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

Αυτο-οργανούμενοι χάρτες και συμβολική αναπαράσταση γνώσης με μορφή κανόνων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στυλιανός Α. Μοδές

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

Αθήνα, Ιούλιος 2006

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

Αυτο-οργανούμενοι χάρτες και συμβολική αναπαράσταση γνώσης με μορφή κανόνων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στυλιανός Α. Μοδές

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Ιουλίου 2006.

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

Αθήνα, Ιούλιος 2006

.....
Στυλιανός Α. Μοδές

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

Περίληψη

Οι αυτο-οργανούμενοι χάρτες (SOM) αποτελούν ένα πολύ δημοφιλές μοντέλο ανάλυσης δεδομένων που χρησιμοποιείται σε εφαρμογές εξόρυξης δεδομένων. Οι αυτο-οργανούμενοι χάρτες έχουν την ιδιότητα να διατηρούν τις τοπολογικές σχέσεις που ισχύουν στο χώρο εισόδου. Έτσι επιτυγχάνεται μείωση των διαστάσεων των δεδομένων και διευκολύνεται κατά πολύ η ανάλυση των δεδομένων. Επίσης, οι αυτο-οργανούμενοι χάρτες έχουν την ιδιότητα να κατηγοριοποιούν τα δεδομένα και προσφέρουν τη δυνατότητα οπτικοποίησης δεδομένων που αλλιώς θα ήταν δύσκολο να μελετηθούν.

Οι αυτο-οργανούμενοι χάρτες, όπως και όλοι οι τύποι νευρωνικών δικτύων, δεν προσφέρουν κάποια πληροφορία σχετικά με τη γνώση που αποκτούν κατά την εκπαίδευση τους και συνήθως χρησιμοποιούνται ως "μαύρα κουτιά" χωρίς να εξετάζεται καθόλου ποιες πληροφορίες υπάρχουν τελικά στο εσωτερικό τους. Σε ορισμένες εφαρμογές εξόρυξης δεδομένων όμως αυτό δεν είναι αρκετό. Η αναπαράσταση της γνώσης που βρίσκεται ενσωματωμένη σε ένα νευρωνικό δίκτυο με μορφή λογικών κανόνων είναι μεγάλης σημασίας γιατί έτσι μπορεί να γίνει εύκολα κατανοητή από τους ανθρώπους και να αξιοποιηθεί από τους αναλυτές των δεδομένων. Παρόλη την εφαρμογή και χρησιμότητα των αυτο-οργανούμενων χαρτών στην εξόρυξη δεδομένων η έρευνα για την εξαγωγή γνώσης από νευρωνικά δίκτυα έχει επικεντρωθεί κυρίως σε δίκτυα πρόσθιας τροφοδότησης.

Το αντικείμενο της διπλωματικής εργασίας είναι η μελέτη μεθόδων εξαγωγής συμβολικής γνώσης σε μορφή κανόνων από αυτο-οργανούμενους χάρτες. Στα πλαίσια της εργασίας υλοποιήθηκαν δύο μεθοδολογίες που εκμεταλλεύονται την κατηγοριοποίηση που κάνει ένας αυτο-οργανούμενος χάρτης για την παραγωγή κανόνων. Η πρώτη μέθοδος αναζητά συσχετίσεις ανάμεσα στο SOM και τους χάρτες που δημιουργούνται για κάθε μεταβλητή από τα διανύσματα βαρών των νευρώνων και οι συσχετίσεις αυτές μετατρέπονται σε κανόνες. Στη δεύτερη μέθοδο γίνεται ομαδοποίηση των νευρώνων του χάρτη και για κάθε ομάδα δημιουργούνται κανόνες βασισμένοι στις στατιστικές ιδιότητες των νευρώνων που ανήκουν στην ομάδα αυτή.

Οι δύο μέθοδοι εφαρμόστηκαν σε τρία πραγματικά σύνολα δεδομένων ώστε να είναι δυνατή η αξιολόγηση και η σύγκριση των αποτελεσμάτων τους. Επίσης, αναπτύχθηκαν μέθοδοι αυτόματου συνδυασμού και αξιολόγησης της εκφραστικής δύναμης των κανόνων. Από τα αποτέλεσμα των εφαρμογών αποδείχθηκε ότι η εξαγωγή κανόνων από αυτο-οργανούμενους χάρτες είναι εφικτή καθώς παράγονται μικρά και απλά σύνολα κανόνων που περιγράφουν τα δεδομένα με πολύ ικανοποιητική ακρίβεια.

Λέξεις κλειδιά: αυτο-οργανούμενοι χάρτες, εξαγωγή κανόνων, αναπαράσταση γνώσης, εξόρυξη δεδομένων, κατηγοριοποίηση

Abstract

Self-organizing maps (SOM) are widely used for analyzing data in data mining applications. The SOM performs a topology-preserving mapping of the input data to the output units. This enables dimensionality reduction of the input data and facilitates any further data analysis. The SOM also performs data classification and is suitable for visualizing data that would be otherwise difficult to interpret.

Self-organizing maps, as every class of neural networks, do not offer any insight into the knowledge they acquire during training and are usually used as "black boxes" without any interest for the knowledge embedded in their structure. In many applications black box prediction is not satisfactory. Representing knowledge embedded in neural networks in the form of symbolic rules is very important because they can be easily understood by humans and utilized by domain experts. Despite the usefulness of self-organizing maps in data mining, research on extracting knowledge from neural networks has mainly focused on feedforward neural networks.

The subject of this diploma thesis is symbolic rule extraction methods from self-organizing maps. Two different methods that make use of the classification performed by a self-organizing map to create rules have been implemented. The first method searches for relations between the SOM and component matrices and converts these relations into symbolic rules. In the second method, the SOM is clustered and rules are formed for each cluster based on statistical measures.

These two methods where applied to three different real-world datasets in order to evaluate and compare their results. Techniques for automated combination of rules and evaluation of their expressive power were also implemented. The experimental results showed that rule extraction from self-organizing maps is feasible as small and simple rule sets are produced that can describe the data with high accuracy.

Keywords: Self-organizing maps, rule extraction, knowledge representation, data mining, classification

Ευχαριστίες

Θα επιθυμούσα να εκφράσω τις θερμές μου ευχαριστίες στον υπεύθυνο για την εκπόνηση της εργασίας καθηγητή κ. Ανδρέα-Γεώργιο Σταφυλοπάτη για βοήθεια και την καθοδήγησή που προσέφερε, καθώς και στον υποψήφιο διδάκτορα κ. Χρήστο Πατερίτσα για τη σημαντική βοήθεια και συνεργασία για την όσο το δυνατό καλύτερη προσέγγιση του αντικειμένου που πραγματεύεται η παρούσα εργασία.

Περιεχόμενα

Ευρετήριο σχημάτων

Ευρετήριο πινάκων

1

Εισαγωγή

Το κείμενο αυτό αποτελεί τη διπλωματική εργασία με τίτλο "Αυτο-οργανούμενοι χάρτες και συμβολική αναπαράσταση γνώσης με μορφή κανόνων" η οποία πραγματοποιήθηκε κατά την διάρκεια του ακαδημαϊκού έτους 2005-2006 από το φοιτητή Στυλιανό Μοδέ του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Η εργασία αυτή έγινε στο Εργαστήριο Ευφυών Υπολογιστικών Συστημάτων υπό την επίβλεψη του καθηγητή κ. Ανδρέα-Γεώργιου Σταφυλοπάτη και με τη βοήθεια και καθοδήγηση του υποψήφιου διδάκτορα κ. Χρήστου Πατερίτσα.

1.1 Στόχοι της εργασίας

Η εργασία αυτή μελετάει τη χρήση των αυτο-οργανούμενων χαρτών σε εφαρμογές εξόρυξης δεδομένων και ειδικότερα τις δυνατότητες για την εξαγωγή γνώσης από αυτο-οργανούμενους χάρτες και τη συμβολική αναπαράσταση της με μορφή λογικών κανόνων. Οι στόχοι της εργασίας είναι:

- Μελέτη μεθόδων για την εξαγωγή γνώσης από αυτο-οργανούμενους χάρτες.
- Δημιουργία παραλλαγών και συνδυασμών των μεθόδων.
- Αναπαράσταση γνώσης με μορφή συμβολικών κανόνων.
- Τρόποι αξιολόγησης ακρίβειας και χρησιμότητας κανόνων.
- Υλοποίηση των μεθόδων στη γλώσσα προγραμματισμού Matlab®
- Εφαρμογή σε πραγματικά σύνολα δεδομένων ώστε να αξιολογηθούν και να συγκριθούν τα αποτελέσματα τους.

1.2 Οργάνωση του κειμένου

Η εργασία οργανώνεται στα παρακάτω κεφάλαια:

- Κεφάλαιο 1: Εισαγωγή όπου αναφέρονται οι στόχοι της εργασίας και η οργάνωση των κεφαλαίων.
- Κεφάλαιο 2: Γενικές αρχές της τεχνητής νοημοσύνης και των νευρωνικών δικτύων.
- Κεφάλαιο 3: Εκπαίδευση και τρόπος λειτουργίας των αυτο-οργανούμενων χαρτών και τεχνικές ομαδοποίησης δεδομένων που μπορούν να εφαρμοστούν σε αυτο-οργανούμενους χάρτες.
- Κεφάλαιο 4: Προτασιακή λογική και τρόποι αξιολόγησης συμβολικών κανόνων. Χρησιμότητα της αναπαράστασης γνώσης με μορφή κανόνων και εφαρμογές.
- Κεφάλαιο 5: Μέθοδος 1 για εξαγωγή κανόνων με βάση τον U-matrix.
- Κεφάλαιο 6: Μέθοδος 2 για εξαγωγή κανόνων με βάση στατιστικές ιδιότητες του SOM.
- Κεφάλαιο 7: Πειραματικά αποτελέσματα από την εφαρμογή των μεθόδων σε τρία διαφορετικά σύνολα δεδομένων και τελική αξιολόγηση των μεθόδων.
- Παράρτημα: Πλήρης πηγαίος κώδικας της εφαρμογής

2

Νευρωνικά δίκτυα

2.1 Τεχνητή νοημοσύνη

Κατά καιρούς έχουν διατυπωθεί διάφοροι ορισμοί της τεχνητής νοημοσύνης (artificial intelligence - AI), από τους οποίους άλλοι επικεντρώνονται στη διαδικασία σκέψης και συλλογισμού και άλλοι στη συμπεριφορά [1]. Ένας από τους πρώτους ορισμούς που διατυπώθηκαν από τους Barr και Feigenbaum αναφέρει ότι "τεχνητή νοημοσύνη είναι ο τομέας της επιστήμης των υπολογιστών που ασχολείται με τη σχεδίαση ευφυών (νοημόνων) υπολογιστικών συστημάτων, δηλαδή συστημάτων που επιδεικνύουν χαρακτηριστικά που σχετίζονται με τη νοημοσύνη στην ανθρώπινη συμπεριφορά".

Από τον παραπάνω ορισμό προκύπτει ότι για να ορισθεί τι μπορεί να κάνει ένα νοήμον υπολογιστικό σύστημα πρέπει πρώτα να ορισθεί ο όρος "νοημοσύνη". Για τη νοημοσύνη δεν υπάρχει κάποιος αυστηρός και γενικά αποδεκτός ορισμός. Η νοημοσύνη σχετίζεται με την ευφυΐα, τη λογική, τη διανόηση, την ικανότητα επίλυσης προβλημάτων, τη μάθηση από την εμπειρία, την ικανότητα συλλογισμού, την κατανόηση, την ορθολογιστική και αναλυτική σκέψη, την εξαγωγή συμπερασμάτων κτλ.

Έχει αποδειχθεί ότι όσο πιο απλή και αυτονόητη είναι για τους ανθρώπους μια λειτουργία, τόσο πιο δύσκολα μπορεί να μεταφερθεί σε έναν υπολογιστή και να περιγραφεί με ένα πρόγραμμα. Αυτό συμβαίνει γιατί συνήθως οι απλές λειτουργίες εκτελούνται μηχανικά χωρίς ιδιαίτερη σκέψη και συνεπώς είναι πολύ δύσκολο να περιγραφεί πως πραγματοποιήθηκαν. Ορισμένα παραδείγματα τέτοιων λειτουργιών είναι η διαδικασία ανάγνωσης χαρακτήρων, ο σχηματισμός λέξεων και η κατανόηση του νοήματος κειμένων ή ομιλιών, η διαδικασία της μετακίνησης από το σπίτι στο γραφείο περπατώντας και η επιλογή διαδρομής, η αναγνώριση προσώπων. Αντίθετα, αν μια δουλειά είναι δύσκολη και απαιτεί σχεδιασμό των βημάτων που πρέπει να εκτελεστούν για να επιτευχθεί το επιθυμητό αποτέλεσμα, τότε δεν είναι τόσο δύσκολο να περιγραφούν αυτά τα βήματα με κάποιο πρόγραμμα όπως για παράδειγμα στην επίλυση ενός μαθηματικού προβλήματος.

