcwc,"p"+name+".cwc");
    IOFiles.arrayToFile(cws,name+".cws");    IOFiles.arrayToFile(pcws,"p"+name+".cws");
    IOFiles.arrayToFile(cvc,name+".cvc");    IOFiles.arrayToFile(pcvc,"p"+name+".cvc");
    IOFiles.arrayToFile(cvs,name+".cvs");    IOFiles.arrayToFile(pcvs,"p"+name+".cvs");
}
//εκκίνηση της διεργασίας εκπαίδευσης.
void start(int rules,int epochs,double accuracy)
{
    if(runner==null)
    {
        if(rules!=aRules)              //πιθανός επαναπροσδιορισμός του αριθμού των
        {                              //κανόνων του συστήματος.
            aRules = rules;
            bRules = (int)(aRules*0.6+0.5);
            cRules = (int)(aRules*1.4+0.5);
            createSpecNfo();
            constructNewWeights();
            storeWeights();
        }
        this.epochs = epochs;                  //καθορισμός του αριθμού των εποχών εκπαίδευσης.
        this.accuracy = accuracy/100.0;        //καθορισμός του ποσοστού ακρίβειας υποκατάστασης.
        runner = new Thread(this);
        runner.start();
    }
}
//διακοπή της διαδικασίας εκπαίδευσης.
void stop()
{
    runner = null;
}
public void run()
{
    Thread thisThread = Thread.currentThread();
    Learning learn = new Learning();                    //αντικείμενο μέσω του οποίου εκτελείται
                                                        //ο αλγόριθμος backpropagation.
    double xc[] = new double[inputs];
    int errors;                                         //αριθμός των σφαλμάτων υποκατάστασης..
    boolean stop;
    double h,a,stepH,stepA;                             //learning rate & momentum.

    stepH = stepA = (0.1-0.0001)/(double)(epochs-1);          //προσδιορισμός των βημάτων μείωσης.
    h = a = 0.1;
    stop = false;
    for(int loop = 0;loop<epochs;loop++)    //εκτέλεση του backpropagation για τον
    {                                       //συγκεκριμένο αριθμό εποχών.
        if(runner!=thisThread||stop)
            break;
        for(int index = 0;index<training.length;index++)
        {
            Supfunis.copy(training[index],xc);
            //εκτέλεση ενός βήματος του αλγορίθμου backpropagation.
            learn.gradientDescent(h,a,d[index],xc,axs,awc,aws,avc,avs,
                                  paxs,pawc,paws,pavc,pavs);
            errors = Supfunis.countResubstitutions(training,axs,awc,aws,avc,avs); //υπολογισμός
                                                                                  //σφαλμάτων
                                                                                  //υποκατάστασης.
            System.out.print(errors+" ");
            write.setText(String.valueOf(errors)+"/"+String.valueOf(patterns));    //εμφάνιση του
                                                                                   //αποτελέσματος.
```

```
      if(((double)(patterns-errors)/(double)patterns)>=accuracy)     //εάν ικανοποιηθεί το ποσοστό
         stop = true;                                                //ακρίβειας υποκατάστασης ο
                                                                     //αλγόριθμος εκπαίδευσης
                                                                     //διακόπτεται.
      if(runner!=thisThread||stop)
         break;
   }
   h -= stepH;          //γραμμική μείωση του ρυθμού μάθησης.
   a -= stepA;          //γραμμική μείωση του όρου ορμής.
}
System.out.println("\n");
//επανάληψη του αλγορίθμου εκπαίδευσης για το δεύτερο σύνολο κανόνων, δηλαδή με αριθμό
//κανόνων μικρότερο του βέλτιστου.
h = a = 0.1;
stop = false;
for(int loop = 0;loop<epochs;loop++)
{
   if(runner!=thisThread||stop)
      break;
   for(int index = 0;index<training.length;index++)
   {
      Supfunis.copy(training[index],xc);
      learn.gradientDescent(h,a,d[index],xc,bxs,bwc,bws,bvc,bvs,
                  pbxs,pbwc,pbws,pbvc,pbvs);
      errors = Supfunis.countResubstitutions(training,bxs,bwc,bws,bvc,bvs);
      System.out.print(errors+" ");
      write.setText(String.valueOf(errors)+"/"+String.valueOf(patterns));
      if(((double)(patterns-errors)/(double)patterns)>=accuracy)
         stop = true;
      if(runner!=thisThread||stop)
         break;
   }
   h -= stepH;
   a -= stepA;
}
System.out.println("\n");
//επανάληψη του αλγορίθμου εκπαίδευσης για το τρίτο σύνολο κανόνων, δηλαδή με αριθμό
//κανόνων μεγαλύτερο του βέλτιστου.
h = a = 0.1;
stop = false;
for(int loop = 0;loop<epochs;loop++)
{
   if(runner!=thisThread||stop)
      break;
   for(int index = 0;index<training.length;index++)
   {
      Supfunis.copy(training[index],xc);
      learn.gradientDescent(h,a,d[index],xc,cxs,cwc,cws,cvc,cvs,
                  pcxs,pcwc,pcws,pcvc,pcvs);
      errors = Supfunis.countResubstitutions(training,cxs,cwc,cws,cvc,cvs);
      System.out.print(errors+" ");
      write.setText(String.valueOf(errors)+"/"+String.valueOf(patterns));
      if(((double)(patterns-errors)/(double)patterns)>=accuracy)
         stop = true;
      if(runner!=thisThread||stop)
         break;
   }
   h -= stepH;
   a -= stepA;
}
```

```
        storeWeights();
        btn.setText("Start");
        btnSwitch.setEnabled(true);
        runner = null;
    }
/*
    public static void main(String args[])
    {
        Initial init = new Initial("ionosphere",34,"ionosphere.txt",10,100,80,new JTextField());
        init.start(12,500,95);
    }
*/
}
```

```
//**********
// Update.java
//**********
import java.io.*;

//κλάση που ενθυλακώνει την απαραίτητη λειτουργικότητα που απαιτείται για την επανεκπαίδευση
//του συστήματος (από την πλευρά του server) αναφορικά με ένα συγκεκριμένο πρόβλημα
//ταξινόμησης, ο αλγόριθμος που χρησιμοποιείται είναι αυτός της υβριδικής μάθησης.
class Update implements Runnable
{
    String name;    //το αναγνωριστικό του προβλήματος ταξινόμησης.
    int inputs;     //ο αριθμός των χαρακτηριστικών κάθε προτύπου εισόδου.
    int aRules;     //ο βέλτιστος αριθμός κανόνων, που χρησιμοποιείται τόσο στον απλό όσο
                    //και στον πολυεπίπεδο ταξινομητή.
    int bRules;     //αριθμός κανόνων μικρότερος του βέλτιστου.
    int cRules;     //αριθμός κανόνων μεγαλύτερος του βέλτιστου.
    int outputs;    //αριθμός των κόμβων εξόδου.
    int patterns;   //ο αριθμός των προτύπων που περιέχονται στο σύνολο εκπαίδευσης.
    double axs[],paxs[];            //πίνακες που περιέχουν όλες τις παραμέτρους του
    double awc[][],pawc[][];        //ταξινομητή (τόσο του απλού όσο και του πολυεπίπεδου)
    double aws[][],paws[][];        //κατά τα t-οστό και (t-1)-οστό βήματα εκτέλεσης του
    double avc[][],pavc[][];        //αλγορίθμου επανεκπαίδευσης.
    double avs[][],pavs[][];
    double bxs[],pbxs[];
    double bwc[][],pbwc[][];
    double bws[][],pbws[][];
    double bvc[][],pbvc[][];
    double bvs[][],pbvs[][];
    double cxs[],pcxs[];
    double cwc[][],pcwc[][];
    double cws[][],pcws[][];
    double cvc[][],pcvc[][];
    double cvs[][],pcvs[][];
    double data[][];        //πίνακας στον οποίο είναι αποθηκευμένα τα πρότυπα του συνόλου
                            //εκπαίδευσης, καθώς και η κατηγορία που ανήκει το καθένα.
    double received[][];    //πίνακας στον οποίο είναι αποθηκευμένα
                            //τα πρότυπα επανεκπαίδευσης.
    Thread runner;          //μεταβλητές για τον έλεγχο των νημάτων, της διεργασίας.
    Thread thisThread;

    //μέθοδος εγκατάστασης μέσω της οποίας ανακτώνται οι βασικές παράμετροι του προβλήματος
    //όπως επίσης ανακτώνται και οι παράμετροι του αλγορίθμου επανεκπαίδευσης.
    Update()
```

```
{
   try                              //ανακτώνται βασικές παράμετροι του προβλήματος ταξινόμησης
   {                                //από το αρχείο spec.nfo.
      FileReader fr;
      BufferedReader br;
      String str;

      fr = new FileReader("spec.nfo");
      br = new BufferedReader(fr);                  //αναγνωριστικό του προβλήματος.
      str = br.readLine();
      name = str;                                   //πλήθος των χαρακτηριστικών κάθε δείγματος.
      str = br.readLine();
      inputs = Integer.parseInt(str);
      str = br.readLine();
      //πλήθος των κανόνων που χρησιμοποιεί ο κάθε στοιχειώδης ταξινομητής.
      aRules = Integer.parseInt(str);
      str = br.readLine();
      bRules = Integer.parseInt(str);
      str = br.readLine();
      cRules = Integer.parseInt(str);
      str = br.readLine();
      outputs = Integer.parseInt(str);          //πλήθος των κόμβων εξόδου.
      str = br.readLine();
      patterns = Integer.parseInt(str);         //το πλήθος των προτύπων βάσει των οποίων
                                                //έγινε η αρχική εκπαίδευση του συστήματος.
      br.close();
      FileWriter fw = new FileWriter(name+".dat");//δημιουργία ενός κενού [name].dat αρχείου.
      fw.close();
   }
   catch(IOException e)    {}
   //ανάκτηση των απαραίτητων παραμέτρων του αλγορίθμου επανεκπαίδευσης
   //από τα αντίστοιχα byte αρχεία.
   axs = IOFiles.fileToArray(name+".axs",inputs);
   paxs = IOFiles.fileToArray("p"+name+".axs",inputs);
   awc = IOFiles.fileToArray(name+".awc",inputs,aRules);
   pawc = IOFiles.fileToArray("p"+name+".awc",inputs,aRules);
   aws = IOFiles.fileToArray(name+".aws",inputs,aRules);
   paws = IOFiles.fileToArray("p"+name+".aws",inputs,aRules);
   avc = IOFiles.fileToArray(name+".avc",aRules,outputs);
   pavc = IOFiles.fileToArray("p"+name+".avc",aRules,outputs);
   avs = IOFiles.fileToArray(name+".avs",aRules,outputs);
   pavs = IOFiles.fileToArray("p"+name+".avs",aRules,outputs);
   bxs = IOFiles.fileToArray(name+".bxs",inputs);
   pbxs = IOFiles.fileToArray("p"+name+".bxs",inputs);
   bwc = IOFiles.fileToArray(name+".bwc",inputs,bRules);
   pbwc = IOFiles.fileToArray("p"+name+".bwc",inputs,bRules);
   bws = IOFiles.fileToArray(name+".bws",inputs,bRules);
   pbws = IOFiles.fileToArray("p"+name+".bws",inputs,bRules);
   bvc = IOFiles.fileToArray(name+".bvc",bRules,outputs);
   pbvc = IOFiles.fileToArray("p"+name+".bvc",bRules,outputs);
   bvs = IOFiles.fileToArray(name+".bvs",bRules,outputs);
   pbvs = IOFiles.fileToArray("p"+name+".bvs",bRules,outputs);
   cxs = IOFiles.fileToArray(name+".cxs",inputs);
   pcxs = IOFiles.fileToArray("p"+name+".cxs",inputs);
   cwc = IOFiles.fileToArray(name+".cwc",inputs,cRules);
   pcwc = IOFiles.fileToArray("p"+name+".cwc",inputs,cRules);
   cws = IOFiles.fileToArray(name+".cws",inputs,cRules);
   pcws = IOFiles.fileToArray("p"+name+".cws",inputs,cRules);
   cvc = IOFiles.fileToArray(name+".cvc",cRules,outputs);
   pcvc = IOFiles.fileToArray("p"+name+".cvc",cRules,outputs);
```

```
    cvs = IOFiles.fileToArray(name+".cvs",cRules,outputs);
    pcvs = IOFiles.fileToArray("p"+name+".cvs",cRules,outputs);
    data = IOFiles.fileToArray(name+".res",patterns,(inputs+1));        //αποθηκεύονται σε πίνακα τα
                                                                        //πρότυπα βάσει των οποίων
                                                                        //έγινε η εκπαίδευση (αρχική).
}
//εκκίνηση διαδικασίας επανεκπαίδευσης.
void start()
{
    if(runner==null)
    {
        runner = new Thread(this);
        runner.start();
    }
}
//διακοπή διαδικασίας επανεκπαίδευσης.
void stop()
{
    runner = null;
}
public void run()
{
    thisThread = Thread.currentThread();
    boolean repeat;
    boolean status;

    while(runner==thisThread)
    {

        //έλεγχος (ανά 10 sec) για το εάν έχουν αποσταλεί από κάποιο
        //PDA καινούργια πρότυπα επανεκπαίδευσης.
        repeat = true;
        while(repeat)
        {
            if(runner!=thisThread)
                break;
            while(!newPatternsExist())
            {
                if(runner!=thisThread)
                    break;
                try
                {
                    Thread.sleep(10000);
                }
                catch(InterruptedException e)   {}
            }
            if(createReceivedTable())
                repeat = false;
        }
        if(runner!=thisThread)
            break;
        //εκτελείται ο αλγόριθμος επανεκπαίδευσης για το σύνολο με τον βέλτιστο αριθμό κανόνων,
        //μόλις ο αλγόριθμος τελειώσει αποθηκεύονται στα αντίστοιχα byte αρχεία, οι ανανεωμένες
        //παράμετροι του συστήματος για το εν λόγω πλήθος κανόνων.
        status = aUpdate();
        System.out.println("\n");
        IOFiles.arrayToFile(axs,name+".axs");    IOFiles.arrayToFile(paxs,"p"+name+".axs");
        IOFiles.arrayToFile(awc,name+".awc");    IOFiles.arrayToFile(pawc,"p"+name+".awc");
        IOFiles.arrayToFile(aws,name+".aws");    IOFiles.arrayToFile(paws,"p"+name+".aws");
        IOFiles.arrayToFile(avc,name+".avc");    IOFiles.arrayToFile(pavc,"p"+name+".avc");
```