Η κυριότερη προσέγγιση της τεχνητής νοημοσύνης στηρίζεται στην επεξεργασία συμβόλων που έχει μακρύ παρελθόν αλλά και μερικά σοβαρά άλυτα προβλήματα. Ένα σύμβολο από μόνο του δεν υποδηλώνει κάτι αλλά αποκτά κάποιο νόημα μόνο όταν συνδεθεί με άλλα σύμβολα. Επιπλέον, τα ονόματα των συμβόλων επιλέγονται με τέτοιο τρόπο ώστε να έχουν νόημα για τους ανθρώπους. Πρακτικά είναι αδύνατο να γραφεί συμβολικός κώδικας που να αναπαριστά πλήρως και με σαφήνεια μία μοναδική άποψη του κόσμου. Συνήθως τα προγράμματα επεξεργασίας συμβόλων χρησιμοποιούν μία αφηρημένη αναπαράσταση του κόσμου και ένα μηχανισμό για την επεξεργασία τους.

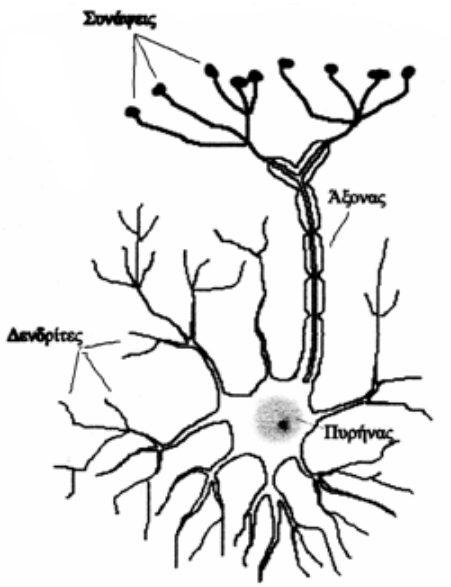
Ωστόσο, οι φυσικές διεργασίες δε λειτουργούν με αυτόν τον τρόπο. Για παράδειγμα, στον ανθρώπινο εγκέφαλο δεν υπάρχει κάποιος γενικός μηχανισμός ελέγχου των λειτουργιών που επιτελούν τα διάφορα τμήματά του, ενώ η γνώση και οι μηχανισμοί επεξεργασίας της δεν εντοπίζονται με ακρίβεια σε συγκεκριμένα σημεία του. Απόδειξη αυτού είναι ότι μικρές τοπικές βλάβες στον εγκέφαλο του ανθρώπου δεν προκαλούν απώλεια συγκεκριμένων πληροφοριών. Με βάση τα παραπάνω μπορούν να διακριθούν δύο προσεγγίσεις για την τεχνητή νοημοσύνη:

- Η κλασική ή συμβολική τεχνητή νοημοσύνη (symbolic AI) που βασίζεται στην κατανόηση των νοητικών διεργασιών και ασχολείται με την προσομοίωση της ανθρώπινης νοημοσύνης προσεγγίζοντας τη με αλγορίθμους και συστήματα που βασίζονται στη γνώση, χρησιμοποιώντας ως δομικές μονάδες τα σύμβολα. Ένα σύμβολο μπορεί να αναπαριστά μία έννοια ή μία σχέση ανάμεσα σε έννοιες. Παραδείγματα αυτής της κατηγορίας είναι οι εφαρμογές της τεχνητής νοημοσύνης που χρησιμοποιούν αναπαράσταση γνώσης όπως λογική, κανόνες, πλαίσια κτλ.
- Η υπολογιστική νοημοσύνη (computational intelligence, soft computing) που βασίζεται στη μίμηση βιολογικών διεργασιών όπως η διαδικασία της εξέλιξης των ειδών ή η λειτουργία του εγκεφάλου. Παραδείγματα τέτοιων τεχνικών αποτελούν τα τεχνητά νευρωνικά δίκτυα και οι γενετικοί αλγόριθμοι.

Τα τεχνητά νευρωνικά δίκτυα (neural networks) μιμούνται την κατανεμημένη λειτουργία του ανθρωπίνου εγκεφάλου και αποτελούνται από πολλά απλά δομικά στοιχεία που ονομάζονται νευρώνες, που ελέγχονται από προσαρμοζόμενες παραμέτρους και είναι ικανά να μαθαίνουν, να γενικεύουν και να αποκρίνονται με εξυπνάδα σε νέα ερεθίσματα. Τα τεχνητά νευρωνικά δίκτυα συγκαταλέγονται ανάμεσα στα πιο ευρύτατα χρησιμοποιούμενα εργαλεία της τεχνητής νοημοσύνης για τη μοντελοποίηση αγνώστων συστημάτων με μη γραμμική συμπεριφορά χωρίς να χρειάζονται κάποιο μαθηματικό μοντέλο αυτών.

2.2 Βιολογικά νευρωνικά δίκτυα

Η ικανότητα του ανθρώπου να σκέφτεται, να θυμάται και να επιλύει προβλήματα εντοπίζεται στον εγκέφαλο. Όπως είναι γνωστό από τη βιολογία, η δομική μονάδα του εγκεφάλου είναι ο νευρώνας (neuron). Η δομή ενός τυπικού βιολογικού νευρώνα φαίνεται στο σχήμα 1.



Σχήμα 1. Αναπαράσταση βιολογικού νευρώνα

Ο βιολογικός νευρώνας αποτελείται από το σώμα που αποτελεί τον πυρήνα του, τους δενδρίτες μέσω των οποίων λαμβάνει σήματα από γειτονικούς νευρώνες (σημεία εισόδου) και τον άξονα που είναι η έξοδος του νευρώνα και το μέσο σύνδεσης του με άλλους νευρώνες. Σε κάθε δενδρίτη υπάρχει ένα απειροελάχιστο κενό που ονομάζεται σύναψη. Οι συνάψεις μέσω χημικών διαδικασιών επιταχύνουν ή επιβραδύνουν τη ροή ηλεκτρικών φορτίων προς το σώμα του νευρώνα. Η ικανότητα μάθησης και μνήμης που παρουσιάζει ο εγκέφαλος οφείλεται στην ικανότητα των συνάψεων να μεταβάλουν την αγωγιμότητα τους. Τα ηλεκτρικά σήματα που εισέρχονται στο σώμα των νευρώνων μέσω των δενδριτών συνδυάζονται και αν το αποτέλεσμα ξεπερνά κάποια τιμή κατωφλίου το σήμα διαδίδεται με τη βοήθεια του άξονα προς άλλους νευρώνες.

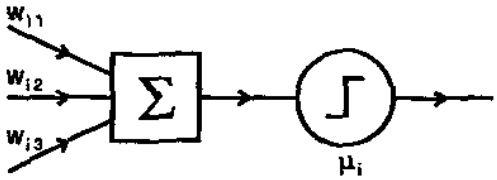
Ο ανθρώπινος εγκέφαλος αποτελείται από περίπου 100 δισεκατομμύρια νευρώνες και σε κάθε νευρώνα αντιστοιχούν κατά μέσο όρο περίπου 1000 συνάψεις οπότε προκύπτει ότι υπάρχουν περίπου 100 τρισεκατομμύρια συνάψεις. Κάθε προσπάθεια να αντιγραφεί η δομή και η λειτουργία του εγκεφάλου σε τέτοια κλίμακα είναι πρακτικά αδύνατη.

Ο χρόνος απόκρισης των βιολογικών νευρώνων είναι της τάξης των χιλιοστών του δευτερολέπτου (msec), πολύ αργότερος δηλαδή σε σύγκριση με τα σημερινά ηλεκτρονικά κυκλώματα. Παρόλα αυτά, ο εγκέφαλος είναι σε θέση να λαμβάνει πολύπλοκες αποφάσεις εξαιρετικά γρήγορα. Αυτό οφείλεται στο ότι η υπολογιστική δυνατότητα του εγκεφάλου και η πληροφορία που περιέχει είναι διαμοιρασμένα σε ολόκληρο τον όγκο του, δηλαδή ό εγκέφαλος αποτελεί ένα παράλληλο και κατανεμημένο υπολογιστικό σύστημα. Τα παραπάνω χαρακτηριστικά είναι ο λόγος για την έρευνα που πραγματοποιείται ώστε να μοντελοποιηθεί ο ανθρώπινος εγκέφαλος με τα τεχνητά νευρωνικά δίκτυα.

2.3 Τεχνητά νευρωνικά δίκτυα

Ο τεχνητός νευρώνας είναι ένα υπολογιστικό μοντέλο που τα μέρη του οποίου μπορούν να αντιστοιχιστούν άμεσα με αυτά του βιολογικού νευρώνα. Όπως φαίνεται στο σχήμα 2 ένας τεχνητός νευρώνας δέχεται κάποια σήματα εισόδου που μεταβάλλονται από μία τιμή βάρους ο ρόλος της οποίας είναι αντίστοιχος του ρόλου της σύναψης στο

βιολογικό νευρώνα. Η τιμή του βάρους μπορεί να είναι θετική ή αρνητική, σε αντιστοιχία με την επιταχυντική ή επιβραδυντική λειτουργία της σύναψης.



Σχήμα 2. Μοντέλο τεχνητού νευρώνα

Το σώμα του τεχνητού νευρώνα αποτελείται από δύο μέρη, από τον αθροιστή ο οποίος προσθέτει τα επηρεασμένα από τα βάρη σήματα εισόδου, και από τη συνάρτηση ενεργοποίησης, ένα είδος φίλτρου το οποίο διαμορφώνει την τελική τιμή του σήματος εξόδου σε συνάρτηση με την έξοδο του αθροιστή και την τιμή κατωφλίου της συνάρτησης ενεργοποίησης. Ένας νευρώνας μπορεί να έχει πολλές εξόδους αλλά όλες θα έχουν την ίδια τιμή.

Τα τεχνητά νευρωνικά δίκτυα (στη συνέχεια θα αναφέρονται απλά ως νευρωνικά δίκτυα) είναι συστήματα επεξεργασίας δεδομένων που αποτελούνται από ένα πλήθος τεχνητών νευρώνων οργανωμένων σε δομές παρόμοιες με αυτές του ανθρωπίνου εγκεφάλου. Συνήθως οι τεχνητοί νευρώνες είναι οργανωμένοι σε μία σειρά από στρώματα ή επίπεδα. Το πρώτο από αυτά ονομάζεται επίπεδο εισόδου και χρησιμοποιείται για την εισαγωγή δεδομένων. Στη συνέχεια, μπορεί να ακολουθούν προαιρετικά ένα ή περισσότερα κρυφά επίπεδα και στο τέλος υπάρχει το επίπεδο εξόδου.

Τα νευρωνικά δίκτυα πραγματοποιούν δύο βασικές λειτουργίες, τη μάθηση και την ανάκληση. Μάθηση είναι η διαδικασία τροποποίησης της τιμής των βαρών του δικτύου ώστε δοθέντος συγκεκριμένου διανύσματος εισόδου να παραχθεί συγκεκριμένο διάνυσμα εξόδου. Η διαδικασία αυτή ονομάζεται και εκπαίδευση. Ανάκληση είναι η διαδικασία υπολογισμού ενός διανύσματος εξόδου για συγκεκριμένο διάνυσμα εισόδου και τιμές βαρών.

Η τροποποίηση των βαρών ενός νευρωνικού δικτύου μπορεί να γίνει με τρία διαφορετικά είδη μάθησης. Στη μάθηση με επίβλεψη (supervised learning) δίνονται στο δίκτυο ζευγάρια διανυσμάτων εισόδου και επιθυμητής εξόδου και αυτό παράγει με την τρέχουσα κατάσταση βαρών μία έξοδο που αρχικά διαφέρει από την επιθυμητή. Αυτή η διαφορά ονομάζεται σφάλμα και με βάση αυτή και ενός αλγορίθμου εκπαίδευσης γίνεται η αναπροσαρμογή των βαρών. Στην ενισχυτική μάθηση η έξοδος χαρακτηρίζεται ως “καλή” ή “κακή” με βάση μια αριθμητική κλίμακα και τα βάρη αναπροσαρμόζονται με βάση αυτό το χαρακτηρισμό.

Στη μάθηση χωρίς επίβλεψη ή ανταγωνιστική μάθηση (unsupervised, competitive learning) η απόκριση του δικτύου βασίζεται στην ικανότητά του να αυτο-οργανώνεται με βάση τα διανύσματα εισόδου καθώς δεν υπάρχουν αντίστοιχα διανύσματα εξόδου. Αυτή η εσωτερική οργάνωση γίνεται έτσι ώστε σε συγκεκριμένο σύνολο εισόδων να αντιδρά ένας συγκεκριμένος νευρώνας. Πρακτικά, τα νευρωνικά δίκτυα αυτής της κατηγορίας καλούνται να μάθουν να κατηγοριοποιούν τα δεδομένα εισόδου.

3

Αυτο-οργανούμενοι χάρτες

Η παρουσίαση των νευρωνικών δικτύων στο προηγούμενο κεφάλαιο βασίζεται αποκλειστικά στην έννοια του συναπτικού βάρους για την αναπαράσταση της πληροφορίας με τη μορφή κάποιας συνάρτησης εισόδου-εξόδου. Η αλλαγή των συναπτικών βαρών μεταξύ των νευρώνων συνεπάγεται και την τροποποίηση της συνάρτησης αυτής και συνεπώς την αλλαγή της πληροφορίας που φυλάσσεται στο δίκτυο. Σε τέτοια νευρωνικά δίκτυα η θέση του κάθε νευρώνα μέσα στην αρχιτεκτονική διάταξη του δικτύου δεν παίζει κάποιο ρόλο. Σε δίκτυα πολλών στρωμάτων είναι σημαντική η πληροφορία ότι ο νευρώνας Α βρίσκεται στο στρώμα Χ, αλλά ακόμη και τότε, η ακριβής θέση του νευρώνα μέσα στο συγκεκριμένο στρώμα δεν έχει ιδιαίτερη σημασία.

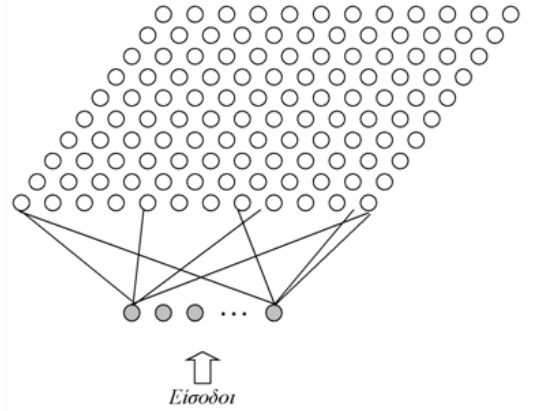
Η τοπολογική πληροφορία, δηλαδή η σχετική διάταξη των νευρώνων στο δίκτυο, εμφανίζεται να παίζει σημαντικό ρόλο σε διάφορα τμήματα του εγκεφάλου που εκτελούν συγκεκριμένες λειτουργίες, όπως την αντίληψη του ήχου, της αφής, της εικόνας, κλπ. Τα τμήματα αυτά διαθέτουν αυστηρή τοπολογική οργάνωση έτσι ώστε οι νευρώνες που διεγείρονται από συναφή ή γειτονικά εξωτερικά ερεθίσματα να βρίσκονται κοντά ο ένας με τον άλλο. Τέτοιες δομές αποκαλούνται χάρτες.

Ένα μοντέλο νευρωνικών δικτύων που να παρουσιάζει τοπολογική οργάνωση των νευρώνων προτάθηκε από τον T. Kohonen και ονομάζεται Αυτο-οργανούμενος Χάρτης (Self Organizing feature Map – SOM) [2]. Το δίκτυο SOM είναι μία από τις πιο χαρακτηριστικές περιπτώσεις δικτύων που χρησιμοποιεί ανταγωνιστική μάθηση.

3.1 Ο αλγόριθμος SOM

Ένας αυτο-οργανούμενος χάρτης αποτελείται από ένα πλήθος νευρώνων οι οποίοι είναι τοποθετημένοι σε κάποια γεωμετρική τοπολογία όπως ευθεία, επίπεδο, σφαίρα με πιο συνηθισμένη εφαρμογή τους χάρτες δύο διαστάσεων δηλαδή επίπεδα. Στο σχήμα 3

φαίνεται πως δεδομένα εισόδου n διαστάσεων απεικονίζονται σε ένα δυσδιάστατο πλέγμα νευρώνων SOM. Το πλέγμα μπορεί να είναι τετραγωνικό ή εξαγωνικό που είναι καλύτερο για την οπτική απεικόνιση του πλέγματος. Για κάθε νευρώνα υπάρχει ένα διαφορετικό διάνυσμα βαρών $m_i = [\mu_{i1}, \mu_{i2}, ..., \mu_{in}]^T \in \mathbb{R}^n$ μέσω του οποίου είναι συνδεδεμένος με τα δεδομένα εισόδου.



Σχήμα 3. Αυτο-οργανούμενος χάρτης δύο διαστάσεων

Κάθε διάνυσμα εισόδου $x = [\xi_1, \xi_2, ..., \xi_n]^T \in \mathbb{R}^n$ συγκρίνεται με όλα τα βάρη m_i και στη θέση που υπάρχει καλύτερο ταίριασμα σύμφωνα με κάποιο μέτρο απόστασης αναδεικνύεται ο νικητής νευρώνας. Η ακριβής απόσταση του προτύπου από τον νευρώνα δεν έχει σημασία, αρκεί να βρεθεί η θέση του νευρώνα στο πλέγμα. Ως μέτρο απόστασης μπορεί να χρησιμοποιηθεί η ευκλείδεια απόσταση μεταξύ των δύο διανυσμάτων

$$\|x - m_i\| = \sqrt{\sum_{k=1}^{n} (x_k - m_{ik})^2}$$

και νικητής νευρώνας ορίζεται ο νευρώνας c για τον οποίο ισχύει:

$$\|x - m_c\| = \min_i \{\|x - m_i\|\}$$

Στη συνέχεια ανανεώνονται τα βάρη των νευρώνων του χάρτη σύμφωνα με τη σχέση

$$m_i(t+1) = m_i(t) + a(t)h_{ci}(t)[x(t) - m_i(t)]$$

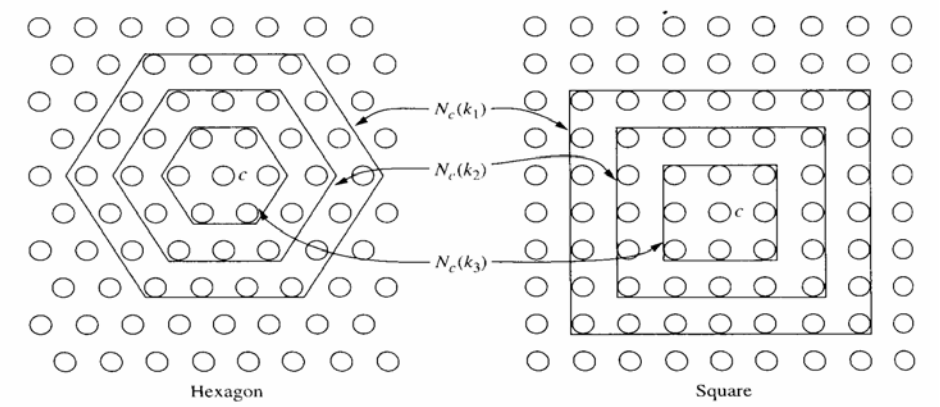
όπου η μεταβλητή t δηλώνει διακριτές χρονικές στιγμές, α(t) είναι ο ρυθμός μάθησης ($0 < \alpha(t) < 1$) και $h_{ci}(t)$ είναι η συνάρτηση γειτονιάς γύρω από το νευρώνα νικητή c.

Η δημιουργία και εκπαίδευση του SOM ξεκινάει από μια αρχική τυχαία κατάσταση, όπου όλα τα βάρη είναι τυχαίοι αριθμοί και καταλήγει σε τοπολογική τακτοποίηση των νευρώνων περνώντας μέσα από δύο φάσεις. Στη πρώτη φάση εκπαίδευσης γίνεται η δημιουργία των γειτονιών στο πλέγμα και η τακτοποίηση των νευρώνων σε αυτές. Στο δεύτερο στάδιο γίνεται ρύθμιση των τιμών των βαρών κάθε νευρώνα ώστε αυτές να πάρουν τις τελικές τους τιμές.

Ο ρυθμός μάθησης μειώνεται με το χρόνο κατά τη διάρκεια της εκπαίδευσης ώστε οι τιμές των βαρών να μην ταλαντεύονται και αλλάζουν συνέχεια αλλά να συγκλίνουν. Στην αρχή της φάσης εκπαίδευσης πρέπει ο ρυθμός μάθησης να έχει τιμή κοντά στη μονάδα και μετά από κάποιο αριθμό εποχών εκπαίδευσης να μειώνεται μονοτονικά. Η μείωση μπορεί να είναι γραμμική, εκθετική ή αντιστρόφως ανάλογη του χρόνου. Στο στάδιο της ρύθμισης η τιμή του ρυθμού μάθησης μένει σταθερή σε μικρά επίπεδα και συνήθως επιλέγεται η τιμή $\alpha = 0.02$.

Η συνάρτηση γειτονιάς επιλέγεται συνήθως έτσι ώστε να είναι $h_{ci}(t) = h(\|r_c - r_i\|, t)$ όπου $r_c, r_i \in \mathbb{R}^2$ είναι οι θέσεις των νευρώνων στο πλέγμα και για τη συνάρτηση h_{ci} ισχύει

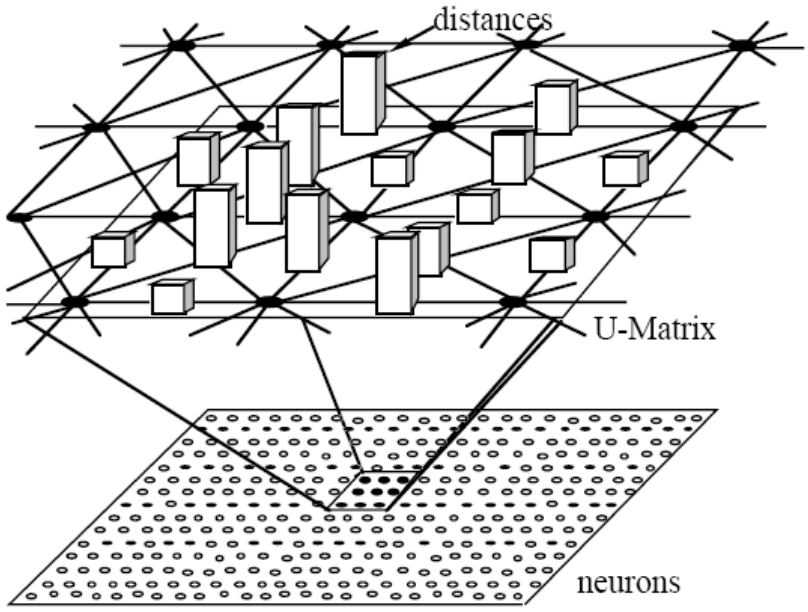
ότι $h_{ci} \rightarrow 0$ καθώς η απόσταση από το νευρώνα νικητή $||r_c - r_i||$ μεγαλώνει. Δύο απλές επιλογές για την συνάρτηση γειτονιάς όπου η ακτίνα της γειτονιάς μειώνεται με το χρόνο φαίνονται στο σχήμα 4. Για τη συνάρτηση γειτονιάς μπορεί επίσης να χρησιμοποιηθεί η γκαουσιανή συνάρτηση με πλάτος που να μειώνεται εκθετικά.



Σχήμα 4. Παραδείγματα συνάρτησης γειτονιάς ($k_1 < k_2 < k_3$)

3.2 U-matrix

Σε ένα SOM δεν είναι δυνατό να φανούν τα όρια μεταξύ των ομάδων των νευρώνων που δημιουργούνται σε αυτό. Για να υπάρξει μια γραφική απεικόνιση των ομάδων μπορεί να χρησιμοποιηθεί η μέθοδος του U-matrix (Unified distance matrix) [3]. Ο πίνακας U-matrix σχηματίζεται όταν για κάθε νευρώνα υπολογίζεται η μέση τιμή των αποστάσεών του από τους γειτονικούς του νευρώνες (σχήμα 5).