```
      IOFiles.arrayToFile(avs,name+".avs");    IOFiles.arrayToFile(pavs,"p"+name+".avs");
      if(status)
      {
         if(runner!=thisThread)
            break;
         //όταν η επανεκπαίδευση για το προηγούμενο σύνολο κανόνων τερματίζεται ομαλά
         //εκτελείται ο αλγόριθμος επανεκπαίδευσης για το σύνολο με τον υποβέλτιστο αριθμό
         //κανόνων, μόλις ο αλγόριθμος τελειώσει αποθηκεύονται στα αντίστοιχα byte αρχεία οι
         //ανανεωμένες παράμετροι του συστήματος για το εν λόγω πλήθος κανόνων.
         status = bUpdate();
         System.out.println("\n");
         IOFiles.arrayToFile(bxs,name+".bxs");    IOFiles.arrayToFile(pbxs,"p"+name+".bxs");
         IOFiles.arrayToFile(bwc,name+".bwc");    IOFiles.arrayToFile(pbwc,"p"+name+".bwc");
         IOFiles.arrayToFile(bws,name+".bws");    IOFiles.arrayToFile(pbws,"p"+name+".bws");
         IOFiles.arrayToFile(bvc,name+".bvc");    IOFiles.arrayToFile(pbvc,"p"+name+".bvc");
         IOFiles.arrayToFile(bvs,name+".bvs");    IOFiles.arrayToFile(pbvs,"p"+name+".bvs");
         if(status)
         {
            if(runner!=thisThread)
               break;
            //όταν η επανεκπαίδευση για το προηγούμενο σύνολο κανόνων τερματίζεται ομαλά
            //εκτελείται ο αλγόριθμος επανεκπαίδευσης για το σύνολο με τον μεγαλύτερο του
            //βέλτιστου αριθμού κανόνων, μόλις ο αλγόριθμος τελειώσει αποθηκεύονται στα
            //αντίστοιχα byte αρχεία οι ανανεωμένες παράμετροι του συστήματος για το εν λόγω
            //πλήθος κανόνων.
            cUpdate();
            System.out.println("\n");
            IOFiles.arrayToFile(cxs,name+".cxs");    IOFiles.arrayToFile(pcxs,"p"+name+".cxs");
            IOFiles.arrayToFile(cwc,name+".cwc");    IOFiles.arrayToFile(pcwc,"p"+name+".cwc");
            IOFiles.arrayToFile(cws,name+".cws");    IOFiles.arrayToFile(pcws,"p"+name+".cws");
            IOFiles.arrayToFile(cvc,name+".cvc");    IOFiles.arrayToFile(pcvc,"p"+name+".cvc");
            IOFiles.arrayToFile(cvs,name+".cvs");    IOFiles.arrayToFile(pcvs,"p"+name+".cvs");
         }
      }
   }
}
//εκτελείται ο αλγόριθμος υβριδικής μάθησης για το σύνολο με βέλτιστο πλήθος κανόνων,
//επιστρέφεται true μόνο στην περίπτωση κανονικού τερματισμού του.
boolean aUpdate()
{
   //υπολογισμός του αρχικού αριθμού σφαλμάτων υποκατάστασης.
   int limit = Supfunis.countResubstitutions(data,axs,awc,aws,avc,avs);
   Learning learn = new Learning();         //αντικείμενο μέσω του οποίου επιτυγχάνεται
                                            //η εκτέλεση του αλγορίθμου υβριδικής μάθησης.
   double h;                //ρυθμός μάθησης.
   double a;                //όρος ορμής.
   //μεταβλητές πίνακα double για την προσωρινή αποθήκευση των παραμέτρων του ταξινομητή.
   double keepXs[] = new double[axs.length];
   double keepPXs[] = new double[axs.length];
   double keepWc[][] = new double[awc.length][awc[0].length];
   double keepPWc[][] = new double[awc.length][awc[0].length];
   double keepWs[][] = new double[awc.length][awc[0].length];
   double keepPWs[][] = new double[awc.length][awc[0].length];
   double keepVc[][] = new double[avc.length][avc[0].length];
   double keepPVc[][] = new double[avc.length][avc[0].length];
   double keepVs[][] = new double[avc.length][avc[0].length];
   double keepPVs[][] = new double[avc.length][avc[0].length];
   int temp;

   for(int i = 0;i<100;i++)          //η φάση της επανεκπαίδευσης αποτελείται από 100 εποχές.
```

```
    {
      for(int j = 0;j<received.length;j++)    //σε κάθε εποχή παρουσιάζονται όλα τα πρότυπα
      {                                       //επανεκπαίδευσης ακριβώς μια φορά.
        save(axs,keepXs);   save(paxs,keepPXs);    //αποθηκεύονται οι τιμές των βαρών
        save(awc,keepWc);   save(pawc,keepPWc);//αφού ενδεχόμενα να χρειαστεί αναίρεση
        save(aws,keepWs);   save(paws,keepPWs); //της μεταβολής που προξενείται από τον
        save(avc,keepVc);   save(pavc,keepPVc);   //αλγόριθμο επανεκπαίδευσης.
        save(avs,keepVs);   save(pavs,keepPVs);
        h = a = 0.1;
        for(int times = 0;times<4;times++)//σε κάθε πρότυπο γίνονται 4 απόπειρες τροποποίησης
        {                                       //του διανύσματος των βαρών με τα η και α να λαμβάνουν
                                                //διαδοχικά τιμές 0.1, 0.01, 0.001, 0.0001.
          //εκτελείται ένα βήμα του τροποποιημένου αλγορίθμου backpropagation.
          learn.gradientDescent(h,a,null,received[j],axs,awc,aws,avc,avs,
                    paxs,pawc,paws,pavc,pavs);
          temp = Supfunis.countResubstitutions(data,axs,awc,aws,avc,avs);    //υπολογισμός των
          System.out.print(temp+" ");                                        //σφαλμάτων
                                                                             //υποκατάστασης.
          if(limit>temp)          //εάν ο αριθμός σφαλμάτων υποκατάστασης εμφανίσει μείωση
          {                       //τότε οι τροποποιημένες τιμές του διανύσματος βαρών
            limit = temp;         //διατηρούνται, και η επανεκπαίδευση βάσει του συγκεκριμένου
            break;                //προτύπου διακόπτεται.
          }
          h /= 10.0;              //σε αντίθετη περίπτωση μειώνονται εκθετικά οι τιμές των η και α.
          a /= 10.0;
          //ακόμη αναιρούνται οι τροποποιήσεις που έχουν επέλθει στις παραμέτρους του
          //ταξινομητή, οι οποίες αποτυπώνονται στα διανύσματα των βαρών.
          save(keepXs,axs);   save(keepPXs,paxs);
          save(keepWc,awc);   save(keepPWc,pawc);
          save(keepWs,aws);   save(keepPWs,paws);
          save(keepVc,avc);   save(keepPVc,pavc);
          save(keepVs,avs);   save(keepPVs,pavs);
        }
        if(newPatternsExist()||(runner!=thisThread))
          return false;
      }
    }
    return true;
  }
  //όμοια με προηγούμενα εκτελείται ο αλγόριθμος υβριδικής μάθησης, η διαφορά είναι ότι
  //το σύνολο σε αυτή την περίπτωση περιέχει το υποβέλτιστο πλήθος κανόνων.
  boolean bUpdate()
  {
    int limit = Supfunis.countResubstitutions(data,bxs,bwc,bws,bvc,bvs);
    Learning learn = new Learning();
    double h;
    double a;
    double keepXs[] = new double[bxs.length];
    double keepPXs[] = new double[bxs.length];
    double keepWc[][] = new double[bwc.length][bwc[0].length];
    double keepPWc[][] = new double[bwc.length][bwc[0].length];
    double keepWs[][] = new double[bwc.length][bwc[0].length];
    double keepPWs[][] = new double[bwc.length][bwc[0].length];
    double keepVc[][] = new double[bvc.length][bvc[0].length];
    double keepPVc[][] = new double[bvc.length][bvc[0].length];
    double keepVs[][] = new double[bvc.length][bvc[0].length];
    double keepPVs[][] = new double[bvc.length][bvc[0].length];
    int temp;

    for(int i = 0;i<100;i++)
```

```
    {
      for(int j = 0;j<received.length;j++)
      {
        save(bxs,keepXs);   save(pbxs,keepPXs);
        save(bwc,keepWc);   save(pbwc,keepPWc);
        save(bws,keepWs);   save(pbws,keepPWs);
        save(bvc,keepVc);   save(pbvc,keepPVc);
        save(bvs,keepVs);   save(pbvs,keepPVs);
        h = a = 0.1;
        for(int times = 0;times<4;times++)
        {
          learn.gradientDescent(h,a,null,received[j],bxs,bwc,bws,bvc,bvs,
                    pbxs,pbwc,pbws,pbvc,pbvs);
          temp = Supfunis.countResubstitutions(data,bxs,bwc,bws,bvc,bvs);
          System.out.print(temp+" ");
          if(limit>temp)
          {
            limit = temp;
            break;
          }
          h /= 10.0;
          a /= 10.0;
          save(keepXs,bxs);   save(keepPXs,pbxs);
          save(keepWc,bwc);   save(keepPWc,pbwc);
          save(keepWs,bws);   save(keepPWs,pbws);
          save(keepVc,bvc);   save(keepPVc,pbvc);
          save(keepVs,bvs);   save(keepPVs,pbvs);
        }
        if(newPatternsExist()||(runner!=thisThread))
          return false;
      }
    }
    return true;
}
//όμοια με προηγούμενα εκτελείται ο αλγόριθμος υβριδικής μάθησης, η διαφορά είναι ότι
//το σύνολο σε αυτή την περίπτωση περιέχει μεγαλύτερο αριθμό κανόνων από το βέλτιστο
//πλήθος κανόνων.
boolean cUpdate()
{
    int limit = Supfunis.countResubstitutions(data,cxs,cwc,cws,cvc,cvs);
    Learning learn = new Learning();
    double h;
    double a;
    double keepXs[] = new double[cxs.length];
    double keepPXs[] = new double[cxs.length];
    double keepWc[][] = new double[cwc.length][cwc[0].length];
    double keepPWc[][] = new double[cwc.length][cwc[0].length];
    double keepWs[][] = new double[cwc.length][cwc[0].length];
    double keepPWs[][] = new double[cwc.length][cwc[0].length];
    double keepVc[][] = new double[cvc.length][cvc[0].length];
    double keepPVc[][] = new double[cvc.length][cvc[0].length];
    double keepVs[][] = new double[cvc.length][cvc[0].length];
    double keepPVs[][] = new double[cvc.length][cvc[0].length];
    int temp;

    for(int i = 0;i<100;i++)
    {
      for(int j = 0;j<received.length;j++)
      {
        save(cxs,keepXs);   save(pcxs,keepPXs);
```

```
            save(cwc,keepWc);   save(pcwc,keepPWc);
            save(cws,keepWs);   save(pcws,keepPWs);
            save(cvc,keepVc);   save(pcvc,keepPVc);
            save(cvs,keepVs);   save(pcvs,keepPVs);
            h = a = 0.1;
            for(int times = 0;times<4;times++)
            {
               learn.gradientDescent(h,a,null,received[j],cxs,cwc,cws,cvc,cvs,
                        pcxs,pcwc,pcws,pcvc,pcvs);
               temp = Supfunis.countResubstitutions(data,cxs,cwc,cws,cvc,cvs);
               System.out.print(temp+" ");
               if(limit>temp)
               {
                  limit = temp;
                  break;
               }
               h /= 10.0;
               a /= 10.0;
               save(keepXs,cxs);   save(keepPXs,pcxs);
               save(keepWc,cwc);   save(keepPWc,pcwc);
               save(keepWs,cws);   save(keepPWs,pcws);
               save(keepVc,cvc);   save(keepPVc,pcvc);
               save(keepVs,cvs);   save(keepPVs,pcvs);
            }
            if(newPatternsExist()||(runner!=thisThread))
               return false;
         }
      }
      return true;
   }
   //αποθήκευση των στοιχειών ενός πίνακα σε έναν πίνακα ίδιων διαστάσεων.
   void save(double source[],double destin[])
   {
      for(int i = 0;i<source.length;i++)
         destin[i] = source[i];
   }
   //αποθήκευση των στοιχειών ενός δισδιάστατου
   //πίνακα σε έναν πίνακα ίδιων διαστάσεων.
   void save(double source[][],double destin[][])
   {
      for(int i = 0;i<source.length;i++)
         for(int j = 0;j<source[0].length;j++)
            destin[i][j] = source[i][j];
   }
   boolean createReceivedTable()
   {
      File file = new File("new"+name+".dat");
      File arxeio;

      arxeio = new File(name+".dat");          //μετονομάζεται το αρχείο new[name].dat (το οποίο
      arxeio.delete();                         //περιέχει τα καινούργια πρότυπα επανεκπαίδευσης που
      file.renameTo(new File(name+".dat")); //έχουν σταλεί από το PDA) ως [name].dat, παράλληλα
      new File("new"+name+".dat").delete(); //το new[name].dat διαγράφεται.

      int count = 0;
      //ανακτώνται τα πρότυπα επανεκπαίδευσης από το αρχείο [name].dat.
      try
      {
         FileInputStream fis = new FileInputStream(name+".dat");
         BufferedInputStream bis = new BufferedInputStream(fis);
```

```
            DataInputStream dis = new DataInputStream(bis);

            count = 0;
            try
            {
               while(true)
               {
                  dis.readDouble();
                  count++;
               }
            }
            catch(EOFException e)   {}
            dis.close();
         }
         catch(IOException e)    {}
         if(count/inputs>=1)    //εάν περιέχεται έστω και ένα πρότυπο επανεκπαίδευσης τότε
         {                      //επιστρέφεται true οπότε ξεκινάει μια νέα διαδικασία επανεκπαίδευσης.
            received = IOFiles.fileToArray(name+".dat",count/inputs,inputs);
            return true;
         }
         else                   //σε αντίθετη περίπτωση επιστρέφεται false.
            return false;
      }
      //έλεγχος για το εάν υπάρχει αρχείο new[name].dat.
      boolean newPatternsExist()
      {
         File file = new File("new"+name+".dat");

         return file.exists();
      }
/*
      public static void main(String args[])
      {
         Update u = new Update();
         u.start();
      }
*/
}
```

```
//*******
//client_checking.java
//*******
//κλάση υλοποίησης σύνδεσης με εξυπηρετητή για την ενημέρωση των συνόλων των βαρών
//υπάρχουν τρία σύνολα βαρών a,b,c
//κάθε σύνολο βαρών αποτελείται από 5 αρχεία βαρών xs,wc,ws,vc,vs
//συνολικά λοιπόν αναφερόμαστε σε 15 αρχεία
//κάθε αρχείο περιέχει τιμές ανάλογα με τις διαστάσεις του νευρωνικού δικτύου
//οι τιμές είναι σε μορφή byte, χωρίς κενά
import java.net.*;
import java.io.*;
import java.util.Random;
import java.awt.*;
import java.lang.*;

// η κλάση υποστηρίζει νήματα (Runnable)
public class client_checking implements Runnable
{
        //αναγνωριστικό προβλήματος
        public String fil;
        public BufferedOutputStream bos = null;
        public DataOutputStream dos = null;
        //το πεδίο αναγραφής μηνυμάτων του PDA
        public TextField features=new TextField();
        //αριθμός θύρας
        public int port_number;
        public static Thread runner;
        public static Thread  thisThread;
        //IP διεύθυνση
        public static String ip_adress;
        //κενός κατασκευαστής
        public client_checking()
        {
        }
//η συνάρτηση που στέλνει τα δεδομένα που έχει συλλέξει το PDA μέχρι τη στιγμή σύνδεσης με το
//server
   public void connect_send(String ip_adress,int port_number,String file_name)
   {
    Socket client = null;
    try
      {
     boolean repeat;
     boolean repeat2=true;
     while((repeat2==true)&&(runner==thisThread))
     {
         //γίνεται άνοιγμα του αρχείου από το οποίο θα διαβαστούν οι τιμές
         FileInputStream fis = new FileInputStream(file_name);
         BufferedInputStream bis = new BufferedInputStream(fis);
         DataInputStream dis = new DataInputStream(bis);
       repeat = true;
       //γίνεται προσπάθεια σύνδεσης μέχρι να έχουμε επιτυχία
       while(repeat&&(runner==thisThread))
       {
          try
          {
             features.setText("Connecting Server ...");
             client = new Socket(ip_adress,port_number);
             repeat = false;
          }
          catch(IOException e)
```