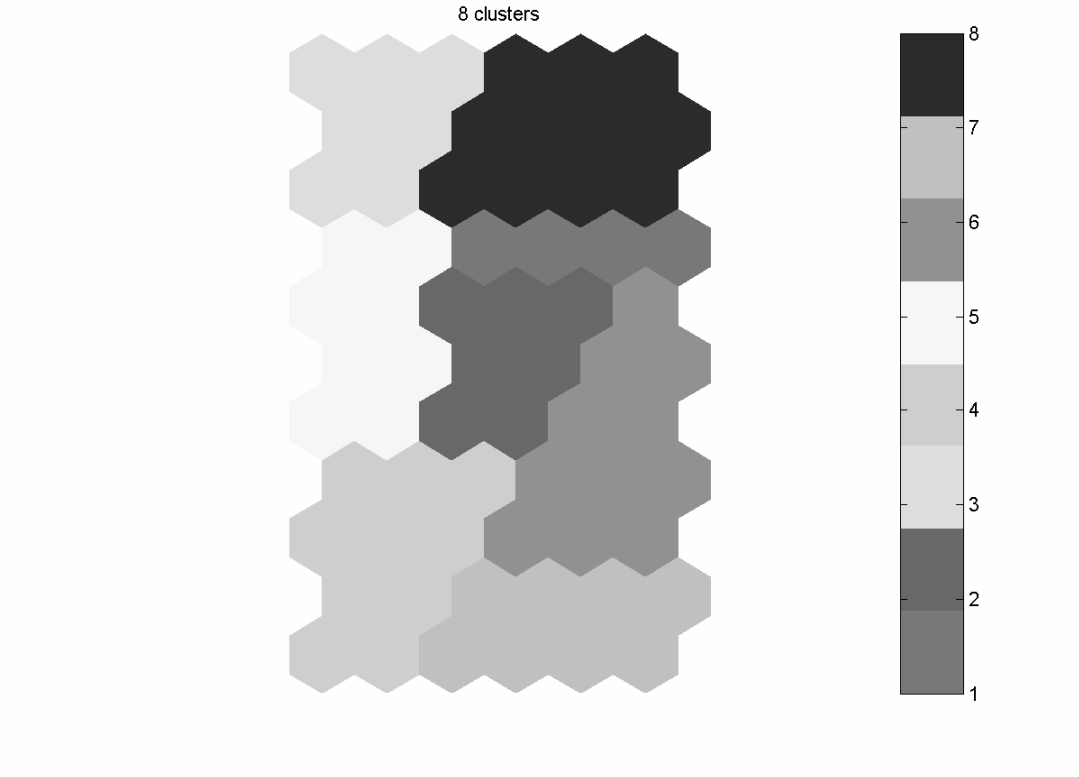
Σχήμα 5. Σχηματισμός U-matrix

Οι τιμές αυτές μπορούν να θεωρηθούν ως τρίτη διάσταση του πλέγματος του χάρτη και να δημιουργηθεί ένα σχήμα 3 διαστάσεων με λόφους και κοιλάδες. Οι νευρώνες που βρίσκονται στην ίδια κοιλάδα είναι αρκετά όμοιοι μεταξύ τους και αναμένεται να ανήκουν

στην ίδια ομάδα. Η απεικόνιση μπορεί επίσης να γίνει χρησιμοποιώντας διαφορετικό χρωματισμό για τους νευρώνες του χάρτη ανάλογα με την τιμή που έχει ο U-matrix.

3.3 Τεχνικές ομαδοποίησης

Ομαδοποίηση (clustering) είναι μια υποδιαίρεση των δεδομένων σε ομάδες με παρόμοια χαρακτηριστικά [4]. Τυπικά μια ομαδοποίηση Q ορίζεται ως ένα σύνολο ομάδων Q_i, i = 1,…,C όπου C το πλήθος των ομάδων και κάθε πρότυπο του συνόλου δεδομένων ανήκει σε μια ομάδα Q_i. Η υποδιαίρεση γίνεται με τέτοιο τρόπο ώστε να ελαχιστοποιούνται οι αποστάσεις των δεδομένων μέσα στην ίδια ομάδα και να μεγιστοποιούνται οι αποστάσεις μεταξύ δεδομένων από διαφορετικές ομάδες. Μέσω των ομάδων που δημιουργούνται χάνονται κάποιες λεπτομέρειες του συνόλου δεδομένων αλλά είναι ευκολότερη η ποσοτική περιγραφή των δεδομένων ιδιαίτερα αν αυτά έχουν μεγάλο όγκο. Οι διάφορες τεχνικές ομαδοποίησης μπορούν να εφαρμοστούν είτε κατ' ευθείαν στα αρχικά δεδομένα του προβλήματος είτε στο SOM που προκύπτει από αυτά ομαδοποιώντας τους νευρώνες του πλέγματος. Έτσι επιτυγχάνεται ομαδοποίηση δυο επιπέδων καθώς το SOM πραγματοποιεί μια αρχική ομαδοποίηση των δεδομένων μειώνοντας σημαντικά το συνολικό υπολογιστικό φορτίο της ομαδοποίησης. Η διαδικασία αυτή πρέπει να θεωρείται έγκυρη όταν οι ομάδες που δημιουργούνται χρησιμοποιώντας το SOM είναι όμοιες με αυτές που δημιουργούνται στα αρχικά δεδομένα.



Σχήμα 6. Παράδειγμα ομαδοποίησης σε SOM

3.3.1 Ορισμός αποστάσεων ομάδων

Οι αποστάσεις των προτύπων που βρίσκονται μέσα στην ίδια ομάδα όπως και οι αποστάσεις μεταξύ προτύπων διαφορετικών ομάδων μπορούν να οριστούν με διάφορους τρόπους που οδηγούν σε διαφορετικά αποτελέσματα ομαδοποίησης. Για την απόσταση μεταξύ των προτύπων $\|\cdot\|$ χρησιμοποιείται η ευκλείδεια νόρμα. Θέτοντας $x_i, x_{i'} \in Q_k$, $i \neq i'$, $x_j \in Q_l$, $k \neq l$, N_k = πλήθος των προτύπων στην ομάδα Q_k και $c_k = \frac{1}{N_k}\Sigma_{x_i \in Q_k} x_i$.

- Αποστάσεις μέσα στην ίδια ομάδα

α) Μέση απόσταση (μεταξύ όλων των προτύπων της ομάδας)

$$S_a = \frac{\Sigma_{i,i'} \|x_i - x_{i'}\|}{N_k (N_k - 1)}$$

β) Απόσταση πλησιέστερου γείτονα, ελάχιστη απόσταση μεταξύ προτύπων.

$$S_{nn} = \frac{\Sigma_i \min_{i'} \{\|x_i - x_{i'}\|\}}{N_k}$$

γ) Κεντροειδής απόσταση, απόσταση προτύπων με το κέντρο της ομάδας

$$S_C = \frac{\Sigma_i \|x_i - c_k\|}{N_k}$$

- Αποστάσεις μεταξύ ομάδων

α) Απλή σύνδεση (single), ελάχιστη απόσταση μεταξύ προτύπων των δύο ομάδων (Σχήμα 7).

$$d_s = \min_{i,j} \{\|x_i - x_j\|\}$$

β) Πλήρης σύνδεση (complete), μέγιστη απόσταση μεταξύ προτύπων των δύο ομάδων (Σχήμα 7).

$$d_{co} = \max_{i,j} \{\|x_i - x_j\|\}$$

γ) Μέση σύνδεση (average), μέση απόσταση μεταξύ όλων των ζευγών προτύπων των δυο ομάδων.

$$d_a = \frac{\Sigma_{i,j} \|x_i - x_j\|}{N_k N_l}$$

δ) Κεντροειδής σύνδεση (centroid), απόσταση μεταξύ των κέντρων των δυο ομάδων

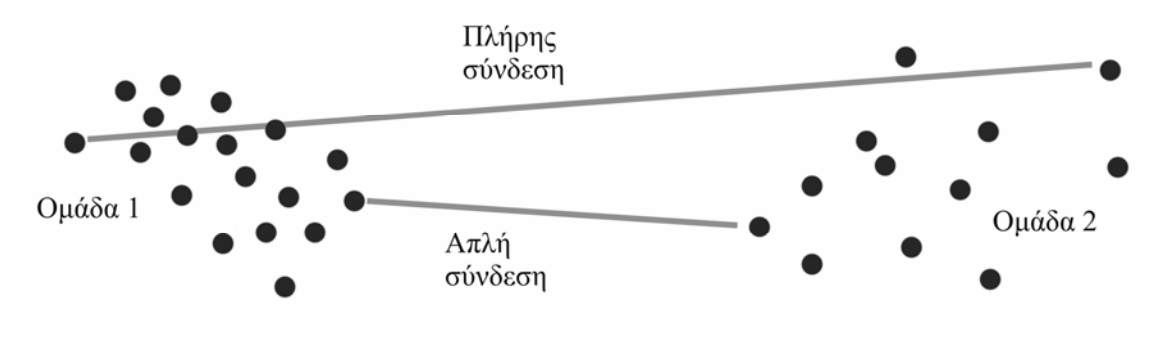
$$d_{ce} = \|c_k - c_l\|$$

ε) Σύνδεση Ward, όπου ελαχιστοποιείται το άθροισμα των τετραγώνων των προτύπων μέσα στην ομάδα που προκύπτει από τον συνδυασμό δύο άλλων ομάδων r και s.

$$d^2_{ward} = N_r N_s \frac{\|c_r - c_s\|^2}{(N_r + N_s)}$$

Από τις αποστάσεις που αναφέρθηκαν παραπάνω οι αποστάσεις πλησιέστερου γείτονα S_{nn} και d_s βασίζονται σε τοπικά κριτήρια και είναι αρκετά ευαίσθητες σε θόρυβο. Αντίθετα, οι αποστάσεις που βασίζονται στα συγκεντρωτικά στοιχεία των ομάδων όπως η

μέση τιμή των αποστάσεων όλων των προτύπων της ομάδας δεν αντιμετωπίζουν τέτοια προβλήματα.



Σχήμα 7. Απλή και πλήρης σύνδεση μεταξύ δυο ομάδων

3.3.2 Δείκτες αξιοπιστίας ομαδοποίησης

Από τους διαφορετικούς τρόπους που μπορούν να οριστούν οι αποστάσεις μεταξύ προτύπων και μεταξύ ομάδων γίνεται φανερό ότι για ένα σύνολο δεδομένων μπορούν να υπάρχουν διαφορετικές ομαδοποιήσεις. Το βέλτιστο πλήθος των ομάδων δεν είναι γνωστό εκ των προτέρων και οι αλγόριθμοι ομαδοποίησης δεν καταλήγουν σε μια βέλτιστη ομαδοποίηση αλλά δίνουν αποτελέσματα για διάφορες τιμές του πλήθους των ομάδων. Επιπλέον, ακόμα και αν το πλήθος των ομάδων είναι προκαθορισμένο μπορεί ο ίδιος αλγόριθμος να δίνει διαφορετικά αποτελέσματα αν κάποιο βήμα του περιέχει τυχαίες διαδικασίες όπως για παράδειγμα τυχαία αρχικοποίηση.

Δημιουργείται έτσι η ανάγκη επιλογής της καταλληλότερης ομαδοποίησης για ένα σύνολο δεδομένων. Αν η ομαδοποίηση μπορεί να οπτικοποιηθεί σε δυο διαστάσεις η επιλογή μπορεί να γίνει από κάποιον ειδικό του τομέα από όπου προέρχονται τα δεδομένα. Αυτό όμως δεν προσφέρει κάποιον τυπικό τρόπο επιλογής ομαδοποίησης και για αυτό έχουν προταθεί διάφοροι αντικειμενικοί δείκτες αξιοπιστίας που χρησιμοποιώντας ποσοτικά χαρακτηριστικά των ομάδων ξεχωρίζουν τις πιο ενδιαφέρουσες και σημαντικές ομαδοποιήσεις. Ένας από τους πιο γνωστούς δείκτες, ο οποίος θα χρησιμοποιηθεί και στην παρούσα εργασία, είναι ο δείκτης Davies-Bouldin [5].

Ο δείκτης Davies-Bouldin χρησιμοποιεί την κεντροειδή απόσταση S_c για τις αποστάσεις μέσα στην ίδια ομάδα και την d_{ce} για τις αποστάσεις μεταξύ ομάδων. Το μέτρο ομοιότητας R_{ij} ορίζεται έτσι ώστε να ικανοποιεί τις παρακάτω συνθήκες:

1. $R_{ij} \geq 0$
2. $R_{ij} = R_{ji}$
3. Αν $S_{ci} = 0$ και $S_{cj} = 0$ τότε $R_{ij} = 0$
4. Αν $S_{cj} > S_{ck}$ και $d_{ij} = d_{ik}$ τότε $R_{ij} > R_{ik}$
5. Αν $S_{cj} = S_{ck}$ και $d_{ij} < d_{ik}$ τότε $R_{ij} > R_{ik}$

Ένα απλό μέτρο που ικανοποιεί τις συνθήκες είναι $R_{ij} = (S_{ci} + S_{cj}) / d_{ij}$. Ο δείκτης Davies-Bouldin ορίζεται ως

$$DB_C = \frac{1}{C}\sum_{i=1}^{C} R_i$$

όπου $R_i = \max R_{ij}$, $i = 1,\ldots,C$, $i \neq j$ και C είναι το πλήθος των ομάδων. Ο δείκτης DB είναι η μέση ομοιότητα μεταξύ κάθε ομάδας και της αμέσως πιο κοντινής της άρα η καλύτερη ομαδοποίηση των δεδομένων θα ελαχιστοποιεί το δείκτη. Σχεδιάζοντας τον δείκτη DB ως συνάρτηση του πλήθους των ομάδων C μπορεί να βρεθεί το βέλτιστο C εκεί που ο δείκτης DB θα έχει ελάχιστο.