```
            {
                    features.setText("Server Not Found ...");
        //το PDA περιμένει 1 δευτερόλεπτο πριν ξαναπροσπαθήσει για σύνδεση
                    try
                    {
                            Thread.sleep(1000);
                    }
                    catch(InterruptedException l)
                    {
                            l.printStackTrace();
                    }
            }
        }
          //στο σημείο αυτό έχει επιτευχθεί σύνδεση
          //αμφίδρομη σύνδεση για να έχουμε ανταλλαγή δεδομένων
        try
        {
           bos = new BufferedOutputStream(client.getOutputStream());
           dos = new DataOutputStream(bos);
        }
        catch(IOException e)
        {
        }
        repeat=true;
        features.setText("Sending Patterns ...");
//ξεκινά η μετάδοση των δεδομένων
        while(repeat&&(runner==thisThread))
        {
          String tmp=new String("");
                try
                  {
                   try
                            {
                             tmp=String.valueOf(dis.readDouble());
                             dos.writeDouble(Double.parseDouble(tmp));
                             dos.flush();
                            }
//εξαίρεση που δείχνει ότι τέλειωσε το αρχείο
                        catch(EOFException eof)
                            {
                            repeat2 = false;
                            repeat = false;
                            try
                                {
                                   dos.close();
                                   dis.close();
                                   client.close();
                                }
                            catch(IOException error){}
//μετά το διάβασμα, το αρχείο διαγράφεται
                            new File(fil+".dat").delete();
                            repeat = false;
                            }
                    }
                        catch(IOException e)
                          {
                             try
                               {
                                  dos.close();
                                  dis.close();
```

```
                                client.close();
                            }
                            catch(IOException error)    {}
                            repeat = false;
                        }
                }
            }
            }
//εξαίρεση εάν δεν βρεθεί το αρχείο από το οποίο θα σταλούν τα δεδομένα
             catch(FileNotFoundException e)
             {
                    features.setText("Unavailable Patterns");
             }
             }
//μέθοδος αποδοχής των βαρών από το server
   public void connect_receive(String ip_adress,int port_number,String file_name)
   {
      Socket client = null;
      BufferedInputStream bis = null;
      DataInputStream dis = null;
      boolean repeat=true;
      //προσπάθεια για σύνδεση
         while(repeat&&(runner==thisThread))
         {
            try
            {
               client = new Socket(ip_adress,port_number);
               repeat = false;
            }
//εξαίρεση  για διακοπή της σύνδεσης από το server
            catch(IOException e)
            {
                    features.setText("Disconnected From Server ...");
                    try
                    {
                            Thread.sleep(1000);
                    }
                    catch(InterruptedException l)
                    {
                            l.printStackTrace();
                    }
            }
         }
//στο σημείο αυτό έχουμε σύνδεση
//αμφίδρομη σύνδεση για ανταλλαγή δεδομένων
        try
         {
            bis = new BufferedInputStream(client.getInputStream());
            dis = new DataInputStream(bis);
         }
         catch(IOException e)
         {
         }
          features.setText("Updating Weights ...");
//η μεταβλητή tmp  είναι αυτή που δέχεται τις τιμές από το server
         double tmp=0;
            try
            {
                    //το αρχείο στο οποίο αποθηκεύονται οι τιμές που στέλνονται από το server
                    FileOutputStream fos = new FileOutputStream(file_name);
```

```
                BufferedOutputStream bos = new BufferedOutputStream(fos);
                DataOutputStream dos = new DataOutputStream(bos);
                repeat=true;
                while((repeat==true)&&(runner==thisThread))
                 {
                        try
                        {
                          tmp=dis.readDouble();
                        }
                        catch(EOFException e)
                        {
                           try
                           {
                              dos.close();
                              dis.close();
                              client.close();
                              repeat=false;
                           }
                           catch(IOException error){}
                         }
                        catch(IOException iods)
                          {
                                try
                                {
                                  dos.close();
                                  repeat=false;
                                 }
                                catch(IOException ioe){}
                //εάν έχουμε διακοπή της σύνδεσης τότε διαγράφεται το μέχρι στιγμής αρχείο
                                try
                                {
                                     fos = new FileOutputStream(file_name);
                                     bos = new BufferedOutputStream(fos);
                                     dos = new DataOutputStream(bos);
                                     try
                                        {
                                          dos.close();
                                          dis.close();
                                          client.close();
                                          repeat=false;
                                        }
                                     catch(IOException ioe)
                                        {}
                                      }
                                     catch(FileNotFoundException fnfe)
                                       {}
                                }
                        dos.writeDouble(tmp);
                        dos.flush();
                }
                try
                {
                dos.close();
                }
                catch(IOException error){}
            }
        catch(IOException e)
                {
                }
}
```

```
//η μέθοδος-νήμα η οποία σειριακά εκτελεί τις διαδικασίες αποστολής των δεδομένων και λήψης των
//βαρών
   public void run()
   {
         thisThread=Thread.currentThread();
//η θύρα 6000 χρησιμοποιείται για την μετάδοση του αρχείου με το όνομα του αναγνωριστικού
//του προβλήματος και κατάληξη dat
         connect_send(ip_adress,port_number,fil+".dat");
//οι θύρες 6001-6015 χρησιμοποιούνται για την μετάδοση των αρχείων των βαρών
//οι τιμές προσωρινά αποθηκεύονται με πρόθεμα κατάληξης tmp
         connect_receive(ip_adress,port_number+1,fil+".tmpaxs");
         connect_receive(ip_adress,port_number+2,fil+".tmpawc");
         connect_receive(ip_adress,port_number+3,fil+".tmpaws");
         connect_receive(ip_adress,port_number+4,fil+".tmpavc");
         connect_receive(ip_adress,port_number+5,fil+".tmpavs");

         connect_receive(ip_adress,port_number+6,fil+".tmpbxs");
         connect_receive(ip_adress,port_number+7,fil+".tmpbwc");
         connect_receive(ip_adress,port_number+8,fil+".tmpbws");
         connect_receive(ip_adress,port_number+9,fil+".tmpbvc");
         connect_receive(ip_adress,port_number+10,fil+".tmpbvs");

         connect_receive(ip_adress,port_number+11,fil+".tmpcxs");
         connect_receive(ip_adress,port_number+12,fil+".tmpcwc");
         connect_receive(ip_adress,port_number+13,fil+".tmpcws");
         connect_receive(ip_adress,port_number+14,fil+".tmpcvc");
         connect_receive(ip_adress,port_number+15,fil+".tmpcvs");

   //εξετάζεται αν όλα τα αρχεία που έχουν πρόθεμα κατάληξης tmp  έχουν μέγεθος αρχείου
   //διαφορετικό από 0
         File axs=new File(fil+".tmpaxs");
         File awc=new File(fil+".tmpawc");
         File aws=new File(fil+".tmpaws");
         File avc=new File(fil+".tmpavc");
         File avs=new File(fil+".tmpavs");

         File bxs=new File(fil+".tmpbxs");
         File bwc=new File(fil+".tmpbwc");
         File bws=new File(fil+".tmpbws");
         File bvc=new File(fil+".tmpbvc");
         File bvs=new File(fil+".tmpbvs");

         File cxs=new File(fil+".tmpcxs");
         File cwc=new File(fil+".tmpcwc");
         File cws=new File(fil+".tmpcws");
         File cvc=new File(fil+".tmpcvc");
         File cvs=new File(fil+".tmpcvs");
         //εάν τα αρχεία βαρών με πρόθεμα κατάληξης tmp έχουν μέγεθος διαφορετικό από 0
         //τότε διαγράφονται τα κανονικά αρχεία βαρών
         //και τα αρχεία tmp παίρνουν το κανονικό τους όνομα
         if((axs.length()!=0)&&(awc.length()!=0)&&(aws.length()!=0)&&(avc.length()!=0)&&
         (avs.length()!=0)&&(bxs.length()!=0)&&
         (bwc.length()!=0)&&(bws.length()!=0)&&
         (bvc.length()!=0)&&(bvs.length()!=0)&&(cxs.length()!=0))
         {
               if(new File(fil+".axs").exists())
               new File(fil+".axs").delete();
               if(new File(fil+".awc").exists())
               new File(fil+".awc").delete();
               if(new File(fil+".aws").exists())
```

```
                    new File(fil+".aws").delete();
                    if(new File(fil+".avc").exists())
                    new File(fil+".avc").delete();
                    if(new File(fil+".avs").exists())
                    new File(fil+".avs").delete();
                    axs.renameTo(new File(fil+".axs"));
                    awc.renameTo(new File(fil+".awc"));
                    aws.renameTo(new File(fil+".aws"));
                    avc.renameTo(new File(fil+".avc"));
                    avs.renameTo(new File(fil+".avs"));

                    if(new File(fil+".bxs").exists())
                    new File(fil+".bxs").delete();
                    if(new File(fil+".bwc").exists())
                    new File(fil+".bwc").delete();
                    if(new File(fil+".bws").exists())
                    new File(fil+".bws").delete();
                    if(new File(fil+".bvc").exists())
                    new File(fil+".bvc").delete();
                    if(new File(fil+".bvs").exists())
                    new File(fil+".bvs").delete();
                    bxs.renameTo(new File(fil+".bxs"));
                    bwc.renameTo(new File(fil+".bwc"));
                    bws.renameTo(new File(fil+".bws"));
                    bvc.renameTo(new File(fil+".bvc"));
                    bvs.renameTo(new File(fil+".bvs"));

                    if(new File(fil+".cxs").exists())
                    new File(fil+".cxs").delete();
                    if(new File(fil+".cwc").exists())
                    new File(fil+".cwc").delete();
                    if(new File(fil+".cws").exists())
                    new File(fil+".cws").delete();
                    if(new File(fil+".cvc").exists())
                    new File(fil+".cvc").delete();
                    if(new File(fil+".cvs").exists())
                    new File(fil+".cvs").delete();
                    cxs.renameTo(new File(fil+".cxs"));
                    cwc.renameTo(new File(fil+".cwc"));
                    cws.renameTo(new File(fil+".cws"));
                    cvc.renameTo(new File(fil+".cvc"));
                    cvs.renameTo(new File(fil+".cvs"));
                    //επιτυχής μετάδοση δεδομένων οπότε και το κατάλληλο μήνυμα στο PDA
                    features.setText("Updating Completed");
            }
else
            {//ανεπιτυχής μετάδοση δεδομένων
                    features.setText("Disconnected - Retry");
    }
 }//μέθοδος εκκίνησης
 void start()
 {
        if(runner==null)
        {
        runner=new Thread(this);
        runner.start();
        }
 } //μέθοδος διακοπής
 void stop()
 {
```

```
            runner=null;
    }}


//*******
//client_initial.java
//*******
//κλάση αρχικοποίησης ενός νέου προβλήματος στο PDA
//για να έχουμε αρχικοποίηση πρέπει να συμπεριληφθούν οι απαραίτητες πληροφορίες στο
//αρχείο info.txt και επίσης να αποθηκευθούν τα νέα βάρη
//υπάρχουν τρία σύνολα βαρών a,b,c
//κάθε σύνολο βαρών αποτελείται από 5 αρχεία βαρών xs,wc,ws,vc,vs
//συνολικά λοιπόν αναφερόμαστε σε 15 αρχεία
//κάθε αρχείο περιέχει τιμές ανάλογα με τις διαστάσεις του νευρωνικού δικτύου
//οι τιμές είναι σε μορφή byte, χωρίς κενά
import java.net.*;
import java.io.*;
import java.util.Random;
import java.awt.*;
import java.lang.*;

// η κλάση υποστηρίζει νήματα (Runnable)
public class client_initial implements Runnable
{
            //αναγνωριστικό προβλήματος
            public String fil="";
            public String info="";
            public BufferedOutputStream bos = null;
            public DataOutputStream dos = null;
            //το πεδίο αναγραφής μηνυμάτων του PDA
            public TextField features=new TextField();
            //αριθμός θύρας
            public int port_number;
            public Thread runner;
            public Thread  thisThread;
            public static String ip_adress;
            public static String new_name;
            public static String path="";
            //κατασκευαστής κενός
            public client_initial()
            {}
//η συνάρτηση που λαμβάνει τα βάρη
   public void connect_receive(String ip_adress,int port_number,String file_name)
   {
      Socket client = null;
      BufferedInputStream bis = null;
      DataInputStream dis = null;
      boolean repeat=true;
     //προσπάθεια σύνδεσης  με τον εξυπηρετητή
     while(repeat&&(runner==thisThread))
        {
          try
          {
             client = new Socket(ip_adress,port_number);
             repeat = false;
          }
          catch(IOException e)
          {
             features.setText("Disconnected From Server ...");
          }
```

```
        }
      //στο σημείο αυτό έχει επιτευχθεί σύνδεση
      //αμφίδρομη σύνδεση για ανταλλαγή δεδομένων
      try
        {
          bis = new BufferedInputStream(client.getInputStream());
          dis = new DataInputStream(bis);
        }
        catch(IOException e)
        {
        }
      features.setText("Initializing ...");
//η μεταβλητή tmp λαμβάνει τις τιμές σε μορφή double
      double tmp=0;
        try
          {
          //το αρχείο στο οποίο αποθηκεύονται οι τιμές που στέλνει ο server
                    FileOutputStream fos = new FileOutputStream(file_name);
                    BufferedOutputStream bos = new BufferedOutputStream(fos);
                    DataOutputStream dos = new DataOutputStream(bos);
                    repeat=true;
          //διαδικασία ανάγνωσης από το socket
                    while((repeat==true)&&(runner==thisThread))
                    {
                    try
                       {
                        tmp=dis.readDouble();
                       }
//εξαίρεση που εγείρεται όταν τελειώσει η μετάδοση των αριθμών
                    catch(EOFException e)
                       {
                         try
                           {
                             dos.close();
                             dis.close();
                             client.close();
                             repeat=false;
                           }
                         catch(IOException error){}
                      }
                    //εάν έχουμε διακοπή σύνδεσης τότε πρέπει να αποδεσμευτούν οι πόροι
                                catch(IOException iods)
                                {
                                        try
                                        {
                                        dos.close();
                                        repeat=false;
                                        }
                                        catch(IOException ioe){}
                                         try
                                         {
                                          fos = new FileOutputStream(file_name);
                                          bos = new BufferedOutputStream(fos);
                                         dos = new DataOutputStream(bos);
                                         try
                                           {
                                                dos.close();
                                                dis.close();
                                                client.close();
                                               repeat=false;
```

```
                                                }
                                                catch(IOException ioe)
                                                {}
                                                }
                                                catch(FileNotFoundException fnfe){}
                                        }
                                dos.writeDouble(tmp);
                                dos.flush();
                        }
                                        try
                                        {
                                        dos.close();
                                        }
                catch(IOException error){}
                }
        catch(IOException e)
                {
                }
   }
//μέθοδος που δέχεται τις πληροφορίες από το server, δηλαδή το αρχείο spec.nfo τις οποίες
//αποθηκεύει στο info.txt
public void connect_receive_line(String ip_adress,int port_number,String file_name)
   {
      Socket client = null;
      BufferedInputStream bis = null;
      BufferedReader dis=null;
      boolean repeat=true;
//προσπάθεια για σύνδεση
         while(repeat&&(runner==thisThread))
         {
            try
            {
               client = new Socket(ip_adress,port_number);
               repeat = false;
            }
            catch(IOException e)
            {
                     features.setText("Disconnected From Server ...");
            }
         }
//στο σημείο αυτό έχει γίνει σύνδεση
//αμφίδρομη σύνδεση για ανταλλαγή δεδομένων
         try
         {
            bis = new BufferedInputStream(client.getInputStream());
            dis       = new BufferedReader(new InputStreamReader(bis));
         }
         catch(IOException e)
         {
         }
          features.setText("Initializing ...");
//επαναληπτικός  βρόχος 5 στιγμών για να διαβαστούν οι πληροφορίες
         String tmp="";
           for(int g=0;g<6;g++)
              {
                     if(runner!=thisThread)break;
                      try
                      {
           //για να μην έχει κενό στο τέλος
                      if(g!=5)
```

```
                    {
                    String down = dis.readLine();
                    //η πρώτη πληροφορία που διαβάζει είναι το αναγνωριστικό του προβλήματος
                            if(g==0)new_name = down;
                              tmp=tmp+down+" ";
                              }
                              else
                               tmp=tmp+dis.readLine();
                               }
                             catch(IOException iods)
                               {
                                    try
                                         {
                                         dis.close();
                                         client.close();
                                         }
                                    catch(IOException ioe){}
                                }
                              }
//οι πληροφορίες επικολλούνται στο αρχείο info.txt
                              try
                    {
                    //στο αρχείο στο οποίο αποθηκεύονται οι πληροφορίες, διαχωρίζονται με αλλαγή
                    //γραμμής
                    tmp=tmp+'\r';
                    tmp=tmp+'\n';
                    FileOutputStream fos = new FileOutputStream(file_name,true);
                    fos.write(tmp.getBytes());
                    fos.flush();
                    fos.close();
                   }
                 catch(IOException e)  { }
                    try
                     {
                        dis.close();
                        client.close();
                     }
                     catch(IOException error){}
   }
//η μέθοδος αυτή διαβάζει την πρώτη γραμμή από το spec.nfo ώστε να βρει το αναγνωριστικό του
//προβλήματος
public void connect_receive_lineII(String ip_adress,int port_number,String file_name)
   {
      Socket client = null;
      BufferedInputStream bis = null;
      BufferedReader dis=null;
      boolean repeat=true;
      //προσπάθεια για σύνδεση
         while(repeat&&(runner==thisThread))
         {
            try
            {
                features.setText("Connecting Server ...");
               client = new Socket(ip_adress,port_number);
               repeat = false;
            }
            catch(IOException e)
            {
                     features.setText("Server Not Found ...");
            }
```

```
        }
//στο σημείο αυτό έχει πραγματοποιηθεί σύνδεση
//αμφίδρομη σύνδεση για ανταλλαγή δεδομένων
        try
        {
           bis = new BufferedInputStream(client.getInputStream());
           dis = new BufferedReader(new InputStreamReader(bis));
        }
        catch(IOException e)
        {
        }
         features.setText("Initializing ...");
        String tmp="";
//επαναληπτικός βρόχος 5 στιγμών
          for(int g=0;g<6;g++)
                {
                    if(runner!=thisThread)break;
                     try
                     {
                    if(g!=5)
                    {
                    String down = dis.readLine();
                    if(g==0) new_name = down;
                    tmp=tmp+down+" ";
                    }
                    else
                    tmp=tmp+dis.readLine();
                    }
                  catch(IOException iods)
                    {
                      try
                      {
                              dis.close();
                              client.close();
                      }
                        catch(IOException ioe){}
                     }
                }
               try
                    {
                       dis.close();
                       client.close();
                    }
                    catch(IOException error){}
  }
  //η μέθοδος επιστρέφει true αν υπάρχει το αναγνωριστικό του προβλήματος στο αρχείο info.txt
  // και επιστρέφει false αν δεν υπάρχει
  public boolean isEqual(String file_name,String info)
  {
        boolean m_bln=true;
        try
        {
                        FileInputStream fis = new FileInputStream(info);
                        BufferedInputStream bis = new BufferedInputStream(fis);
                        BufferedReader dis=new BufferedReader(new InputStreamReader(bis));
                        try
                        {
                        boolean bln=true;
                        while(bln==true)
                        {
```

```
                                        String rd=dis.readLine();
                                        if(rd==null)
                                        {
                                                bln=false;
                                                break;
                                        }
//από το αρχείο info.txt διαβάζεται η πρώτη γραμμή μέχρι το πρώτο κενό
//η τιμή αυτή αποθηκεύεται στο chr πίνακα  και συγκρίνεται με το αναγνωριστικό του προβλήματος

                        char[] chr=new char[100];
                                        rd.getChars(0,rd.indexOf(' '),chr,0);
                                        String r=String.valueOf(chr,0,rd.indexOf(' '));
                                        if(r.compareTo(file_name)==0)
                                        {
                                        m_bln=false;
                                        }
                                }
                                return m_bln;
                                }
                                catch(Exception e)
                                {
                                        return m_bln;
                                }
                        }
                        catch(FileNotFoundException e)
                        {
                                return true;
                        }

    }
    public void run()
    {
            thisThread=Thread.currentThread();
            //η θέση στην οποία βρίσκεται το αρχείο info.txt αναφέρεται ως path
            info=path+"info.txt";
            //εύρεση αναγνωριστικού προβλήματος
            connect_receive_lineII(ip_adress,port_number+16,info);
            //επικολλάται στο όνομα το path
            fil = path+new_name;
            System.out.println(fil);
            //ανάγνωση αρχείων βαρών από τις θύρες 6001-6015
            connect_receive(ip_adress,port_number+1,fil+".tmpaxs");
            connect_receive(ip_adress,port_number+2,fil+".tmpawc");
            connect_receive(ip_adress,port_number+3,fil+".tmpaws");
            connect_receive(ip_adress,port_number+4,fil+".tmpavc");
            connect_receive(ip_adress,port_number+5,fil+".tmpavs");

            connect_receive(ip_adress,port_number+6,fil+".tmpbxs");
            connect_receive(ip_adress,port_number+7,fil+".tmpbwc");
            connect_receive(ip_adress,port_number+8,fil+".tmpbws");
            connect_receive(ip_adress,port_number+9,fil+".tmpbvc");
            connect_receive(ip_adress,port_number+10,fil+".tmpbvs");

            connect_receive(ip_adress,port_number+11,fil+".tmpcxs");
            connect_receive(ip_adress,port_number+12,fil+".tmpcwc");
            connect_receive(ip_adress,port_number+13,fil+".tmpcws");
            connect_receive(ip_adress,port_number+14,fil+".tmpcvc");
            connect_receive(ip_adress,port_number+15,fil+".tmpcvs");
            //έλεγχος αν η τιμή του αναγνωριστικού του νέου προβλήματος συμπεριλαμβάνεται στο
            //αρχείο
```

```
if(isEqual(new_name,info))
{
        connect_receive_line(ip_adress,port_number+16,info);
        features.setText("New Set Added");
}
else
{
features.setText("Set Not Added");
}
File axs=new File(fil+".tmpaxs");
File awc=new File(fil+".tmpawc");
File aws=new File(fil+".tmpaws");
File avc=new File(fil+".tmpavc");
File avs=new File(fil+".tmpavs");

File bxs=new File(fil+".tmpbxs");
File bwc=new File(fil+".tmpbwc");
File bws=new File(fil+".tmpbws");
File bvc=new File(fil+".tmpbvc");
File bvs=new File(fil+".tmpbvs");

File cxs=new File(fil+".tmpcxs");
File cwc=new File(fil+".tmpcwc");
File cws=new File(fil+".tmpcws");
File cvc=new File(fil+".tmpcvc");
File cvs=new File(fil+".tmpcvs");

//εάν είχαμε επιτυχής σύνδεση, δηλαδή αν τα αρχεία βαρών έχουν μη-μηδενικά μεγέθη, τότε
//αφαιρείται το πρόθεμα tmp από την κατάληξη ,ώστε να αποκτήσουν την κανονική ονομασία
//τους
if((axs.length()!=0)&&(awc.length()!=0)&&(aws.length()!=0)&&(avc.length()!=0)&&
(avs.length()!=0)&&(bxs.length()!=0)&&
(bwc.length()!=0)&&(bws.length()!=0)&&
(bvc.length()!=0)&&(bvs.length()!=0)&&(cxs.length()!=0))
{
        if(new File(fil+".axs").exists())
        new File(fil+".axs").delete();
        if(new File(fil+".awc").exists())
        new File(fil+".awc").delete();
        if(new File(fil+".aws").exists())
        new File(fil+".aws").delete();
        if(new File(fil+".avc").exists())
        new File(fil+".avc").delete();
        if(new File(fil+".avs").exists())
        new File(fil+".avs").delete();
        axs.renameTo(new File(fil+".axs"));
        awc.renameTo(new File(fil+".awc"));
        aws.renameTo(new File(fil+".aws"));
        avc.renameTo(new File(fil+".avc"));
        avs.renameTo(new File(fil+".avs"));

        if(new File(fil+".bxs").exists())
        new File(fil+".bxs").delete();
        if(new File(fil+".bwc").exists())
        new File(fil+".bwc").delete();
        if(new File(fil+".bws").exists())
        new File(fil+".bws").delete();
        if(new File(fil+".bvc").exists())
        new File(fil+".bvc").delete();
        if(new File(fil+".bvs").exists())
```

```
                    new File(fil+".bvs").delete();
                    bxs.renameTo(new File(fil+".bxs"));
                    bwc.renameTo(new File(fil+".bwc"));
                    bws.renameTo(new File(fil+".bws"));
                    bvc.renameTo(new File(fil+".bvc"));
                    bvs.renameTo(new File(fil+".bvs"));

                    if(new File(fil+".cxs").exists())
                    new File(fil+".cxs").delete();
                    if(new File(fil+".cwc").exists())
                    new File(fil+".cwc").delete();
                    if(new File(fil+".cws").exists())
                    new File(fil+".cws").delete();
                    if(new File(fil+".cvc").exists())
                    new File(fil+".cvc").delete();
                    if(new File(fil+".cvs").exists())
                    new File(fil+".cvs").delete();
                    cxs.renameTo(new File(fil+".cxs"));
                    cwc.renameTo(new File(fil+".cwc"));
                    cws.renameTo(new File(fil+".cws"));
                    cvc.renameTo(new File(fil+".cvc"));
                    cvs.renameTo(new File(fil+".cvs"));
            //Επιτυχής μετάδοση οπότε και κατάλληλο μήνυμα
                    features.setText("Initialization Completed");
            }
            else
            {
//Ανεπιτυχής μετάδοση
            features.setText("Disconnected - Retry");
            }
    }
//μέθοδος έναρξης
    void start()
    {
            if(runner==null)
            {
            runner=new Thread(this);
            runner.start();
            }
    }
    //μέθοδος διακοπής
    void stop()
    {
            runner=null;
    }
}



//*******
//gif.java
//*******
//η κλάση υλοποιεί ένα κινούμενο γραφικό ως γρήγορη εναλλαγή στάσιμων εικόνων και δημιουργία
//της ψευδαίσθησης  της κίνησης
import java.awt.*;

class gif extends Canvas implements Runnable
{
    Image img[] = new Image[29];
    int index;
```

```
   Thread runner;
//αρχικοποίηση του πίνακα Image με τις στάσιμες εικόνες
   gif()
   {
      super();
      for(int i = 0;i<img.length;i++)
            img[i] = Toolkit.getDefaultToolkit().getImage("bar"+
                                          String.valueOf(i)+".jpg");
//ο δείκτης δείχνει την μεσαία εικόνα στην σειριακή εναπόθεση
      index = 14;
   }
//μέθοδος εκκίνησης
   void start()
   {
      if(runner==null)
      {
         runner = new Thread(this);
         runner.start();
      }
   }
//μέθοδος διακοπής
   void stop()
   {
      runner = null;
   }
//μέθοδος νήματος
   public void run()
   {
      Thread thisThread = Thread.currentThread();

      while(runner==thisThread)
      {
          while(index<img.length)
          {
//σχεδιασμός των εικόνων με χρονικές διαφορές 33msec
//ο σειριακός κύκλος πραγματοποιείται από το 1 έως 29
         repaint();
         try
         {
            Thread.sleep(33);
         }
         catch(InterruptedException e)   {}
         index++;
          }
          index=27;

         while(index>=0)
          {
         repaint();
         try
         {
            Thread.sleep(33);
         }
         catch(InterruptedException e)   {}
         index--;
          }
        index=0;
      }
   }
//μέθοδος ενημέρωσης του πλαισίου
```

```
   public void update(Graphics g)
   {
      paint(g);
   }
//σχεδιασμός στην οθόνη
   public void paint(Graphics g)
   {
      if(img[index]!=null)
         g.drawImage(img[index],0,0,300,50,this);
   }

}


//*******
//random_num.java
//*******
/**
 *Προσομοίωση αισθητήρα τύπου Α (ασύγχρονη μετάδοση)
 *ο αισθητήρας δημιουργεί τυχαίους αριθμούς από ένα διάστημα
 *που καθορίζει ο χρήστης και τους στέλνει σε μια διεύθυνση, σε κατάλληλο port
 *Ο χρήστης καθορίζει επίσης και το ρυθμό μετάδοσης των αριθμών
 */
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;


public class random_num extends Frame implements ActionListener
{
   /**
    * Δηλώσεις μεταβλητών
    */
          public static Client et;
          public boolean m_pushed=true;
          GridBagLayout layout = new GridBagLayout();
          GridBagConstraints constraints =new GridBagConstraints();

          //Χρώμα που βρίσκονται τα κουμπιά lbl1,lbl2
          Color c=new Color(0,191,255);

          //Το κάτω άκρο του διαστήματος τυχαίων αριθμών
          Label lbl1=new Label("From value:");

          //Το άνω άκρο του διαστήματος τυχαίων αριθμών
          Label lbl2=new Label("To value  :");

          //Συχνότητα αποστολής
          Label lbl3=new Label("Timer (msec):");

          //IP Διεύθυνση
          Label lbl4=new Label("IP address :");

          //Αριθμός θύρας
          Label lbl5=new Label("Port number :");

          TextField txt1=new TextField();
          TextField txt2=new TextField();
```

```
TextField txt3=new TextField();
TextField txt4=new TextField();
TextField txt5=new TextField();

//Τα κουμπιά λειτουργίας
Button btn1=new Button("Start");
Button btn2=new Button("Stop");

// κατασκευαστής
public random_num()
{
        //Η επικεφαλίδα του πλαισίου
        super ("  Simulated Sensor  ");
        //Το χρώμα του πλαισίου είναι lightGray
        setBackground(Color.lightGray);

        setLayout(layout);
        constraints.fill=GridBagConstraints.BOTH;

        //Αρχική κάτω τιμή 0
        add(lbl1);
        add(txt1);
        txt1.setText("0");
        build(constraints,0,0,1,1,100,20);
        layout.setConstraints(lbl1,constraints);
        build(constraints,1,0,1,1,100,0);
        layout.setConstraints(txt1,constraints);
        lbl1.setBackground(c);

        //Αρχική πάνω τιμή 1
        add(lbl2);
        add(txt2);
        txt2.setText("1");
        build(constraints,2,0,1,1,100,0);
        layout.setConstraints(lbl2,constraints);
        build(constraints,3,0,1,1,100,0);
        layout.setConstraints(txt2,constraints);
        lbl2.setBackground(c);

        //Αρχικό χρονικό διάστημα 1 sec
        add(lbl3);
        add(txt3);
        txt3.setText("1000");
        build(constraints,0,1,1,1,70,20);
        layout.setConstraints(lbl3,constraints);
        build(constraints,1,1,1,1,30,0);
        layout.setConstraints(txt3,constraints);

        //Αρχική διεύθυνση IP 169.254.29.26
        add(lbl4);
        add(txt4);
        txt4.setText("169.254.29.26");


        build(constraints,0,2,1,1,90,20);
        layout.setConstraints(lbl4,constraints);
        build(constraints,1,2,2,1,10,0);
        layout.setConstraints(txt4,constraints);

        //Αρχικός αριθμός θύρας 8001
```

```
                add(lbl5);
                add(txt5);
                txt5.setText("8001");
                build(constraints,0,3,1,1,70,20);
                layout.setConstraints(lbl5,constraints);
                build(constraints,1,3,1,1,30,0);
                layout.setConstraints(txt5,constraints);
                txt5.setBackground(Color.red);