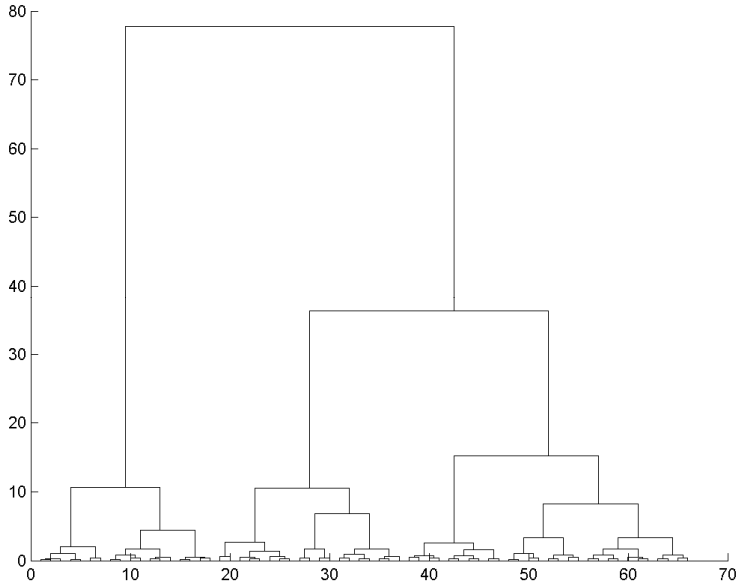
3.3.3 Ιεραρχικοί αλγόριθμοι ομαδοποίησης

Οι τεχνικές ομαδοποίησης χωρίζονται σε δυο κύριες κατηγορίες, τους ιεραρχικούς και τους κατατμητικούς αλγόριθμους. Οι ιεραρχικοί αλγόριθμοι μπορούν να χωριστούν σε συσσωρευτικούς (agglomerative) και διαιρετικούς (divisive). Οι συσσωρευτικοί αλγόριθμοι ακολουθούν την παρακάτω διαδικασία:

1. Αρχικοποίηση όπου κάθε πρότυπο αποτελεί από μόνο του μια ομάδα
2. Υπολογισμός αποστάσεων μεταξύ όλων των ομάδων
3. Συγχώνευση των δυο ομάδων με την μικρότερη απόσταση
4. Επιστροφή στο βήμα 2 μέχρι να σχηματιστεί μια μόνο ομάδα

Οι διαιρετικοί αλγόριθμοι ακολουθούν την αντίστροφη πορεία θεωρώντας αρχικά ότι όλα τα πρότυπα ανήκουν στην ίδια ομάδα και στη συνέχεια διαχωρίζουν διαδοχικά την ομάδα σε δυο μικρότερες μέχρι να κάθε πρότυπο να αποτελεί μια ομάδα ή να ικανοποιηθεί κάποιο κριτήριο όπως ένας προκαθορισμένος μέγιστος αριθμός ομάδων. Οι ιεραρχικοί αλγόριθμοι είναι υπολογιστικά πιο αποδοτικοί από τους διαιρετικούς και γι' αυτό χρησιμοποιούνται πιο συχνά στην πράξη.

Το αποτέλεσμα των ιεραρχικών αλγορίθμων είναι το δέντρο ομάδων που ονομάζεται δενδρόγραμμα (σχήμα 8). Στο δενδρόγραμμα κάθε κόμβος του δέντρου είναι μία ομάδα και οι κόμβοι που έχουν τον ίδιο γονέα είναι οι ομάδες στις οποίες διαχωρίζεται ο κοινός κόμβος γονέας. Από το δενδρόγραμμα μπορεί να μελετηθεί η δομή των δεδομένων και να βρεθεί ο αριθμός των ομάδων.



Σχήμα 8. Παράδειγμα δενδρογράμματος

Ένα δενδρόγραμμα που ομαδοποιεί Ν πρότυπα θα έχει Ν επίπεδα. Από αυτά τα περισσότερα δεν προσφέρουν κάποια σημαντική πληροφορία αφού θα διαφέρουν κατά ένα μικρό αριθμό προτύπων. Άρα το δενδρόγραμμα δεν προσφέρει μια μοναδική ομαδοποίηση αλλά πρέπει να κοπεί σε ορισμένο ύψος ανάλογα με το πλήθος των ομάδων που επιθυμεί ο χρήστης. Εναλλακτικά, μπορεί να κοπεί σε διάφορα σημεία και να αξιολογηθούν οι ομαδοποιήσεις που δημιουργούνται και τελικά να επιλεγεί αυτή που θεωρείται καλύτερη αναφορικά με το κριτήριο της αξιολόγησης. Σαν κριτήριο μπορεί να χρησιμοποιηθεί ο δείκτης Davies-Bouldin που αναφέρθηκε παραπάνω.

3.3.4 Κατατμητικοί αλγόριθμοι ομαδοποίησης

Οι κατατμητικοί (partitive) αλγόριθμοι ομαδοποίησης χωρίζουν το σύνολο δεδομένων σε ομάδες προσπαθώντας να ελαχιστοποιήσουν κάποιο κριτήριο ή κάποια συνάρτηση σφάλματος. Σε αντίθεση με τους ιεραρχικούς αλγόριθμους, οι μεριστικοί δεν βασίζονται στην ομαδοποίηση που είχε βρεθεί στο προηγούμενο βήμα της εκτέλεσης τους αλλά υπολογίζουν κάθε φορά μια νέα ομαδοποίηση. Οι κατατμητικοί αλγόριθμοι ακολουθούν την παρακάτω γενική διαδικασία:

1. Καθορισμός πλήθους ομάδων
2. Αρχικοποίηση κέντρων ομάδων
3. Υπολογισμός διαχωρισμού των δεδομένων
4. Ανανέωση κέντρων ομάδων
5. Επιστροφή στο βήμα 3 εκτός αν ο διαχωρισμός δεν αλλάζει ή ο αλγόριθμος συγκλίνει

Ένας κατατμητικός αλγόριθμος που χρησιμοποιείται συχνά είναι ο αλγόριθμος ταξινόμησης κ-μέσων. Αρχικά επιλέγονται τυχαία Κ πρότυπα που αποτελούν τα κέντρα των Κ ομάδων. Στη συνέχεια κάθε πρότυπο κατηγοριοποιείται στην ομάδα της οποίας το κέντρο βρίσκεται πιο κοντά με την έννοια της ευκλείδειας απόστασης. Στη συνέχεια υπολογίζονται τα νέα κέντρα των ομάδων και ο αλγόριθμος επαναλαμβάνεται μέχρι να μην αλλάζει καμία ομάδα. Σκοπός του αλγορίθμου είναι η ελαχιστοποίηση της συνάρτησης $E = \sum_{i=1}^{K} \sum_{x \in Q_i} \|x - c_i\|$ δηλαδή η ελαχιστοποίηση της απόστασης κάθε προτύπου από το κέντρο της ομάδας στην οποία ανήκει.

Αν το βέλτιστο πλήθος των ομάδων δεν είναι γνωστό εκ των προτέρων ο αλγόριθμος των κ-μέσων επαναλαμβάνεται για τιμές του κ από 2 μέχρι $\sqrt{N}$ όπου Ν ο αριθμός των προτύπων. Στη συνέχεια επιλέγεται το βέλτιστο κ με χρήση κάποιου κριτηρίου όπως ο δείκτης Davies-Bouldin. Για ασφαλέστερα αποτελέσματα, ο αλγόριθμος των κ-μέσων πρέπει να εκτελείται πολλές φορές με τυχαία αρχικοποίηση κάθε φορά.

4

Συμβολική αναπαράσταση γνώσης και νευρωνικά δίκτυα

Τα νευρωνικά δίκτυα χρησιμοποιούνται για κατηγοριοποίηση δεδομένων και προβλέψεις χωρίς να είναι γνωστό πως απέκτησαν και χρησιμοποιούν αυτή τη γνώση. Αντίθετα, η γνώση που εκφράζεται σε μορφή συμβολικών κανόνων είναι πολύ πιο κατανοητή και προσιτή σε ανθρώπους. Η εξαγωγή κανόνων από νευρωνικά δίκτυα είναι ένα σημαντικό πρόβλημα στο πεδίο της μηχανικής μάθησης το οποίο δεν έχει μέχρι τώρα λυθεί με ικανοποιητικό τρόπο.

4.1 Προτασιακή λογική

Ο απλούστερος και πιο κατανοητός τρόπος για τη συμβολική παράσταση γνώσης είναι η προτασιακή λογική (propositional logic). Οι δομή των κανόνων της προτασιακής λογικής είναι τέτοια που πάντα δημιουργούνται σύνορα αποφάσεων στον πολυδιάστατο χώρο του συνόλου δεδομένων. Η γενική μορφή των κανόνων είναι

$$\text{ΑΝ } \mathbf{X} \in \mathrm{X}^{(i)} \text{ ΤΟΤΕ ΚΑΤΗΓΟΡΙΑ}(\mathbf{X}) = C_k$$

δηλαδή αν το πρότυπο **X** ανήκει στο υποσύνολο $\mathrm{X}^{(\mathrm{i})}$ του χώρου του συνόλου δεδομένων τότε πρέπει να κατηγοριοποιηθεί στην κατηγορία $\mathrm{C_k}$. Η συνθήκη του κανόνα μπορεί να είναι μια λογική πρόταση του τύπου $\mathrm{L(X_i)}$, όπου $\mathrm{X_i}$ είναι ένα διάστημα τιμών της μεταβλητής, ή ένας συνδυασμός λογικών προτάσεων. Ο συνδυασμός μπορεί να γίνει είτε με σύζευξη (λογικό "ΚΑΙ") οπότε ο κανόνας παίρνει τη μορφή

$$\text{ΑΝ } L(X_1) \wedge L(X_2) \wedge \ldots \wedge L(X_n) \text{ΤΟΤΕ ΚΑΤΗΓΟΡΙΑ}(\mathbf{X}) = C_k$$

είτε με διάζευξη (λογικό "Η") όπου ο κανόνας παίρνει τη μορφή

$$\text{ΑΝ } L_1(X_1) \vee L_2(X_2) \vee \ldots \vee L_n(X_n) \text{ΤΟΤΕ ΚΑΤΗΓΟΡΙΑ}(\mathbf{X}) = C_k$$

είτε με συνδυασμό συζεύξεων και διαζεύξεων οπότε ο κανόνας γίνεται

$$\text{ΑΝ}\left(L_1(X_1) \wedge L_2(X_2) \wedge \ldots \wedge L_m(X_m)\right) \vee L_{m+1}(X_{m+1}) \vee L_{m+2}(X_{m+2}) \vee \ldots \vee L_n(X_n)$$
$$\text{ΤΟΤΕ ΚΑΤΗΓΟΡΙΑ}(\mathbf{X}) = C_k$$

Τα σύνορα αποφάσεων που δημιουργεί ένα σύνολο κανόνων δεν μπορούν να προσεγγίσουν καλά τα αποτελέσματα των νευρωνικών δικτύων αν υπάρχουν μόνο λίγοι κανόνες. Η ακρίβεια του συνόλου κανόνων βελτιώνεται με την εισαγωγή περισσότερων κανόνων, κάτι που όμως οδηγεί σε λιγότερο κατανοητή περιγραφή. Αναγκαστικά σε ένα σύστημα κανόνων θα πρέπει να υπάρχει ένας συμβιβασμός μεταξύ απλότητας και ακρίβειας.