                //Αρχικοποίηση κουμπιών
                constraints.fill=GridBagConstraints.NONE;
                add(btn1);
                add(btn2);
                btn1.setBackground(Color.lightGray);
                btn2.setBackground(Color.lightGray);
                build(constraints,0,4,2,1,0,20);
                layout.setConstraints(btn1,constraints);
                build(constraints,2,4,2,1,0,0);
                layout.setConstraints(btn2,constraints);
                btn1.addActionListener(this);
                btn2.addActionListener(this);

                //Μέγεθος παραθύρου
                setSize(250,150);
                setVisible(true);

                //προσθήκη για να κλείνει το παράθυρο από
                //το X στο πάνω δεξιά μέρος του πλαισίου
                addWindowListener
                (
                        new WindowAdapter()
                        {
                                public void windowClosing(WindowEvent e)
                                {
                                        try
                                        {
                                        if(et.socket!=null)et.socket.close();
                                        }
                                        catch(IOException u)
                                        {
                                                u.printStackTrace();
                                        }
                                        System.exit(0);
                                }
                        }
                );
        }

//μέθοδος αρχικοποίησης των συστατικών του GridBagLayout
public void build(GridBagConstraints gbc,int gx,int gy, int gw,int gh,int wx,int wy)
{
                gbc.gridx=gx;
                gbc.gridy=gy;
                gbc.gridwidth=gw;
                gbc.gridheight=gh;
                gbc.weightx=wx;
                gbc.weighty=wy;
   }

   //μέθοδος που χειρίζεται τα γεγονότα που συμβαίνουν
```

```
    public  void actionPerformed(ActionEvent e)
        {
                //Κουμπί Start
                if (e.getSource()==btn1)
                {
                        //Το κουμπί μπορεί να πατηθεί μόνο μία φορά
                        //μετά πρέπει να πατηθεί Stop για να ξεκλειδώσει
                        if(m_pushed==true)
                        {
                        m_pushed=false;
                        btn1.setBackground(Color.gray);
                        btn1.setEnabled(false);
                        System.out.println("START");
                        et=new Client(this);
                        et.ip_adr=txt4.getText();
                        et.port_num=txt5.getText();
                        et.from_num=txt1.getText();
                        et.to_num=txt2.getText();
                        et.tim_val=txt3.getText();
                        et.start();
                }

        }
                //Κουμπί STOP
                else if (e.getSource()==btn2)
                {
                        m_pushed=true;
                        btn1.setBackground(Color.lightGray);
                        btn1.setEnabled(true);
                        System.out.println("STOP");
                        et.requestStop();
                        txt5.setBackground(Color.red);
                }
        }



        //Κύρια μέθοδος
        //Δεν χρειάζονται ορίσματα από το command prompt
        public static void main(String args[])
        {
                random_num app=new random_num();
        }
}

//********
//client.java
//********
/**
 * η κλάση αυτή πραγματοποιεί τη σύνδεση με το PDA
 * τα αντικείμενα της καλούνται ως νήματα
 */
class Client extends Thread
{
//η μεταβλητή m_bln λειτουργεί ως σημαφόρος για να σταματούν τα νήματα
   public volatile boolean m_bln=false;
   public String from_num="";
   public String to_num="";
   public String tim_val="";
   public String ip_adr="";
```

```
  public String port_num="";
  public String ran_num="";
  Random r=new Random();
  public random_num app;
  public static Socket socket;

  //κατασκευαστής
  public Client(random_num app)
  {
        this.app=app;
  }

  //οι εντολές που τρέχουν ως ξεχωριστή διαδικασία
  public  void run()
   {
        //η m_bln είναι υπεύθυνη για να σταματά το νήμα
         m_bln=false;
         boolean tmp_bln=true;
         String[][] strb=null;
                while(m_bln==false)
                  {
                        socket=null;
                        OutputStream os=null;
                        InputStream is=null;
 //χρωματισμός πράσινος σημαίνει ότι η σύνδεση έχει πραγματοποιηθεί
 //και ότι έχουμε ανταλλαγή δεδομένων
                        if(tmp_bln==true)
                        app.txt5.setBackground(Color.green);
                        int ascii;
 //ο τυχαίος αριθμός που στέλνεται ανήκει στο δίαστημα [a,b]
//η μετατροπή από το [0,1] στο [a,b]
//r$[0,1] ->r$[a,b]
//γίνεται με τον τύπο
//r*(b-a)+a
                        ran_num=String.valueOf(r.nextDouble()*(Double.parseDouble(from_num)
                        -Double.parseDouble(to_num))+Double.parseDouble(to_num))+"X";
                         System.out.println(ran_num);
                        try
                        {
                        InetAddress ipaddress = InetAddress.getByName(ip_adr);
                        InetSocketAddress socketaddress = new InetSocketAddress(ipaddress,
                        Integer.parseInt(port_num));
                        socket = new Socket();
//τίθεται όριο στο χρόνο τον οποίο θα περιμένει το socket
//για να σταλεί πληροφορία από server ίση με 1 sec
                        try {
                          socket.setSoTimeout(1000); // 1 second
                            }
                        catch(SocketException e)
                           {
                            e.printStackTrace();
                           }
//σύνδεση του socket με το PDA
                        socket.connect(socketaddress);
                        os = socket.getOutputStream();
                        is = socket.getInputStream();
//το String που στέλνεται πρέπει να τελειώνει με το 'X'
//αυτή είναι σύμβαση ανάμεσα σε αισθητήρα και PDA
                        os.write(ran_num.getBytes());
                        os.flush();
```

```
//αν έχουμε επιτυχή μετάδοση τότε λαμβάνετε '@'
                                ascii = is.read();
                                System.out.println((char)ascii);
//υλοποίηση για να κλείνει το πρόγραμμα από το PDA
                               if((char)ascii=='#')
                                        {
                                         try
                                          {
                                          if(socket!=null)socket.close();
                                          }
                                          catch(IOException mn)
                                          {
                                          mn.printStackTrace();
                                          }
                                        System.exit(0);
                                        }
                               os.close();
                               is.close();
                               socket.close();
                               tmp_bln=true;
                               //το χρονικό διάστημα αναμονής είναι ένας τυχαίος αριθμός
                               //από 0 έως την τιμή που θέτει ο χρήστης
                               try
                                          {
                                          double tmp=Integer.parseInt(tim_val)*r.nextDouble();
                                          sleep((int)tmp);
                                          }
                                          catch(InterruptedException e)
                                          {
                                          };
                             }
                             //εξαίρεση αν υπάρξει πρόβλημα κατά την ανταλλαγή
                             //δεδομένων
                                catch(IOException e)
                                {
                                tmp_bln=false;
                                e.printStackTrace();
        //εάν η σύνδεση διακόπηκε από το PDA τότε το χρώμα είναι πορτοκαλί
                                if(m_bln==false)app.txt5.setBackground(Color.orange);
                               else
                               //αλλιώς το χρώμα είναι κόκκινο αφού έχει πατηθεί το Stop
                               app.txt5.setBackground(Color.red);
                               //os
                                          try
                                                   {
                                                    if(os!=null)os.close();
                                                    }
                                                   catch(IOException mn)
                                    {
                                                   mn.printStackTrace();
                                    }
                                    //is
                                      try
                                                   {
                                                    if(is!=null)is.close();
                                                    }
                                                   catch(IOException mn)
                                    {
                                                   mn.printStackTrace();
                                    }
```

```
                                        //socket
                                        try
                                                {
                                                if(socket!=null)socket.close();
                                                }
                                                catch(IOException mn)
                                        {
                                                mn.printStackTrace();
                                        }
                                }
                                //αυτές οι εντολές εκτελούνται απαραίτητα
                                finally
                                 {
                                 //os
                                        try
                                                {
                                                if(os!=null)os.close();
                                                }
                                                catch(IOException mn)
                                        {
                                                mn.printStackTrace();
                                        }
                                        //is
                                          try
                                                {
                                                if(is!=null)is.close();
                                                }
                                                catch(IOException mn)
                                        {
                                                mn.printStackTrace();
                                           }
                                        //socket
                                        try
                                                {
                                                if(socket!=null)socket.close();
                                                }
                                                catch(IOException mn)
                                        {
                                                mn.printStackTrace();
                                        }
                                        }
                        }
        }
        //η μέθοδος που καλείται από το εξωτερικό νήμα
        //για να κλείσει αυτό το νήμα
        public void requestStop()
        {
        //ο σημαφόρος m_bln αλλάζει τιμή
                m_bln=true;
                try
                        {
                                if(socket!=null)socket.close();
                        }
                                catch(IOException mn)
                        {
                                mn.printStackTrace();
                        }
        }
}
```

```
//***************
//random_numII.java
//***************
/**
 *Προσομοίωση αισθητήρα τύπου Β (σύγχρονη μετάδοση)
 *ο αισθητήρας διαβάζει ομάδες τιμών από ένα αρχείο
 *που καθορίζει ο χρήστης και τις στέλνει σε μια διεύθυνση στο 8000 port
 *Ο χρήστης καθορίζει επίσης και το ρυθμό μετάδοσης των ομάδων
 */
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;


public class random_numII extends Frame implements ActionListener
{
   /**
    * Δηλώσεις μεταβλητών
    */
        public static ClientII et;
        public boolean m_pushed=true;
        GridBagLayout layout = new GridBagLayout();
        GridBagConstraints constraints =new GridBagConstraints();

        //Συχνότητα αποστολής
        Label lbl1=new Label("Timer(msec)");

        //αριθμός χαρακτηριστικών
        //καθορίζει τον αριθμό των στοιχείων
        //που στέλνει ο αισθητήρας
        Label lbl2=new Label("Features:");

        //το όνομα του αρχείου από το οποίο γίνεται το διάβασμα
        Label lbl3=new Label("Filename");

        //IP Διεύθυνση
        Label lbl4=new Label("IP address :");

        //Αριθμός θύρας
        Label lbl5=new Label("Port number :");

        TextField txt1=new TextField();
        TextField txt2=new TextField();
        TextField txt3=new TextField();
        TextField txt4=new TextField();
        TextField txt5=new TextField();

        //Τα κουμπιά λειτουργίας
        Button btn1=new Button("Start");
        Button btn2=new Button("Stop");

        //Ο κατασκευαστής
        public random_numII()
        {
                //Η επικεφαλίδα του πλαισίου
                super ("  Simulated Sensor II ");
                //Το χρώμα του πλαισίου είναι lightGray
```

```
setBackground(Color.lightGray);
setLayout(layout);
constraints.fill=GridBagConstraints.BOTH;

//Αρχικό χρονικό διάστημα 1 sec
add(lbl1);
add(txt1);
txt1.setText("1000");
build(constraints,0,0,1,1,100,20);
layout.setConstraints(lbl1,constraints);
build(constraints,1,0,1,1,100,0);
layout.setConstraints(txt1,constraints);


//Αρχικός αριθμός χαρακτηριστικών 0
add(lbl2);
add(txt2);
txt2.setText("0");
build(constraints,2,0,1,1,100,0);
layout.setConstraints(lbl2,constraints);
build(constraints,3,0,1,1,100,0);
layout.setConstraints(txt2,constraints);


//Αρχικό όνομα αρχείου _.txt
add(lbl3);
add(txt3);
txt3.setText("_.txt");
build(constraints,0,1,1,1,90,20);
layout.setConstraints(lbl3,constraints);
build(constraints,1,1,2,1,10,0);
layout.setConstraints(txt3,constraints);

//Αρχική διεύθυνση IP 169.254.29.26
add(lbl4);
add(txt4);
txt4.setText("169.254.29.26");


build(constraints,0,2,1,1,90,20);
layout.setConstraints(lbl4,constraints);
build(constraints,1,2,2,1,10,0);
layout.setConstraints(txt4,constraints);

//Αρχικός αριθμός θύρας 8000
add(lbl5);
add(txt5);
txt5.setText("8000");
build(constraints,0,3,1,1,70,20);
layout.setConstraints(lbl5,constraints);
build(constraints,1,3,1,1,30,0);
layout.setConstraints(txt5,constraints);
txt5.setBackground(Color.red);

//Αρχικοποίηση κουμπιών
constraints.fill=GridBagConstraints.NONE;
add(btn1);
add(btn2);
btn1.setBackground(Color.lightGray);
btn2.setBackground(Color.lightGray);
```

```
                build(constraints,0,4,2,1,0,20);
                layout.setConstraints(btn1,constraints);
                build(constraints,2,4,2,1,0,0);
                layout.setConstraints(btn2,constraints);
                btn1.addActionListener(this);
                btn2.addActionListener(this);

                //Μέγεθος παραθύρου
                setSize(250,150);
                setVisible(true);

                //προσθήκη για να κλείνει το παράθυρο από
                //το X στο πάνω δεξία μέρος του πλαισίου
                addWindowListener
                (
                        new WindowAdapter()
                        {
                                public void windowClosing(WindowEvent e)
                                {
                                        //allagi
                                        try
                                        {
                                        if(et.socket!=null)et.socket.close();
                                        }
                                        catch(IOException u)
                                        {
                                                u.printStackTrace();
                                        }
                                        System.exit(0);
                                }
                        }
                );
        }
        //μέθοδος αρχικοποίησης των συστατικών του GridBagLayout
        public void build(GridBagConstraints gbc,int gx,int gy, int gw,int gh,int wx,int wy)
        {
                gbc.gridx=gx;
                gbc.gridy=gy;
                gbc.gridwidth=gw;
                gbc.gridheight=gh;
                gbc.weightx=wx;
                gbc.weighty=wy;
}

//μέθοδος που χειρίζεται τα γεγονότα που συμβαίνουν
public  void actionPerformed(ActionEvent e)
        {
                //Κουμπί Start
                if (e.getSource()==btn1)
                {
                        //Το κουμπί μπορεί να πατηθεί μόνο μία φορά
                        //μετά πρέπει να πατηθεί Stop για να ξεκλειδώσει
                        if(m_pushed==true)
                        {
                        m_pushed=false;
                        btn1.setBackground(Color.gray);
                        btn1.setEnabled(false);
                        System.out.println("START");
                        et=new ClientII(this);
```

```
                    et.ip_adr=txt4.getText();
                    et.port_num=txt5.getText();
                    et.tim_val=txt1.getText();
                    et.num_feat=txt2.getText();
                    et.file_name=txt3.getText();
                    et.start();
        }

                }
                //Κουμπί STOP
                else if (e.getSource()==btn2)
                {
                        m_pushed=true;
                        btn1.setBackground(Color.lightGray);
                        btn1.setEnabled(true);
                        System.out.println("STOP");
                        et.requestStop();
                        txt5.setBackground(Color.red);
                }

        }
         //Κύρια μέθοδος
        //Δεν χρειάζονται ορίσματα από το command prompt
        public static void main(String args[])
        {
                random_numII app=new random_numII();
        }
}

//*******
//clientII.java
//*******
/**
 * η κλάση αυτή πραγματοποιεί τη σύνδεση με το PDA
 * τα αντικείμενα της καλούνται ως νήματα
 */
class ClientII extends Thread
{
        //η μεταβλητή m_bln λειτουργεί ως σημαφόρος για να σταματούν τα νήματα
  public volatile boolean m_bln=false;
  public String num_feat="";
  public String file_name="";
  public String tim_val="";
  public String ip_adr="";
  public String port_num="";
  public String ran_num="";
  public random_numII app;
  public static int count=0;
  public static Socket socket;
  Random r=new Random();

  //κατασκευαστής
  public ClientII(random_numII app)
  {
        this.app=app;
  }