4.2 Αξιολόγηση συστημάτων εξαγωγής γνώσης

Τα συστήματα που έχουν σχεδιαστεί για να εξάγουν γνώση σε μορφή κανόνων από νευρωνικά δίκτυα βασίζονται κυρίως σε δίκτυα πρόσθιας τροφοδότησης (feedforward neural networks) παρόλο που η χρήση αυτο-οργανούμενων χαρτών είναι μια επίσης δημοφιλής τεχνική σε εφαρμογές εξόρυξης δεδομένων. Στα πλαίσια αυτών των συστημάτων έχουν προταθεί κάποια κριτήρια για την ταξινόμηση των διάφορων αλγορίθμων [6]. Τα κριτήρια αυτά είναι ουσιαστικά ανεξάρτητα από τον τύπο του νευρωνικού δικτύου του συστήματος άρα έχουν νόημα και για αλγορίθμους που βασίζονται σε αυτο-οργανούμενους χάρτες. Υπάρχουν 5 κύρια κριτήρια για την ταξινόμηση:

1. Η εκφραστική δύναμη των κανόνων
2. Η ποιότητα των κανόνων
3. Το επίπεδο στο οποίο ο αλγόριθμος θεωρεί το νευρωνικό δίκτυο
4. Η αλγοριθμική πολυπλοκότητα
5. Η φορητότητα του αλγορίθμου σε διαφορετικές κλάσεις νευρωνικών δικτύων

Η εκφραστική δύναμη των κανόνων μπορεί εναλλακτικά να εκφραστεί ως η μορφή των κανόνων. Υπάρχουν τρεις περιπτώσεις κανόνων, οι συμβολικοί κανόνες προτασιακής λογικής, οι κανόνες ασαφούς λογικής και οι κανόνες σε λογική πρώτης τάξης (first-order-logic form). Το δεύτερο κριτήριο αποτελεί ουσιαστικά το κριτήριο επιλογής των κανόνων που θα συμπεριληφθούν σε ένα σύστημα κανόνων οπότε θα εξηγηθεί αναλυτικά στην επόμενη παράγραφο.

Για το τρίτο κριτήριο υπάρχουν δυο κύριες προσεγγίσεις, η παιδαγωγική (pedagogical) και η αναλυτική (decompositional). Στην παιδαγωγική προσέγγιση το νευρωνικό δίκτυο θεωρείται ως "μαύρο κουτί" και εξάγονται σχέσεις μεταξύ των εισόδων και των εξόδων χωρίς καμία ανάλυση των ενδιάμεσων στοιχείων του δικτύου. Η αναλυτική προσέγγιση είναι το ακριβώς αντίθετο αφού εξάγονται κανόνες από κάθε νευρώνα του δικτύου και στη συνέχεια συνενώνονται για να σχηματίσουν πιο γενικούς κανόνες για τα δεδομένα. Τέλος, υπάρχουν προσεγγίσεις ανάμεσα στα δύο παραπάνω άκρα που θεωρούν το νευρωνικό δίκτυο σε κάποιο ενδιάμεσο επίπεδο αφαίρεσης για να δημιουργήσουν κανόνες.

Η αλγοριθμική πολυπλοκότητα αναφέρεται στην πολυπλοκότητα του αλγορίθμου που ή των αλγορίθμων που δημιουργούν τους κανόνες και βρίσκονται στο κέντρο των συστημάτων εξαγωγής γνώσης. Αν και η πολυπλοκότητα των αλγορίθμων μπορεί να είναι ένα κρίσιμο ζήτημα σε ορισμένες εφαρμογές, σε πολλούς αλγορίθμους που υπάρχουν στη βιβλιογραφία για νευρωνικά δίκτυα πρόσθιας τροφοδότησης δε γίνεται κανένας

σχολιασμός αυτού του θέματος. Το τελευταίο κριτήριο αναφέρεται στην δυνατότητα των αλγορίθμων να εφαρμοστούν με επιτυχία σε διαφορετικές κλάσεις νευρωνικών δικτύων και διαφορετικές μεθόδους εκπαίδευσης ή αν είναι ειδικά σχεδιασμένοι για μια συγκεκριμένη αρχιτεκτονική νευρωνικών δικτύων.

4.3 Μέτρα σημαντικότητας κανόνων

Σκοπός των μέτρων σημαντικότητας κανόνων είναι να προσφέρουν μια αξιολόγηση των κανόνων και να βάζουν ένα όριο στο μεγάλο αριθμό κανόνων που δημιουργούνται από τους αλγορίθμους. Υπάρχουν δύο βασικοί τρόποι για την αξιολόγηση των κανόνων, ο αντικειμενικός και ο υποκειμενικός [7].

Η αντικειμενική προσέγγιση βασίζεται στη δομή και την ακρίβεια των κανόνων αλλά μερικές φορές δεν μπορεί να περιγράψει πλήρως την πολυπλοκότητα των δεδομένων. Η υποκειμενική προσέγγιση βασίζεται και στον χρήστη ή τον ειδικό που εξετάζει τα αποτελέσματα. Η ανάγκη για την ύπαρξη υποκειμενικής προσέγγισης οφείλεται στο γεγονός ότι μόνο έτσι μπορεί να βρεθεί απροσδόκητη πληροφορία σχετικά με τα δεδομένα που δεν ήταν γνωστή στους ειδικούς. Επίσης, έτσι αξιολογείται και η δυνατότητα που μπορεί να δίνει κάποια νέα πληροφορία ώστε οι χρήστες να μπορούν να δράσουν προς όφελος τους βασισμένοι σε αυτή. Οι δυο αυτοί λόγοι είναι σημαντικοί από μόνοι τους αλλά μπορούν και να συνδυαστούν αφού νέα και απροσδόκητη γνώση μπορεί να οδηγήσει σε διαφορετικές ενέργειες από τον χρήστη.

Η αντικειμενική προσέγγιση προσφέρει ποσοτικές πληροφορίες για την εγκυρότητα και την σημασία των κανόνων. Ένας κανόνα R_i μπορεί να θεωρηθεί με δυο διαφορετικούς τρόπους, ως χαρακτηριστικός (characterizing) και ως διαχωριστικός (differentiating) κανόνας [8]. Ο χαρακτηριστικός κανόνας δηλώνει ότι αν ένα πρότυπο ανήκει στην κατηγορία C_i τότε θα ικανοποιεί και τον αντίστοιχο κανόνα δηλαδή:

$$R_i^C : \quad \mathbf{x} \in C_i \Rightarrow x_k \in [\alpha_k, \beta_k]$$

Ο διαχωριστικός κανόνας δηλώνει ότι αν ένα πρότυπο ικανοποιεί τη συνθήκη του κανόνα που έχει κατασκευαστεί για μια κατηγορία τότε θα ανήκει στην κατηγορία αυτή δηλαδή:

$$R_i^d : \quad x_k \in [\alpha_k, \beta_k] \Rightarrow \mathbf{x} \in C_i$$

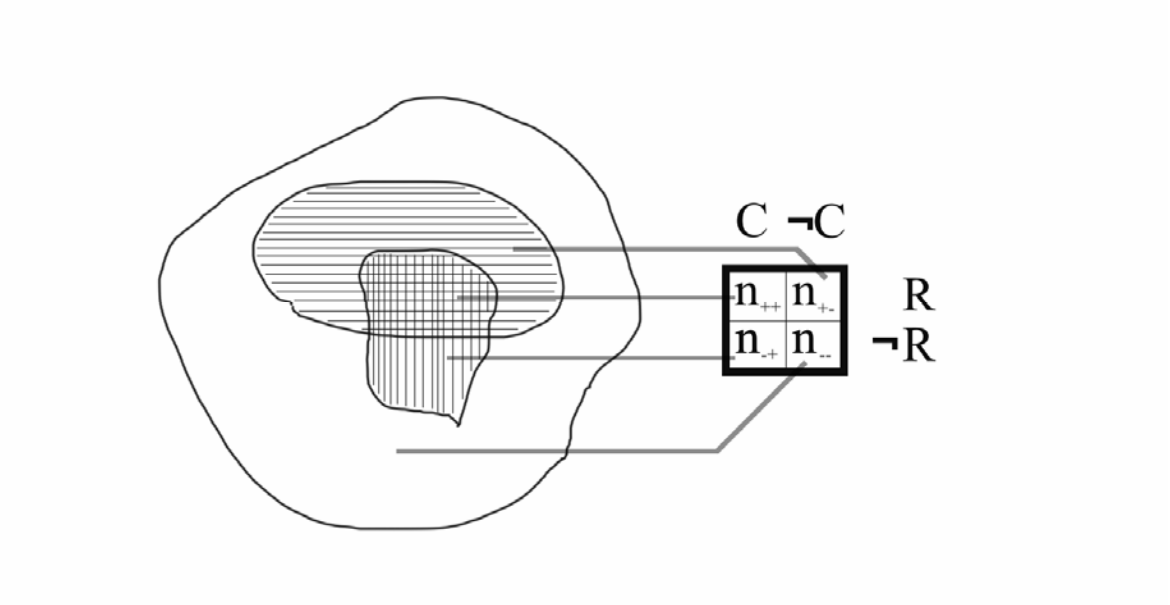
Το σύνολο $[\alpha_k, \beta_k]$ υποδηλώνει τα όρια μέσα στα οποία πρέπει να βρίσκεται η μεταβλητή k του προτύπου. Η εγκυρότητα των παραπάνω κανόνων ορίζεται με ανάλογο τρόπο ως

$$P_i^C = P(x_k \in [\alpha_k, \beta_k] \mid C_i)$$

$$P_i^d = P(C_i \mid x_k \in [\alpha_k, \beta_k])$$

Για την περαιτέρω ανάλυση των μέτρων αξιοπιστίας θεωρούμε ότι έχουμε ένα σύνολο δεδομένων n προτύπων όπου η κατηγορία C_i έχει n_c πρότυπα. Από την εφαρμογή ενός κανόνα σε ένα σύνολο δεδομένων μπορούμε να πάρουμε τις παρακάτω ποσότητες οι οποίες εξηγούνται και γραφικά στο σχήμα 9.

1. n_{++} είναι τα πρότυπα που επαληθεύουν τον κανόνα και ανήκουν στη κατηγορία C_i.
2. n_{+-} είναι τα πρότυπα που επαληθεύουν τον κανόνα και δεν ανήκουν στη κατηγορία C_i.
3. $n_{-+} = n_c - n_{++}$ είναι τα πρότυπα που δεν επαληθεύουν τον κανόνα αλλά ανήκουν στη κατηγορία C_i.
4. $n_{--} = n - n_{++} - n_{+-} - n_{-+} = n - n_c - n_{+-}$ είναι τα πρότυπα που ούτε επαληθεύουν τον κανόνα και ούτε ανήκουν στη κατηγορία C_i.



Σχήμα 9. Σχέση ανάμεσα στον κανόνα R και την κατηγορία C. Η κατηγορία C βρίσκεται στην κάθετα σκιασμένη περιοχή και ο κανόνας R στην οριζόντια σκιασμένη περιοχή.

Με βάση τις παραπάνω ποσότητες μπορούν να οριστούν πολλά διαφορετικά μέτρα αξιολόγησης τα οποία ανάλογα με την έκφραση τους μεγιστοποιούνται ή ελαχιστοποιούνται όταν ο κανόνας περιγράφει τέλεια την κατηγορία και δεν κατηγοριοποιεί λάθος κανένα πρότυπο. Στην ιδανική περίπτωση είναι $n_{+-} = n_{-+} = 0$. Παρακάτω αναφέρονται μερικά από τα κριτήρια που μπορούν να οριστούν.

$$S_1 = \frac{n_{++} + n_{--}}{n_{++} + n_{+-} + n_{-+} + n_{--}}$$

$$S_2 = P_i^d \bullet P_i^C = \frac{n_{++}}{n_{++} + n_{+-}} \bullet \frac{n_{++}}{n_{++} + n_{-+}}$$

$$S_3 = \frac{n_{++}}{n_{++} + n_{+-} + n_{-+}}$$

$$S_4 = \frac{n_{++}}{n_C}$$

$$S_5 = \frac{n_{++}}{n_{+-}}$$

$$S_6 = n_{+-} + n_{-+} = n_{+-} + n_C - n_{++}$$

Το κριτήριο S_1 παρουσιάζει το μειονέκτημα ότι αν τα πρότυπα μιας κατηγορίας είναι πολύ λιγότερα από όλα τα πρότυπα ($n_c << n$) τότε η τιμή του κριτηρίου επηρεάζεται πολύ από την ανάγκη να κατηγοριοποιηθούν τα περισσότερα πρότυπα ως n_{--} και δεν μπορεί να πάρει μεγάλο εύρος τιμών. Επίσης τα n_{--} πρότυπα δεν έχουν κάποιο ουσιαστικό ενδιαφέρον όταν εξετάζεται η σχέση ανάμεσα στον κανόνα R_i και την κατηγορία C_i. Για αυτούς τους λόγους το S_1 δεν χρησιμοποιείται στην πράξη.

Το κριτήριο S_2 είναι το γινόμενο των πιθανοτήτων P_i^C και P_i^d και μπορεί να θεωρηθεί ως μέτρο αμοιβαίας εμπιστοσύνης. Τα κριτήρια S_3, S_4 και S_5 είναι απλούστερες παραλλαγές του S_2 και το S_3 προσεγγίζει το S_2 για $n_{++} >> n_{+-} + n_{-+}$. Το κριτήριο S_6 είναι μέτρο του πλήθους των προτύπων που κατηγοριοποιούνται λάθος.

Υπάρχουν περιπτώσεις που οι συνθήκες του συνόλου κανόνων δημιουργούν επικαλυπτόμενες περιοχές στο χώρο των δεδομένων και υπάρχουν πρότυπα που

επαληθεύουν παραπάνω από έναν κανόνα. Για να λυθεί το πρόβλημα πρέπει να επιλεγεί για αυτά τα πρότυπα σε ποια από τις κατηγορίες θα πρέπει να ενταχθούν. Η επιλογή μπορεί να γίνει με διαφορετικούς τρόπους όπως με τυχαίο τρόπο, να επιλεγεί ο πρώτος κανόνας που επαληθεύεται, να βαθμολογηθούν οι κανόνες ανάλογα με τα πρότυπα που κατηγοριοποιούν ή να επιλεγεί ο κανόνας με τη μικρότερη πιθανότητα λάθους [9]. Η πιθανότητα λάθους (false positive rate) ορίζεται ως

$$FP = \frac{n_{+-}}{n - n_C}$$

και χρησιμοποιείται ώστε να υπάρχει η μικρότερη δυνατή πιθανότητα για λάθος κατηγοριοποίηση προτύπων.

Η ακρίβεια ενός συνόλου κανόνων μπορεί να αυξηθεί κάνοντας πιο αυστηρές τις συνθήκες αλλά με αυτόν τον τρόπο αυξάνονται επίσης τα πρότυπα τα οποία δεν κατηγοριοποιούνται σε καμία κατηγορία και χαρακτηρίζονται ως άγνωστα. Έτσι θα πρέπει να υπάρξει απαραίτητα ένας συμβιβασμός ανάμεσα στην ακρίβεια και τον ρυθμό απόρριψης προτύπων. Ο αριθμός των προτύπων που θεωρούνται άγνωστα είναι ένας ακόμα δείκτης της σημαντικότητας των κανόνων. Ένα σύνολο κανόνων μπορεί να χαρακτηριστεί επίσης και από το μέγεθος του, όσο πιο λίγοι κανόνες υπάρχουν τόσο πιο συμπαγές και κατανοητό είναι.

4.4 Εφαρμογές στην εξόρυξη δεδομένων

Εξόρυξη δεδομένων είναι η μη τετριμμένη ανακάλυψη κρυφής, άγνωστης, σημαντικής και ενδεχομένως χρήσιμης πληροφορίας από δεδομένα [10]. Τα νευρωνικά δίκτυα χρησιμοποιούνται στην εξόρυξη δεδομένων όταν υπάρχει μεγάλος όγκος δεδομένων και ανάγκη για ομαδοποίηση, κατηγοριοποίηση, εκτίμηση χαρακτηριστικών και μείωση των διαστάσεων των δεδομένων. Σε πολλές εφαρμογές εξόρυξης δεδομένων η χρήση νευρωνικών δικτύων και η πρόβλεψη που αυτά προσφέρουν δεν είναι αρκετή αφού η κατανόηση των δεδομένων είναι πολύ σημαντική.

Τα νευρωνικά δίκτυα φτιάχνουν μοντέλα προβλέψεων με υψηλά ποσοστά ακρίβειας προσαρμόζοντας τις εσωτερικές τους παραμέτρους και αποτελούν μη γραμμικά συστήματα με εσωτερικά κατανεμημένες πληροφορίες στα βάρη τους. Γι' αυτό το λόγο τα νευρωνικά δίκτυα θεωρούνται συνήθως ως "μαύρα κουτιά" και χρειάζεται ένα επιπλέον στάδιο επεξεργασίας για να εξαχθούν κανόνες. Τα νευρωνικά δίκτυα έχουν διαφορετικές δυνατότητες αφού εκτελούν υποσυμβολική επεξεργασία δεδομένων και πολλές φορές είναι πιο ισχυρά από ένα σύνολο λογικών κανόνων. Η συνάρτηση που υλοποιούν τα νευρωνικά δίκτυα είναι δύσκολο να κατανοηθεί αλλά μπορεί να απλοποιηθεί και να προσεγγισθεί από σύνολα λογικών κανόνων.

Η μετατροπή της γνώσης που υπάρχει στα νευρωνικά δίκτυα σε ένα σύνολο λογικών κανόνων είναι σημαντική διότι οι κανόνες είναι πολύ πιο κατανοητοί και αποδεκτοί από ανθρώπους σε σχέση με το μεγάλο όγκο αριθμητικών δεδομένων που υπάρχουν σε αυτά. Τα νευρωνικά δίκτυα μερικές φορές αποτυγχάνουν και ιδιαίτερα σε νέα προβλήματα ή σε κρίσιμους τομείς όπως ιατρικές και οικονομικές εφαρμογές μια λανθασμένη πρόβλεψη μπορεί να αποδειχθεί εξαιρετικά επιζήμια. Αντίθετα, η ισχύς των κανόνων μπορεί να επαληθευτεί εύκολα με παρατήρηση από ειδικούς και να αναγνωριστούν τα λάθη που μπορεί να υπάρξουν[11].

Γνώσεις που υπάρχουν για κάποιο πρόβλημα βρίσκονται σε συμβολική μορφή και συνήθως είναι δύσκολο να συνδυαστούν με τα νευρωνικά δίκτυα. Η εξαγωγή γνώσης από

νευρωνικά δίκτυα, ο συνδυασμός της με τη συμβολική γνώση που ήδη υπάρχει και η χρήση των κανόνων σε έμπειρα συστήματα μπορεί να δημιουργήσει συστήματα αυξημένης αξιοπιστίας που θα μπορούν να ανακαλύπτουν σημαντικές αλληλεπιδράσεις των δεδομένων και πληροφορίες για το υπό εξέταση πρόβλημα.

Πολλές φορές τα νευρωνικά δίκτυα αποτελούν ένα κομμάτι μεγαλύτερων συστημάτων που έχουν σκοπό την ανακάλυψη γνώσης σε βάσεις δεδομένων (Knowledge Discovery in Databases, KDD systems). Σκοπών αυτών των συστημάτων είναι ο προσδιορισμός έγκυρων, νέων, χρήσιμων και κατανοητών σχέσεων-προτύπων σε δεδομένα. Σε τέτοια συστήματα τα νευρωνικά δίκτυα υλοποιούν το κομμάτι της εξόρυξης δεδομένων και χρησιμεύουν στην κατηγοριοποίηση δεδομένων και στη δημιουργία κανόνων.

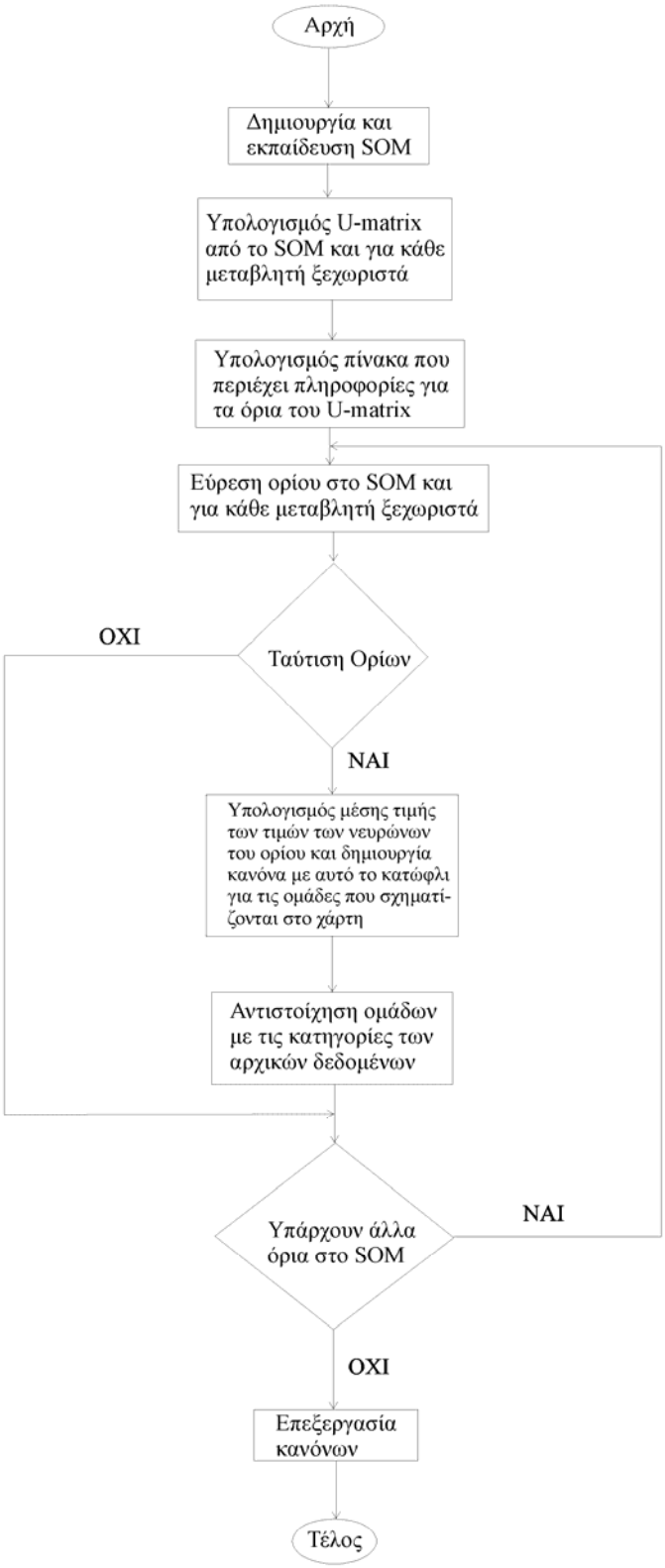
Η χρήση νευρωνικών δικτύων υπερτερεί έναντι άλλων τεχνικών λόγω της έμφυτης δυνατότητας παράλληλης επεξεργασίας και της ικανότητας τους να βοηθούν στη λύση προβλημάτων βελτιστοποίησης στο χώρο των δεδομένων [7]. Εκτός από αυτά τα πλεονεκτήματα, η χρήση ειδικότερα αυτο-οργανούμενων χαρτών για την εξόρυξη δεδομένων προσφέρει κατηγοριοποίηση των δεδομένων και αποδοτική οπτικοποίηση των αποτελεσμάτων ανεξάρτητα του αριθμού των διαστάσεων των δεδομένων. Αν τα πλεονεκτήματα αυτά συνδυαστούν με ένα σύστημα εξαγωγής κανόνων η περιγραφή των δεδομένων μέσω ενός SOM θα είναι ακόμα πιο πλήρης.

5

Εξαγωγή κανόνων με βάση τον U-matrix

Για την δημιουργία κανόνων από αυτό-οργανούμενους χάρτες μπορεί να χρησιμοποιηθεί η ιδιότητά τους να προσφέρουν μια ομαδοποίηση των δεδομένων. Ειδικότερα, ο U-matrix ξεχωρίζει τις ομάδες που σχηματίζονται στο SOM. Ταυτίζοντας τα όρια που σχηματίζονται μεταξύ των ομάδων στον U-matrix με όρια σε πίνακες U-matrix για κάθε μεταβλητή ξεχωριστά μπορούμε να θεωρήσουμε ότι τα όρια της μεταβλητής θα ισχύουν και για τα αρχικά δεδομένα [12]. Από τα όρια των μεταβλητών μπορούν να δημιουργηθούν κανόνες που αν συνδυαστούν μεταξύ τους μπορούν να δημιουργήσουν ένα σύνολο κανόνων που περιγράφει ικανοποιητικά το αρχικό σύνολο δεδομένων. Ο αλγόριθμος (διάγραμμα ροής στο σχήμα 10) είναι ο εξής:

1. Δημιουργία και εκπαίδευση SOM
2. Υπολογισμός U-matrix από το SOM και για κάθε μεταβλητή ξεχωριστά
3. Υπολογισμός πίνακα που περιέχει πληροφορίες για τα όρια του U-matrix
4. Εύρεση ορίου στο SOM και για κάθε μεταβλητή ξεχωριστά
5. Αν υπάρχει ταύτιση τότε υπολογισμός κανόνα στο βήμα 6 αλλιώς πήγαινε στο βήμα 8
6. Υπολογισμός μέσης τιμής των τιμών των νευρώνων του ορίου και δημιουργία κανόνα με αυτό το κατώφλι για τις ομάδες που σχηματίζονται στο χάρτη
7. Αντιστοίχηση ομάδων με τις κατηγορίες των αρχικών δεδομένων
8. Αν υπάρχουν άλλα όρια πήγαινε στο βήμα 4
9. Επεξεργασία κανόνων



Σχήμα 10. Διάγραμμα ροής αλγορίθμου 1

5.1 Δημιουργία και εκπαίδευση SOM

Στο βήμα αυτό γίνεται η εκπαίδευση του SOM και όλες οι αρχικοποιήσεις που χρειάζονται για την εκτέλεση του προγράμματος. Το σύνολο δεδομένων που θα χρησιμοποιηθεί πρέπει να βρίσκεται σε συγκεκριμένη μορφή δυο πινάκων. Ο πρώτος πίνακας έχει μια στήλη για κάθε μεταβλητή και κάθε γραμμή υποδηλώνει ένα πρότυπο. Ο δεύτερος πίνακας περιέχει πληροφορίες για την κατηγορία που ανήκει κάθε πρότυπο του πρώτου πίνακα. Πριν την εκπαίδευση του SOM τα δεδομένα κανονικοποιούνται έτσι ώστε να έχουν μηδενική μέση τιμή και μοναδιαία τυπική απόκλιση. Το μέγεθος του χάρτη μπορεί να καθοριστεί αυτόματα ή να ορισθεί από το χρήστη, στην εφαρμογή χρησιμοποιήθηκε η αυτόματη επιλογή. Σε σύνολα δεδομένων που περιέχουν πολλές διαφορετικές κατηγορίες είναι προτιμότερο το μέγεθος του χάρτη να είναι μεγάλο.