  //οι εντολές που τρέχουν ως ξεχωριστή διαδικασία
  public  void run()
   {
```

```
      //η m_bln είναι υπεύθυνη για να σταματά το νήμα
       m_bln=false;
       boolean tmp_bln=true;
       String[][] strb=null;
       int num=Integer.parseInt(num_feat);
          int num_lenght=0;
FileInputStream fis = null;
 BufferedInputStream bis = null;
 DataInputStream dis = null;
                //οι παρακάτω εντολές βρίσκουν το πλήθος των ομάδων των στοιχείων του αρχείου
          try
            {
                 fis = new FileInputStream(file_name);
                bis = new BufferedInputStream(fis);
                 dis = new DataInputStream(bis);
                 while(true)
                            {
                           dis.readDouble();
                            num_lenght++;
                             }
             }
            catch(IOException e)
             {
               e.printStackTrace();
               try
               {
                 dis.close();
               }
               catch
               (IOException fdh)
               {
                 fdh.printStackTrace();
               }
            }
       //αποθηκεύονται οι τιμές σε πίνακα για να είναι άμεσα προσπελάσιμες
                 try
                   {
                 fis = new FileInputStream(file_name);
                   bis = new BufferedInputStream(fis);
                 dis = new DataInputStream(bis);
                 strb=new String[num_lenght/num][num+1];
                 for(int i=0;i<strb.length;i++)
                   {
                            for(int j=1;j<strb[0].length;j++)
                            {
                            strb[i][j]=String.valueOf(dis.readDouble());
                            }
                 }
                 }
                  catch(IOException e)
                   {
               e.printStackTrace();
               try
               {
                 dis.close();
               }
               catch
               (IOException fdh)
               {
                 fdh.printStackTrace();
```

```
    }
}
    //ξεκινά η μετάδοση στοιχείων
    while(m_bln==false)
    {
       socket=null;
       OutputStream os=null;
       InputStream is=null;
    if(tmp_bln==true)
    //χρωματισμός πράσινος σημαίνει ότι η σύνδεση έχει πραγματοποιηθεί
      //και ότι έχουμε ανταλλαγή δεδομένων
    app.txt5.setBackground(Color.green);
          int ascii;
       try
         {
         //στέλνονται ομάδες στοιχείων
                    for(int b=1;b<=Integer.parseInt(num_feat);b++)
                       {
                 //το String που στέλνεται πρέπει να τελειώνει με το 'X'
                //αυτή είναι σύμβαση ανάμεσα σε αισθητήρα και PDA
                ran_num=strb[count][b]+"X";
                InetAddress ipaddress = InetAddress.getByName(ip_adr);
                InetSocketAddress socketaddress = new InetSocketAddress(ipaddress,
                Integer.parseInt(port_num));
                socket = new Socket();
                //τίθεται όριο στο χρόνο τον οποίο θα περιμένει το socket
                //για να σταλεί πληροφορία από server ίση με 1 sec
                try {
                   socket.setSoTimeout(1000); // 1 second
                }
                catch(SocketException e)
                {
                  e.printStackTrace();
                }
                //σύνδεση του socket με το PDA
                        socket.connect(socketaddress);
                         os = socket.getOutputStream();
                         is = socket.getInputStream();
                         System.out.print(ran_num+" ");
                         os.write(ran_num.getBytes());
                         os.flush();
                         //αν έχουμε επιτυχή μετάδοση τότε λαμβάνετε '@'
                         ascii=is.read();
                         System.out.print((char)ascii+" ");
                         //υλοποίηση για να κλείνει το πρόγραμμα από το PDA
                         if((char)ascii=='#')
                         {
                           try
                           {
                           if(socket!=null)socket.close();
                           }
                          catch(IOException mn)
                          {
                          mn.printStackTrace();
                          }
                          System.exit(0);
                         }
                         os.close();
                         is.close();
                         socket.close();
```

```
                tmp_bln=true;
                }
                  try
                  {
                  //το χρονικό διάστημα αναμονής είναι ένας τυχαίος αριθμός
                  //από 0 έως την τιμή που θέτει ο χρήστης
                              double tmp=Integer.parseInt(tim_val)*r.nextDouble();
                              sleep((int)tmp);
                              }
                     catch(InterruptedException e)
                              {
                                        e.printStackTrace();
                              };
                     count++;
                     if(count>=strb.length)count=0;
  }
  //εξαίρεση αν υπάρξει πρόβλημα κατά την ανταλλαγή
    //δεδομένων
catch(IOException e)
  {
       tmp_bln=false;
     e.printStackTrace();
     if(m_bln==false)
     //εάν η σύνδεση διακόπηκε από το PDA τότε το χρώμα είναι πορτοκαλί
     app.txt5.setBackground(Color.orange);
     else
     //αλλιώς το χρώμα είναι κόκκινο αφού έχει πατηθεί το Stop
     app.txt5.setBackground(Color.red);
      //os
                     try
                              {
                              if(os!=null)os.close();
                              }
                              catch(IOException mn)
                     {
                              mn.printStackTrace();
                     }
        //is
                     try
                              {
                              if(is!=null)is.close();
                              }
                              catch(IOException mn)
                     {
                              mn.printStackTrace();
                     }
        //socket
                     try
                              {
                              if(socket!=null)socket.close();
                              }
                              catch(IOException mn)
                     {
                              mn.printStackTrace();
                     }
  }
  //αυτές οι εντολές εκτελούνται απαραίτητα
   finally
   {
   //os
```

```
                                try
                                        {
                                        if(os!=null)os.close();
                                        }
                                        catch(IOException mn)
                                {
                                        mn.printStackTrace();
                                }
                                //is
                                  try
                                        {
                                        if(is!=null)is.close();
                                        }
                                        catch(IOException mn)
                                {
                                        mn.printStackTrace();
                                   }
                                //socket
                                try
                                        {
                                        if(socket!=null)socket.close();
                                        }
                                        catch(IOException mn)
                                {
                                        mn.printStackTrace();
                                }
                                }
                        }
    }
    //η μέθοδος που καλείται από το εξωτερικό νήμα
    //για να κλείσει αυτό το νήμα
    public void requestStop()
    {
    //ο σημαφόρος m_bln αλλάζει τιμή
            m_bln=true;
            try
                    {
                            if(socket!=null)socket.close();
                    }
                            catch(IOException mn)
                    {
                            mn.printStackTrace();
                    }
    }
}

//********
//server.java
//********
//η κλάση αυτή υλοποιεί την ασύγχρονη μετάδοση δεδομένων από την πλευρά του PDA
import java.net.ServerSocket;
import java.net.Socket;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Random;
import java.awt.*;


public class Server extends Thread
```

```
{
        //ο πίνακας στο οποίο αποθηκεύονται τα δεδομένα από τους αισθητήρες
        public static String xc[];
        public Socket clientSocket;
        public ServerSocket serverSocket;
        //η μεταβλητή για να σταματά η σύνδεση PDA-αισθητήρες
        public volatile boolean m_chk;
        //η μεταβλητή tmp_access χρησιμοποιείται για να ελέγχει τις προσπελάσεις στο xc[]
        public Result tmp_access;
        public String item;
        public int inp;
        public TextField features;
        public String port;
        public EchoThread echo=null;

        //κατασκευαστής
        public Server(String port,String name_item,int input,TextField feat,Result rslt)
        {
                super(port);
                 features = feat;
                 this.port=port;
                this.m_chk=true;
                this.item=name_item;
                this.inp=input;
                xc=new String[input+1];
                //αρχικοποίηση του πίνακα με μηδενικές τιμές
                for(int j=0;j<xc.length;j++)
                {
                        xc[j]="0";
                }
                tmp_access=rslt;
                //αρχικοποίηση του serversocket
                try
        {
        serverSocket = new ServerSocket(Integer.parseInt(port)+8000);
        }
        catch(IOException e)
        {
                e.printStackTrace();
        }
        }
//μέθοδος που σταματά το τρέχων νήμα
public void requestStop()
   {
        m_chk=false;
   }

   //μέθοδος υλοποίησης του νήματος
   public void run()
   {
   //αέναος βρόχος, για να αντιμετωπιστεί το bug της Jeode
             while(true)
             {
                        clientSocket=null;
                try
                {
                        clientSocket = serverSocket.accept();
                }
                catch(IOException e)
                {
```

```
                    e.printStackTrace();
                    continue;
                }
                //η μεταβλητή m_chk καθορίζετε από τις επιλογές του χρήστη
                if(m_chk==true)
                {
                    echo=new EchoThread(clientSocket,this,inp);
                    echo.start();
                }
            }
        }
}

//**************
//EchoThread.java
//**************
//η κλάση υλοποιεί το νήμα που καλείται για κάθε αίτηση πελάτη
class EchoThread extends Thread
{
                public static boolean check=true;;
                public int features;
                Socket s;
                Server sr;
                int inp;
                InputStream is;
                OutputStream os;
                //κατασκευαστής
                EchoThread(Socket s,Server sr,int input)
                {
                    this.s=s;
                    this.sr=sr;
                    this.inp=input;
                }
//μέθοδος νήματος
public void run()
{
    int ascii;
    StringBuffer strb=new StringBuffer();
    Random r = new Random();
    try
    {
      is = s.getInputStream();
      os = s.getOutputStream();
      //σύμβαση οι τιμές να τελειώνουν σε 'X'
      do
      {
        ascii = is.read();
        if(ascii!=(int)'X')
        strb.append((char)ascii);
      }
      while(ascii!=(int)'X');

      //εάν έχουμε επιτυχής αποστολή τότε στέλνεται '@'
      //για κάθε στοιχείο που λαμβάνεται γίνεται έλεγχος στον πίνακα xc
      //για αλλαγές τιμών
       if(strb.toString()!="")sr.xc[s.getLocalPort()-8000]=strb.toString();
      sr.tmp_access.access(sr.xc,sr.item);
      if(check==true)
            {
              os.write('@');
```

```
                }
                else
                {
                    os.write('#');
                }
        //υλοποίηση για κλείσιμο των αισθητήρων από το server
          os.flush();
          os.close();
          is.close();
          s.close();
      }
      // εξαίρεση διακοπής σύνδεσης
      catch(IOException e)
      {
          e.printStackTrace();
       //os
       try
          {
                    if(os!=null)os.close();
          }
                     catch(IOException mn)
          {
                    mn.printStackTrace();
          }
      //is
       try
          {
                    if(is!=null)is.close();
          }
                     catch(IOException mn)
          {
                    mn.printStackTrace();
          }
      //socket
          try
          {
                    if(s!=null)s.close();
          }
                     catch(IOException mn)
          {
                    mn.printStackTrace();
          }
    }
  }
  }

//**********
//server_initial
//**********
//η κλάση αυτή είναι υπεύθυνη για τη θύρα 6016
//αυτή αναλαμβάνει τη μετάδοση του αρχείου spec.nfo στο PDA στο οποίο εμπεριέχονται
//οι απαραίτητες πληροφορίες για αρχικοποίηση ενός νέου προβλήματος στο PDA
import java.net.*;
import java.io.*;

class server_initial implements Runnable
{   //Δηλώσεις μεταβλητών
   int port;
   public BufferedOutputStream bos = null;
   public DataOutputStream dos = null;
```

```
   public String file_name;
   ServerSocket server;
   Socket client;
   //μεταβλητή σημαφόρος ώστε να διακόπτεται το νήμα
   public volatile boolean m_bln=true;
   FileInputStream fis  ;
  BufferedInputStream bis;
  DataInputStream dis;
   //Κατασκευαστής
   server_initial(int port,String file_name)
   {
       this.file_name=file_name;
      this.port = port;
   }
//μέθοδος για την διακοπή του νήματος
   public void requestStop()
   {
           m_bln=false;
           try
           {
           server.close();
           }
           catch(IOException f)
           {
           }
   }//η μέθοδος του νήματος
   public void run()
   {
           server = null;
      client = null;
      fis=null;
      bis=null;
      dis=null;
       boolean repeat;
//δημιουργία του serversocket
      try
      {
         server = new ServerSocket(port);
      }
      catch(IOException e)
      {
         System.out.println("ERROR[1]-->"+e.toString());
      }
           while(m_bln==true)
      {
           System.out.println("trying to connect at...."+port);
        try
        {
          client = server.accept();
        }
        catch(IOException e)
        {
          System.out.println(port+" : "+e.toString());
          break;
        }
   try
   {//άνοιγμα του αρχείου από το οποίο διαβάζονται τα δεδομένα δηλαδή από το spec.nfo
                   FileInputStream fis = new FileInputStream(file_name);
                   BufferedInputStream bis = new BufferedInputStream(fis);
                   BufferedReader dis=new BufferedReader(new InputStreamReader(bis));
```

```
                System.out.println("file  opened");
                //το serversocket παραμένει πάντα ανοιχτό
        try
          {
           bos = new BufferedOutputStream(client.getOutputStream());
          }
        catch(IOException e)
          {
           System.out.println("ERROR[3]-->"+e.toString());
          }
        dos = new DataOutputStream(bos);
//στο σημείο αυτό έχει πραγματοποιηθεί η σύνδεση με πελάτη
        System.out.println("Connected!!!!!");
        repeat = true;
//επαναληπτικός βρόχος 6 στιγμών ώστε να διαβαστούν 6 τιμές από το αρχείο
        for(int g=0;g<7;g++)
        {
          String tmp=new String("");
           try
              {      //σε κάθε γραμμή που διαβάζεται προστίθεται και αλλαγή γραμμής
                        tmp=dis.readLine();
                        dos.writeBytes(tmp+'\n');
                        dos.flush();
                        System.out.println(tmp);
              }
          catch(IOException e)
              {
                      System.out.println("ERROR[3]-->"+e.toString());
                      try
                          {
                            dos.close();
                            dis.close();
                            client.close();
                          }
                          catch(IOException error){}
                     }
        }//τέλος ανάγνωσης πληροφοριών
                     System.out.println("end of file");
                     try
                       {
                         dos.close();
                         dis.close();
                         client.close();
                       }catch(IOException error){}
                   }//περίπτωση να μην υπάρχει το αρχείο spec.nfo
                   catch(FileNotFoundException e)
                       {
                              System.out.println("file not found : "+e.toString());
                       }}}}


//*****************
//server_pc_initial.java
//*****************
//η κεντρική κλάση του εξυπηρετητή-server
//αυτή καθορίζει το πλαίσιο της κατάστασης αρχικοποίησης
import javax.swing.*;
import javax.swing.text.*;
import javax.swing.event.*;
import java.awt.*;
```

```
import java.awt.Toolkit.*;
import java.awt.event.*;
import java.io.*;

//η κλάση δημιουργεί ένα JPanel και αντιμετωπίζει ενέργειες από το χρήστη
public class server_pc_initial extends JPanel implements ActionListener
{
	//Διαχειριστής Διάταξης
	//Διάταξη ευέλικτου πλέγματος
	GridBagLayout layout = new GridBagLayout();
	GridBagConstraints constraints =new GridBagConstraints();
	static JFrame frame;
	Font f=new Font("TimesRoman",Font.BOLD,12);
	Initial init=null ;
	boolean m_first=true;
	//Ετικέτες
	JLabel Ibldsi = new JLabel("Data Set Identifier:",JLabel.LEFT);
	JLabel Iblnoif = new JLabel("Number of Input Features:",JLabel.RIGHT);
	JLabel Ibltd = new JLabel("Training Data:",JLabel.LEFT);
	JLabel Iblnor = new JLabel("Number of Rules:",JLabel.LEFT);
	JLabel Iblnote = new JLabel("Number of Training Epochs:",JLabel.RIGHT);
	JLabel Iblra = new JLabel("Resubstitution Accuracy  (%) :",JLabel.LEFT);
	JLabel Iblre = new JLabel("Resubstitution Error :",JLabel.LEFT);
	//Πεδία κειμένου
	JTextField txtdsi = new JTextField("(blank)",40);
	JTextField txtnoif = new JTextField("0",40);
	JTextField txttd = new JTextField(40);
	JTextField txtnor = new JTextField("0",40);
	JTextField txtnote = new JTextField("0",40);
	JTextField txtra = new JTextField("0",40);
	JTextField txtre = new JTextField("0/0",40);
	//Κουμπί Browse
	JButton btnBrowse= new JButton("Browse");
	//Κλάση γραφικής απεικόνισης του συστήματος αρχείων
	JFileChooser fileChooser = new JFileChooser();
	//Κουμπιά Start, Switch
	JButton btnStart= new JButton("Start");
	JButton btnSwitch= new JButton("Switch");
	//Κατασκευαστής
	public server_pc_initial(String title)
	{
		setLayout(layout);
		constraints.fill=GridBagConstraints.BOTH;
		txtdsi.setFont(f);
		txtnoif.setFont(f);
		txttd.setFont(f);
		txtnor.setFont(f);
		txtnote.setFont(f);
		txtra.setFont(f);
		txtre.setFont(f);
		txtre.setEditable(false);
		txtre.setHorizontalAlignment(JTextField.CENTER);
		btnStart.setBackground(Color.gray);
		btnBrowse.setBackground(Color.gray);
		btnSwitch.setBackground(Color.gray);
		//στοιχεία πρώτης σειράς
		add(Ibldsi);
		add(txtdsi);
		add(Iblnoif);
		add(txtnoif);
```

```
build(constraints,0,0,1,1,100,100);
layout.setConstraints(Ibldsi,constraints);
build(constraints,1,0,1,1,100,100);
layout.setConstraints(txtdsi,constraints);
build(constraints,2,0,1,1,100,100);
layout.setConstraints(Iblnoif,constraints);
build(constraints,3,0,1,1,100,100);
layout.setConstraints(txtnoif,constraints);
//στοιχεία δεύτερης σειράς
add(Ibltd);
add(txttd);
add(btnBrowse);
btnBrowse.addActionListener(this);

build(constraints,0,1,1,1,100,100);
layout.setConstraints(Ibltd,constraints);

constraints.fill=GridBagConstraints.NONE;
build(constraints,1,1,1,1,100,100);
layout.setConstraints(btnBrowse,constraints);
constraints.fill=GridBagConstraints.BOTH;

build(constraints,2,1,2,1,100,100);
layout.setConstraints(txttd,constraints);

//στοιχεία τρίτης σειράς
add(Iblnor);
add(txtnor);
add(Iblnote);
add(txtnote);
build(constraints,0,2,1,1,100,100);
layout.setConstraints(Iblnor,constraints);
build(constraints,1,2,1,1,100,100);
layout.setConstraints(txtnor,constraints);
build(constraints,2,2,1,1,100,100);
layout.setConstraints(Iblnote,constraints);
build(constraints,3,2,1,1,100,100);
layout.setConstraints(txtnote,constraints);

//στοιχεία τέταρτης σειράς
add(Iblra);
add(txtra);

build(constraints,0,3,1,1,100,100);
layout.setConstraints(Iblra,constraints);
build(constraints,1,3,1,1,100,100);
layout.setConstraints(txtra,constraints);

//στοιχεία πέμπτης σειράς
add(Iblre);
add(txtre);
build(constraints,0,4,1,1,100,100);
layout.setConstraints(Iblre,constraints);
build(constraints,1,4,1,1,100,100);
layout.setConstraints(txtre,constraints);
JLabel empty=new JLabel("  ");
add(empty);
build(constraints,3,4,1,1,100,100);
layout.setConstraints(empty,constraints);
```

```
		//στοιχεία έκτης σειράς
		add(btnStart);
		btnStart.addActionListener(this);
		add(btnSwitch);
		btnSwitch.addActionListener(this);
		build(constraints,1,5,1,1,100,100);
		layout.setConstraints(btnStart,constraints);
		build(constraints,3,5,1,1,100,100);
		layout.setConstraints(btnSwitch,constraints);


		frame=new JFrame(title);
		frame.getContentPane().add(this,BorderLayout.CENTER);
		frame.setSize(700,200);
		frame.setVisible(true);
		//προσθήκη για να κλείνει το παράθυρο από το κουμπί στο πάνω δεξιά μέρος
		frame.addWindowListener
		(
			new WindowAdapter()
			{
				public void windowClosing(WindowEvent e)
				{
					System.exit(0);
				}
			}
		);
}
//μέθοδος κατασκευής των στοιχείων του ευέλικτου πλέγματος
public void build(GridBagConstraints gbc,int gx,int gy, int gw,int gh,int wx,int wy)
{
		gbc.gridx=gx;
		gbc.gridy=gy;
		gbc.gridwidth=gw;
		gbc.gridheight=gh;
		gbc.weightx=wx;
		gbc.weighty=wy;
}
//μέθοδος που αντιμετωπίζει τα συμβάντα από το χρήστη
public void actionPerformed(ActionEvent e)
{
			setTextStr(e,txttd,btnBrowse);

			if (e.getSource()==btnStart)
			{
				if (txtdsi.getText().length()*
					txtnoif.getText().length()*
					txttd.getText().length()*
					txtnor.getText().length()*
					txtnote.getText().length()*
					txtra.getText().length()==0)
				{
				//εάν τα πεδία δεν έχουν όλα τιμές τότε εμφανίζεται μήνυμα
				JFrame frame = new JFrame("ERROR");
				JPanel panel=new JPanel();
				JLabel lab=new JLabel("!!!! Not every blank is filled");
				panel.add(lab);
				frame.getContentPane().add(panel,BorderLayout.CENTER);
				frame.setSize(200,50);
				frame.setLocation(150,150);
				frame.show();
```

```
                                        }
                                        else
                                        {
                                         if(btnStart.getText()=="Start")
                                          {
//το κουμπί Start γίνεται Stop
                                            btnStart.setText("Stop");
//εάν ο χρήστης πατήσει Start τότε ξεκινά η εκπαίδευση του προβλήματος σύμφωνα με τα κενά που
//έχει συμπληρώσει
                                            if(m_first==true)
                                            {
                init=new Initial(txtdsi.getText(),Integer.parseInt(txtnoif.getText()), txttd.getText(),
                                                        Integer.parseInt(txtnor.getText()),
                                                        Integer.parseInt(txtnote.getText()),
                                                        Double.parseDouble(txtra.getText()),
                                                        txtre,this.btnStart,this.btnSwitch);
                                                                m_first=false;
                                                }
                                                init.start(Integer.parseInt(txtnor.getText()),
                                                        Integer.parseInt(txtnote.getText()),
                                                        Double.parseDouble(txtra.getText()));
//τα πεδία αναγνωριστικού προβλήματος, αριθμός χαρακτηριστικών , σύνολο εκπαίδευσης
//και τα κουμπιά Browse,Switch είναι ανενεργά
                                                txtdsi.setEditable(false);
                                                txtnoif.setEditable(false);
                                                txttd.setEditable(false);
                                                btnBrowse.setEnabled(false);
                                                btnSwitch.setEnabled(false);
                                }
                                                else
                                                {
                                                        btnSwitch.setEnabled(true);
                                                        init.stop();
                                                        btnStart.setText("Start");
                                                        btnBrowse.setEnabled(false);
                                                        txtdsi.setEditable(false);
                                                        txtnoif.setEditable(false);
                                                        txttd.setEditable(false);
                                                }
                                        }
                        }
//εάν ο χρήστης πατήσει το κουμπί Switch τότε ο εξυπηρετητής ετοιμάζεται να δεχθεί αιτήσεις από
//πελάτη
                        if (e.getSource()==btnSwitch)
                                {
//η κλάση Update περιμένει για δεδομένα από το PDA  με τα οποία θα πραγματοποιήσει
//επανεκπαίδευση
                                        Update u=new Update();
                                        u.start();
//η κλάση server_rs ανοίγει serversocket στις θύρες από 6000 έως 6016 και περιμένει αιτήσεις από
//πελάτες
                                        server_rs tmp=new server_rs(txtdsi.getText());
                                        tmp.start();
//η κλάση UpdaFrame δημιουργεί το παράθυρο κατάστασης ενημέρωσης
                                        UpdateFrame mf = new UpdateFrame("WIFuNeCS/-
                                        /Updating",tmp,u);
//μεταφέρονται οι απαραίτητες τιμές από το ένα παράθυρο στο άλλο
                                        mf.txtdsi.setText(String.valueOf(txtdsi.getText()));
                                        mf.txtnoif.setText(txtnoif.getText());
                                        mf.txttd.setText(txttd.getText());
```

```
                                        mf.txtnor.setText(txtnor.getText());
                                        mf.txtnote.setText(txtnote.getText());
                                        mf.txtra.setText(txtra.getText());
                                        mf.txtre.setText(txtre.getText());
                                        this.frame.dispose();
                                }
        }
//μέθοδος που καλείται αν πατηθεί το κουμπί Browse, οπότε πρέπει να εμφανιστεί το σύστημα
//αρχείων για να διαλέξει ο χρήστης αρχείο
        public void setTextStr(ActionEvent e,JTextField txt,JButton btn)
        {
                if (e.getSource()==btn)
                {
                        int returnVal=fileChooser.showOpenDialog(this);
                        if (returnVal==JFileChooser.APPROVE_OPTION)
                        {
                                File file =fileChooser.getSelectedFile();
                                String str=file.getPath();
//το όνομα του αρχείου που διαλέγεται , τοποθετείται στο πεδίο συνόλου εκπαίδευσης
                                txt.setText(str);
                        }
                }
        }
//η κύρια μέθοδος που εκτελείται με την εφαρμογή
//δεν δέχεται ορίσματα
        public static void main(String srgs[])
        {
                String title="WIFuNECS";
                String tmp=title+"/-/Initialization";
                server_pc_initial example = new server_pc_initial(tmp);
        }
}


//*****************
//server_receiving.java
//*****************
//η κλάση αυτή ανοίγει serversocket στη θύρα 6000 και περιμένει το PDA να συνδεθεί και να στείλει
//δεδομένα που έχουν αποθηκευθεί από προσομοίωση
import java.net.*;
import java.io.*;


class server_receiving implements Runnable
{
//Δηλώσεις μεταβλητών
   int port;
   public String file_name;
   public volatile boolean m_bln=true;
   ServerSocket server;
   Socket client;
   BufferedInputStream bis;
   DataInputStream dis;
        //κατασκευαστής
   server_receiving(int port,String file_name)
   {
      this.file_name=file_name;
      this.port = port;
   }
   //μέθοδος για να διακόπτεται το νήμα
```

```
    public void requestStop()
    {
            m_bln=false;
            try
            {
            server.close();
            }
            catch(IOException f)
            {}
    }
//μέθοδος που εκτελείται παράλληλα
    public void run()
    {
       server = null;
       client = null;
       bis = null;
       dis = null;
       boolean repeat;
//δημιουργία ενός serversocket
       try
       {
          server = new ServerSocket(port);
       }
       catch(IOException e)
       {
          System.out.println("ERROR[1]-->"+e.toString());
       }
//ο server περιμένει κάποιο πελάτη-PDA να συνδεθεί στη θύρα 6000
       while(m_bln==true)
       {
          System.out.println("trying to connect at...."+port);
          try
          {
             client = server.accept();
          }
          catch(IOException e)
          {
             System.out.println(port+" : "+e.toString());
             break;
          }
          try
          {
             bis = new BufferedInputStream(client.getInputStream());
          }
          catch(IOException e)
          {
             System.out.println("ERROR[3]-->"+e.toString());
          }

          dis = new DataInputStream(bis);
          double tmp=0;
          try
          {
//το αρχείο στο οποίο αποθηκεύονται τα δεδομένα έχει το πρόθεμα temp και μετά το αναγνωριστικό
//του προβλήματος
                                    FileOutputStream fos = new FileOutputStream("temp"+file_name);
                                    BufferedOutputStream bos = new BufferedOutputStream(fos);
                                    DataOutputStream dos = new DataOutputStream(bos);
                                    repeat=true;
                                    while(repeat==true)
```

```
                                {
                            try
                            {
                              tmp=dis.readDouble();
                            }
//εάν τελειώσει η μετάδοση προκύπτει κατάλληλη εξαίρεση
                            catch(EOFException e)
                                  {
                                    System.out.println("file transfered");
                               System.out.println("ERROR[4]-->"+e.toString());
                               try
                               {
                                  dos.close();
                                  dis.close();
                                  client.close();
                               }
                               catch(IOException error)
                               {}
//αφού τελειώσει η μετάδοση τότε το αρχείο μετονομάζεται σε
//new +Αναγνωριστικό Προβλήματος
                                  File fl=new File("temp"+file_name);
                           if(fl.length()!=0)  {
                           if(new File("new"+file_name).exists())
                           new File("new"+file_name).delete();
                           fl.renameTo(new File("new"+file_name));
                                                        }
                                  }
                                  catch(IOException iods)
                                  {
                                              try
                                              {
                                              dos.close();
                                                           }
                                                        catch(IOException ioe)
                                                        {}
                                              System.out.println("error connection");

                        //εάν διακοπεί η σύνδεση τότε πρέπει να διαγραφεί το αρχείο που έχει αποθηκευτεί
                        //μέχρι στιγμής
                                             try
                                             {
                                              fos = new FileOutputStream("temp"+file_name);
                                               bos = new BufferedOutputStream(fos);
                                              dos = new DataOutputStream(bos);
                                                try
                                                {
                                                dos.close();
                                                dis.close();
                                                client.close();
                                                System.out.println("file erased..");
                                                if(new File("temp"+file_name).exists())
                                                new File("temp"+file_name).delete();
                                                }
                                              catch(IOException ioe)
                                                 {}
                                              }
                                                catch(FileNotFoundException fnfe) {}
                                  }
                           dos.writeDouble(tmp);
                           dos.flush();
```

```
                }
                        try
                                {
                                dos.close();
                                dis.close();
                                client.close();
                                }
                        catch(IOException ioe){}
        }
//εξαίρεση διακοπής σύνδεσης
        catch(IOException e)
                {
                  System.out.println("connection closed "+port+" "+e.toString());
                }


      }
      }
}

//***********
//server_rs.java
//***********
//κλάση η οποία δημιουργεί νήματα
//τα νήματα αντιστοιχούν ένα σε κάθε θύρα και ελέγχουν για αιτήσεις από πελάτες
//πρέπει να επιτρέπεται παραλληλισμός ,επειδή έχουμε εξυπηρετητή ο οποίος πρέπει να αντιμετωπίζει
//πολλούς πελάτες
public class server_rs
{
                public int arg=6000;
                public server_receiving s0;
                public server_sending s1;
                public server_sending s2;
                public server_sending s3;
                public server_sending s4;
                public server_sending s5;
                public server_sending s6;
                public server_sending s7;
                public server_sending s8;
                public server_sending s9;
                public server_sending s10;
                public server_sending s11;
                public server_sending s12;
                public server_sending s13;
                public server_sending s14;
                public server_sending s15;
                public server_initial s16;
                public Thread t0;
                public Thread t1;
                public Thread t2;
                public Thread t3;
                public Thread t4;
                public Thread t5;
                public Thread t6;
                public Thread t7;
                public Thread t8;
                public Thread t9;
                public Thread t10;
                public Thread t11;
                public Thread t12;
```

```
                public Thread t13;
                public Thread t14;
                public Thread t15;
                public Thread t16;
//κατασκευαστής
        server_rs(String file_name)
        {
//δημιουργούνται τα κατάλληλα ομότυπα για κάθε θύρα που αντιστοιχούν στα νήματα για
//παραλληλισμό
          s0 = new server_receiving(arg,file_name+".dat");
          t0 = new Thread(s0);
          s1 = new server_sending(arg+1,file_name+".axs");
          t1 = new Thread(s1);
          s2 = new server_sending(arg+2,file_name+".awc");
          t2 = new Thread(s2);
          s3 = new server_sending(arg+3,file_name+".aws");
          t3 = new Thread(s3);
          s4 = new server_sending(arg+4,file_name+".avc");
           t4 = new Thread(s4);
          s5 = new server_sending(arg+5,file_name+".avs");
          t5 = new Thread(s5);
          s6 = new server_sending(arg+6,file_name+".bxs");
          t6 = new Thread(s6);
         s7 = new server_sending(arg+7,file_name+".bwc");
         t7 = new Thread(s7);
         s8 = new server_sending(arg+8,file_name+".bws");
         t8 = new Thread(s8);
        s9 = new server_sending(arg+9,file_name+".bvc");
        t9 = new Thread(s9);
        s10 = new server_sending(arg+10,file_name+".bvs");
        t10 = new Thread(s10);
       s11 = new server_sending(arg+11,file_name+".cxs");
        t11 = new Thread(s11);
        s12 = new server_sending(arg+12,file_name+".cwc");
        t12 = new Thread(s12);
        s13 = new server_sending(arg+13,file_name+".cws");
        t13 = new Thread(s13);
        s14 = new server_sending(arg+14,file_name+".cvc");
        t14 = new Thread(s14);
        s15 = new server_sending(arg+15,file_name+".cvs");
        t15 = new Thread(s15);
        s16 = new server_initial(arg+16,"spec.nfo");
        t16 = new Thread(s16);
        }
        //η μέθοδος που εκκινεί τα νήματα
        public  void start()
        {
                System.out.println("Receiving data  at 6000");
                System.out.println("Sending Weights at 6001-15");
                System.out.println("Sending spec.nfo at 6016");
                 t0.start();
                 t1.start();
                 t2.start();
                 t3.start();
                 t4.start();
                 t5.start();
                 t6.start();
                 t7.start();
                 t8.start();
                 t9.start();
```

```
                t10.start();
                t11.start();
                t12.start();
                t13.start();
                t14.start();
                t15.start();
                t16.start();
        }
//μέθοδος διακοπής των νημάτων που έχουν δημιουργηθεί και ελέγχουν τις αιτήσεις σε κάθε θύρα
        public  void requestStop()
        {
                s0.requestStop();
                s1.requestStop();
                s2.requestStop();
                s3.requestStop();
                s4.requestStop();
                s5.requestStop();
                s6.requestStop();
                s7.requestStop();
                s8.requestStop();
                s9.requestStop();
                s10.requestStop();
                s11.requestStop();
                s12.requestStop();
                s13.requestStop();
                s14.requestStop();
                s15.requestStop();
                s16.requestStop();
        }
}



//***************
//server_sending.java
//***************
//η κλάση αυτή είναι υπεύθυνη για τη μετάδοση των αρχείων των βαρών στο PDA
//ακούει αιτήσεις στις θύρες από 6001 έως 6015
import java.net.*;
import java.io.*;

class server_sending implements Runnable
{
//Δηλώσεις μεταβλητών
   int port;
   public BufferedOutputStream bos = null;
   public DataOutputStream dos = null;
   public String file_name;
   public volatile boolean m_bln=true;
   ServerSocket server;
   Socket client;
   FileInputStream fis  ;
   BufferedInputStream bis;
   DataInputStream dis;
   //κατασκευαστής
   server_sending(int port,String file_name)
   {
      this.file_name=file_name;
      this.port = port;
   }
```

```
   //μέθοδος διακοπής νήματος
   public void requestStop()
   {
          m_bln=false;
          try
          {
          server.close();
          }
          catch(IOException f)
          {
          }
   }
//μέθοδος νήματος που επιτρέπει τον παραλληλισμό
   public void run()
   {
      server = null;
      client = null;
      fis=null;
      bis=null;
      dis=null;
       boolean repeat;
           //δημιουργία του serversocket
      try
      {
         server = new ServerSocket(port);

      }
      catch(IOException e)
      {
         System.out.println("ERROR[1]-->"+e.toString());

      }
   while(m_bln==true)
      {
          System.out.println("trying to connect at...."+port);
         try
        {
          client = server.accept();
          System.out.println("server.accept();");
        }
        catch(IOException e)
        {
          System.out.println(port+" : "+e.toString());
          break;
        }
         try
         {
//άνοιγμα του αρχείου βάρους το οποίο πρέπει να μεταδοθεί στο PDA
                    FileInputStream fis = new FileInputStream(file_name);
                    BufferedInputStream bis = new BufferedInputStream(fis);
                    DataInputStream dis = new DataInputStream(bis);
                    System.out.println("file  opened");
//ο server περιμένει για αίτηση
        try
        {
          bos = new BufferedOutputStream(client.getOutputStream());
        }
        catch(IOException e)
        {
          System.out.println("ERROR[3]-->"+e.toString());
```

```
            break;
        }
        dos = new DataOutputStream(bos);
        System.out.println("dos = new DataOutputStream(bos);");
//στο σημείο αυτό έχει πραγματοποιηθεί η σύνδεση
        repeat = true;
        while(repeat)
        {
          String tmp=new String("");
                try
                 {
                  try
                    {
                        tmp=String.valueOf(dis.readDouble());
                        dos.writeDouble(Double.parseDouble(tmp));
                        dos.flush();
                        System.out.println(tmp);
                    }
                  //μόλις τελειώσει η ανάγνωση από το αρχείο εγείρεται εξαίρεση EOFException
                  catch(EOFException eof)
                  {
                                System.out.println("end of file");
                                repeat = false;
                                try
                                {
                                 dos.close();
                                 dis.close();
                                client.close();
                                    }
                            catch(IOException error){}
                            repeat = false;
                        }

                  }
                      catch(IOException e)
                        {
                            System.out.println("ERROR[3]-->"+e.toString());
                            try
                              {
                                 dos.close();
                                 dis.close();
                                 client.close();
                              }
                              catch(IOException error)    {}
                              repeat = false;
                        }
        }               }
//περίπτωση στην οποία δεν βρέθηκε το αρχείο
                      catch(FileNotFoundException e)
                            {
                                System.out.println("file not found : "+e.toString());
                            }
                  }
          }
}
```

```
//***********
//ServerII.java
//***********
//η κλάση αυτή υλοποιεί την σύγχρονη μετάδοση δεδομένων από την πλευρά του PDA
import java.net.*;
import java.util.*;
import java.io.*;
import java.awt.*;

public class ServerII extends Thread
{
	//Δηλώσεις μεταβλητών
	//ο πίνακας στο οποίο αποθηκεύονται τα δεδομένα από τους αισθητήρες
	public static String xc[];
	public Socket clientSocket;
	public ServerSocket serverSocket;
	//η μεταβλητή για να σταματά η σύνδεση PDA-αισθητήρες
	public volatile boolean m_chk;
	//η μεταβλητή tmp_access χρησιμοποιείται για να ελέγχει τις προσπελάσεις στο xc[]
	public Result tmp_access;
	public String item;
	public int inp;
	public TextField features;
	public String port;
	public int feat=0;
	public EchoThreadII echo=null;
	//κατασκευαστής
	public ServerII(String port,String name_item,int input,TextField feat)
	{
		super(port);
		features = feat;
		this.port=port;
		this.m_chk=true;

		this.inp=input;
		this.item=name_item;
		xc=new String[input+1];
		//αρχικοποίηση του πίνακα με μηδενικές τιμές
		for(int j=0;j<xc.length;j++)
		{
			xc[j]="0";
		}
		tmp_access=new Result(xc,features);
		//αρχικοποίηση του serversocket
		try
		{
		serverSocket = new ServerSocket(8000);
		}
		catch(IOException e)
		{
		e.printStackTrace();
		}
	}
	//μέθοδος που σταματά το τρέχων νήμα
	public void requestStop()
	 {
	tmp_access=new Result(xc,features);
	m_chk=false;
	 }
	   //μέθοδος υλοποίησης του νήματος
```

```
 public void run()
 {
   //αέναος βρόχος, για να αντιμετωπιστεί το bug της Jeode
 while(true)
 {
clientSocket=null;
try
{
        clientSocket = serverSocket.accept();
}
catch(IOException e)
{
        e.printStackTrace();
        continue;
}
//η μεταβλητή m_chk καθορίζετε από τις επιλογές του χρήστη
if(m_chk==true)
{
        echo=new EchoThreadII(clientSocket,this,feat,inp);
        echo.start();
        feat++;
}
}
   }
}

//**************
//EchoThreadII.java
//**************
//η κλάση υλοποιεί το νήμα που καλείται για κάθε αίτηση πελάτη
   class EchoThreadII extends Thread
{
        public static boolean check=true;;
        public int features;
        Socket s;
        ServerII sr;
        int inp;
        InputStream is;
        OutputStream os;
        //κατασκευαστής
        EchoThreadII(Socket s,ServerII sr,int feat,int input)
        {
                features=feat%input+1;
                this.s=s;
                this.sr=sr;
                this.inp=input;
        }
        //μέθοδος νήματος
        public void run()
        {
      int ascii;
      StringBuffer strb=new StringBuffer();
      Random r = new Random();
      try
      {
        is = s.getInputStream();
        os = s.getOutputStream();
        //σύμβαση οι τιμές να τελειώνουν σε 'X'
        do
        {
```

```
                    ascii = is.read();
                      if(ascii!=(int)'X')
                      strb.append((char)ascii);
                    }
                    while(ascii!=(int)'X');
                  //μετά από σωστή αποστολή
                  //στέλνει @ στο server
                    System.out.println(strb.toString());
                  sr.xc[features]=strb.toString();
                  //για κάθε στοιχείο που λαμβάνεται γίνεται έλεγχος στον πίνακα xc
                  //για αλλαγές τιμών
                  if(features==inp)sr.tmp_access.access(sr.xc,sr.item);
                 //υλοποίηση για κλείσιμο των αισθητήρων από το server
                 if(check==true)
                 {
                    os.write('@');
                 }
                 else
                 {
                    os.write('#');
                 }
                  os.flush();
                  os.close();
                  is.close();
                  s.close();
               }
               // εξαίρεση διακοπής σύνδεσης
               catch(IOException e)
               {
                        e.printStackTrace();
                        //os
                        try
                                {
                                        if(os!=null)os.close();
                                }
                                catch(IOException mn)
                        {
                                mn.printStackTrace();
                        }
                        //is
                        try
                                {
                                        if(is!=null)is.close();
                                }
                                catch(IOException mn)
                        {
                                mn.printStackTrace();
                        }
                        //socket
                        try
                                {
                                        if(s!=null)s.close();
                                }
                                catch(IOException mn)
                        {
                                mn.printStackTrace();
                        }
               }
        }
}
```

```
//**************
//UpdateFrame.java
//**************
//η κλάση αυτή δημιουργεί το παράθυρο που αντιστοιχεί στην κατάσταση ενημέρωσης
//υπάρχει ένα κινούμενο γραφικό καθώς και ένα κουμπί  Switch για μετάβαση στην κατάσταση
//αρχικοποίησης
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.text.*;
import javax.swing.event.*;

class UpdateFrame extends Frame implements  ActionListener
{
            public gif icon;
            public JLabel empty;
            public JButton btnSwitch;
            //τα πεδία κειμένου δεν παρουσιάζονται στο πλαίσιο, αλλά είναι απαραίτητα για την
            //μεταφορά των πληροφοριών ανάμεσα στα παράθυρα κατάστασης ενημέρωσης και
            //αρχικοποίησης
            public static JTextField txtdsi = new JTextField("(blank)",40);
            public static JTextField txtnoif = new JTextField("0",40);
            public static JTextField txttd = new JTextField(40);
            public static JTextField txtnor = new JTextField("0",40);
            public static JTextField txtnote = new JTextField("0",40);
            public static JTextField txtra = new JTextField("0",40);
            public static JTextField txtre = new JTextField("0/0",40);
            //αντικείμενο της κλάσης που αντιστοιχεί στην κατάσταση αρχικοποίησης
            public server_rs rs;
            public Update u;
   UpdateFrame(String title,server_rs rs,Update u)
   {
      super(title);
//Διαχειριστής διάταξης
//διάταξη ευέλικτου πλέγματος
      GridBagLayout gbl = new GridBagLayout();
      GridBagConstraints gbc = new GridBagConstraints();
      setLayout(gbl);

      this.rs=rs;
      this.u=u;

//στοιχεία πρώτης σειράς
      gbc.fill = GridBagConstraints.BOTH;
      empty=new JLabel("UPDATING MODE");
                  add(empty);
      addConstraints(gbl,empty,gbc,0,0,3,1,1,1,GridBagConstraints.CENTER);
      empty.setHorizontalAlignment(JLabel.CENTER);
//στοιχεία δεύτερης σειράς
//κινούμενο γραφικό , αντικείμενο κλάσης gif()
      icon = new gif();
      addConstraints(gbl,icon,gbc,0,1,3,1,2,2,GridBagConstraints.NORTHWEST);
      add(icon);
      icon.start();

//στοιχεία τρίτης σειράς
```

```
        gbc.fill = GridBagConstraints.NONE;
        btnSwitch=new JButton("Switch");
        btnSwitch.addActionListener(this);
       addConstraints(gbl,btnSwitch,gbc,2,2,1,1,0,0,GridBagConstraints.CENTER);
       add(btnSwitch);
       setBackground(Color.lightGray);
       setSize(305,220);
       setLocation(303,200);
       show();
       addWindowListener
                      (
                              new WindowAdapter()
                              {
                                      public void windowClosing(WindowEvent e)
                                      {
                                              System.exit(0);
                                      }
                              }
                      );
   }
//μέθοδος που αντιμετωπίζει τα συμβάντα από τον χρήστη
   public void actionPerformed(ActionEvent e)
          {
                      if (e.getSource()==btnSwitch)
                              {
                                      String title="WiFuNECS";
                                      String tmp=title+"/-/Initialization";
//εάν ο χρήστης πατήσει το κουμπί  Switch τότε επιστρέφουμε σε κατάσταση αρχικοποίησης
                                      server_pc_initial example = new server_pc_initial(tmp);
//στο νέο παράθυρο που δημιουργείται μεταφέρονται και οι απαραίτητες πληροφορίες
                                      example.txtdsi.setText(txtdsi.getText());
                                      example.txtnoif.setText(txtnoif.getText());
                                      example.txttd.setText(txttd.getText());
                                      example.txtnor.setText(txtnor.getText());
                                      example.txtnote.setText(txtnote.getText());
                                      example.txtra.setText(txtra.getText());
                                      example.txtre.setText(txtre.getText());
                                      example.txtdsi.setEditable(false);
                                      example.btnBrowse.setEnabled(false);
                                      example.txtnoif.setEditable(false);
                                      example.txttd.setEditable(false);
//για να αποδεσμεύσουμε αυτό το παράθυρο , πρώτα πρέπει να σταματήσουν τα serversocket να ακούν
// στις θύρες και επίσης να σταματήσει ο server να εξετάζει αν  το PDA έχει φέρει νέα δεδομένα να
//επανεκπαίδευση.
                                      this.dispose();
                                      this.rs.requestStop();
                                      this.rs=null;
                                      this.u.stop();

                              }
          }
//μέθοδος που κατασκευάζει τα  στοιχεία του πλαισίου και τα τοποθετεί στο ευέλικτο πλέγμα
   public void addConstraints(GridBagLayout gbl,Component c,GridBagConstraints gbc,
                      int gx,int gy,int gw,int gh,int wx,int wy,int ar)
   {
      gbc.gridx = gx;
      gbc.gridy = gy;
      gbc.gridwidth = gw;
      gbc.gridheight = gh;
      gbc.weightx = wx;
```

```
        gbc.weighty = wy;
        gbc.anchor = ar;
        gbl.setConstraints(c,gbc);
    }
}
```