Στη συνέχεια κάθε νευρώνας στο χάρτη παίρνει μια ετικέτα ανάλογα με ποιας κατηγορίας τα πρότυπα κατηγοριοποιούνται σε αυτόν. Για να γίνει αυτό υπολογίζεται ποιος νευρώνας βρίσκεται πιο κοντά σε κάθε πρότυπο και κάθε νευρώνας παίρνει μια ετικέτα σύμφωνα με το πλήθος των προτύπων που κατηγοριοποιεί. Ο νευρώνας μπορεί να πάρει και περισσότερες ετικέτες αν κατηγοριοποιεί πολλά πρότυπα από διαφορετικές ομάδες. Για την εφαρμογή επιλέχθηκε κάθε νευρώνας να έχει μία μόνο ετικέτα που να δείχνει σε ποια κατηγορία ανήκει η πλειοψηφία των προτύπων αγνοώντας τις κατηγορίες των υπόλοιπων προτύπων. Αυτή είναι η απλούστερη επιλογή και μπορεί να οδηγήσει και σε λανθασμένα συμπεράσματα αν σε μεγάλο μέρος του χάρτη υπάρχουν νευρώνες που κατηγοριοποιούν σχεδόν τον ίδιο αριθμό προτύπων από διαφορετικές κατηγορίες. Για να μην υπάρξουν τέτοια προβλήματα τα στάδια της δημιουργίας και επεξεργασίας κανόνων υλοποιήθηκαν έτσι ώστε να καλύπτουν και αυτές τις περιπτώσεις.

5.2 Υπολογισμός U-matrix από το SOM και για κάθε μεταβλητή ξεχωριστά

Ο υπολογισμός του U-matrix από το SOM γίνεται με τον τρόπο που περιγράφηκε στο κεφάλαιο 3. Η κατασκευή του U-matrix για μία μόνο μεταβλητή γίνεται με τον ίδιο τρόπο αλλά μηδενίζοντας κάθε φορά την συμμετοχή όλων των μεταβλητών στον υπολογισμό εκτός από μιας. Έτσι δημιουργούνται πίνακες ίσων διαστάσεων με τον αρχικό U-matrix που δείχνουν την συνεισφορά κάθε μεταβλητής ξεχωριστά.

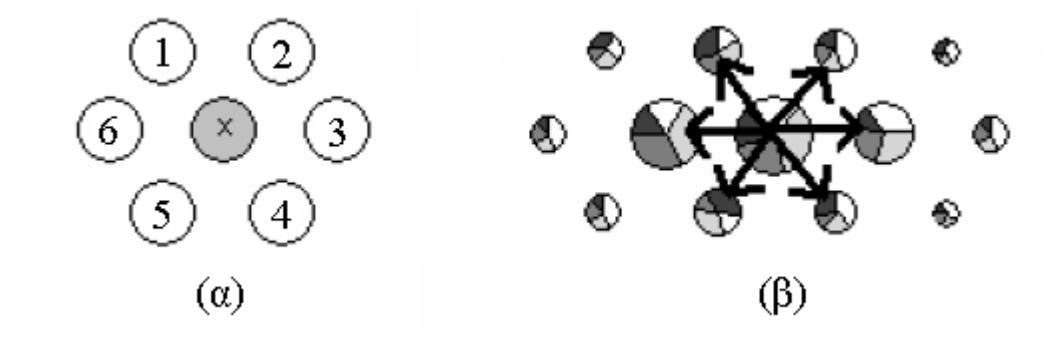
5.3 Υπολογισμός πίνακα που περιέχει πληροφορίες για τα όρια του U-matrix

Βασική ιδιότητα του U-matrix είναι ο διαχωρισμός ομάδων νευρώνων που έχουν κοινά χαρακτηριστικά. Για την εφαρμογή αυτή όμως μας ενδιαφέρουν περισσότερο τα όρια που δημιουργούνται μεταξύ των ομάδων παρά οι ίδιες οι ομάδες. Για το λόγο αυτό δημιουργούμε έναν νέο πίνακα που περιέχει πληροφορίες για τα όρια του U-matrix. Ο πίνακας αυτός ονομάζεται πίνακας διαφοράς ορίων (Boundary Difference Value matrix – BDV) [12].

Σε κάθε νευρώνα του SOM αντιστοιχίζεται μια τιμή που δηλώνει κατά πόσο ο νευρώνας αυτός συμμετέχει σε κάποιο όριο, όσο μεγαλύτερη είναι αυτή η τιμή τόσο πιο ευδιάκριτο είναι το όριο αυτό. Για να βρούμε αυτήν την τιμή εξετάζουμε τα όρια που δημιουργεί ο νευρώνας αυτός σε κάθε δυνατή διεύθυνση στο χάρτη όπως φαίνεται στο σχήμα 11. Τα πιθανά όρια του νευρώνα συγκρίνονται μεταξύ τους και όποιο έχει τη μεγαλύτερη σχετική διαφορά θεωρείται ότι είναι υποψήφιο ως μέρος μεγαλύτερου ορίου και η τιμή του αντιπροσωπεύει το νευρώνα. Η σχετική διαφορά μεταξύ των νευρώνων υπολογίζεται από τον τύπο:

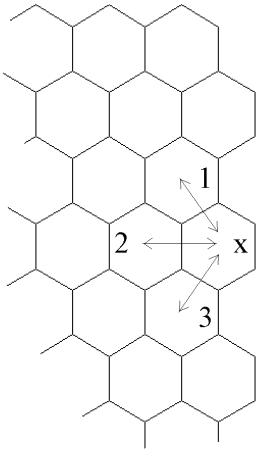
$$BDV = \frac{M_L - M_O}{R_O}$$

όπου M_L είναι η μέση τιμή των αποστάσεων του υπό εξέταση νευρώνα με τους νευρώνες που συμμετέχουν στο όριο, M_O είναι η μέση τιμή των αποστάσεων του υπό εξέταση νευρώνα με τους υπόλοιπους γειτονικούς νευρώνες και R_O είναι το εύρος των αποστάσεων των υπόλοιπων γειτονικών νευρώνων. Το εύρος των αποστάσεων είναι η μέγιστη απόσταση μεταξύ του υπό εξέταση νευρώνα και κάποιου γειτονικού του που δε συμμετέχει στο όριο μείον την ελάχιστη απόσταση του υπό εξέταση νευρώνα και κάποιου γειτονικού που δε συμμετέχει στο όριο.



Σχήμα 11. (α) Γειτονικοί νευρώνες σε εξαγωνικό πλέγμα, **(β)** Πιθανά όρια σε όλες τις διευθύνσεις

Από τον παραπάνω ορισμό προκύπτει πρόβλημα στα άκρα του χάρτη γιατί εκεί δεν υπάρχουν αρκετοί νευρώνες ώστε να εξεταστούν όλες οι πιθανές διευθύνσεις. Ακόμα, αν θεωρήσουμε στις άκρες του χάρτη ότι τα υποψήφια όρια μπορούν να αποτελούνται από μόνο δύο νευρώνες όπως στο σχήμα 12 τότε το πλήθος των νευρώνων που δεν συμμετέχουν είναι σε κάποιες περιπτώσεις επίσης δύο. Με μόνο δυο νευρώνες αυξάνεται η πιθανότητα η απόσταση του υπό εξέταση νευρώνα από αυτούς τους δύο νευρώνες να είναι σχεδόν ίδια. Σε αυτές τις περιπτώσεις ο παρανομαστής R_O γίνεται πολύ μικρός και η τιμή του BDV πολύ μεγάλη χωρίς όμως να δείχνει αυτό κάποιο πραγματικό όριο στο χάρτη.



Σχήμα 12. Πιθανά όρια στα άκρα του SOM

Κατά τον σχηματισμό των ορίων μεγάλη τιμή BDV σημαίνει σημαντικό όριο οπότε το λάθος που περιγράφηκε παραπάνω μπορεί να οδηγήσει τον αλγόριθμο στον σχηματισμό λάθος όριων. Κατά την εφαρμογή παρουσιάστηκαν περιπτώσεις όπου ενώ στο εσωτερικό του χάρτη οι τιμές BDV ήταν από 0 έως 1.5 στα άκρα υπήρχαν τιμές μέχρι 150 δηλαδή πολλές τάξεις μεγέθους μεγαλύτερες. Από την εξέταση του U-matrix στα σημεία αυτά βρέθηκε ότι όχι μόνο δε δικαιολογούνται τόσο υψηλές τιμές αλλά δεν υπάρχουν καν όρια στα σημεία αυτά.

Για να μην επηρεάζουν αυτές οι τιμές τα όρια στο χάρτη μπορούμε να βάλουμε ένα όριο πάνω από το οποίο θα θεωρείται ότι η τιμή BDV είναι λάθος και δεν πρέπει ο συγκεκριμένος νευρώνας να πάρει μέρος στην αναζήτηση ορίων. Από την άλλη μεριά, αφού οι υψηλές αυτές τιμές εμφανίζονται μόνο στα άκρα μπορούμε να υποθέσουμε ότι το όριο έχει ήδη σχηματιστεί κατά το μεγαλύτερο μέρος του και ο νευρώνας που θα καταλήξει δε θα παίξει και τόσο σημαντικό ρόλο στον κανόνα αφού στον σχηματισμό του κανόνα λαμβάνονται υπόψη όλοι οι νευρώνες.

Ένας άλλος τρόπος για να αποφευχθεί αυτό το πρόβλημα είναι να χρησιμοποιηθεί ένας εναλλακτικός ορισμός για το BDV όπου δε θα υπάρχει το εύρος των αποστάσεων των νευρώνων δηλαδή:

$$BDV = M_L - M_O$$

Με χρήση αυτού του τύπου σε όλο το μήκος του χάρτη υπάρχουν συγκρίσιμες τιμές BDV. Καθεμία από αυτές τις παραλλαγές δίνει διαφορετικά αποτελέσματα που αναλύονται στο κεφάλαιο 7. Η υλοποίηση των μεθόδων γίνεται στα BDVMatrixcalcRatio για τον πρώτο τύπο υπολογισμού και BDVMatrixcalcDiff για το δεύτερο.

5.4 Εύρεση ορίων

Ως όριο στο χάρτη θεωρούμε κάθε γραμμή που αρχίζοντας από ένα νευρώνα σε κάποια άκρη του χάρτη τελειώνει σε έναν άλλο νευρώνα που βρίσκεται επίσης σε άκρη και με αυτόν τον τρόπο χωρίζει το χάρτη σε δυο μέρη. Στην παρούσα εφαρμογή δε γίνεται αναζήτηση για κλειστά όρια που βρίσκονται αποκλειστικά στο εσωτερικό του χάρτη αφού αυτά μπορούν να προσεγγιστούν με καλή επιτυχία συνδυάζοντας όλα τα όρια από τις άκρες.

Η αναζήτηση των ορίων γίνεται αρχίζοντας από τον νευρώνα που βρίσκεται σε άκρη και έχει τη μεγαλύτερη τιμή BDV. Στη συνέχεια βρίσκονται όλοι οι γειτονικοί

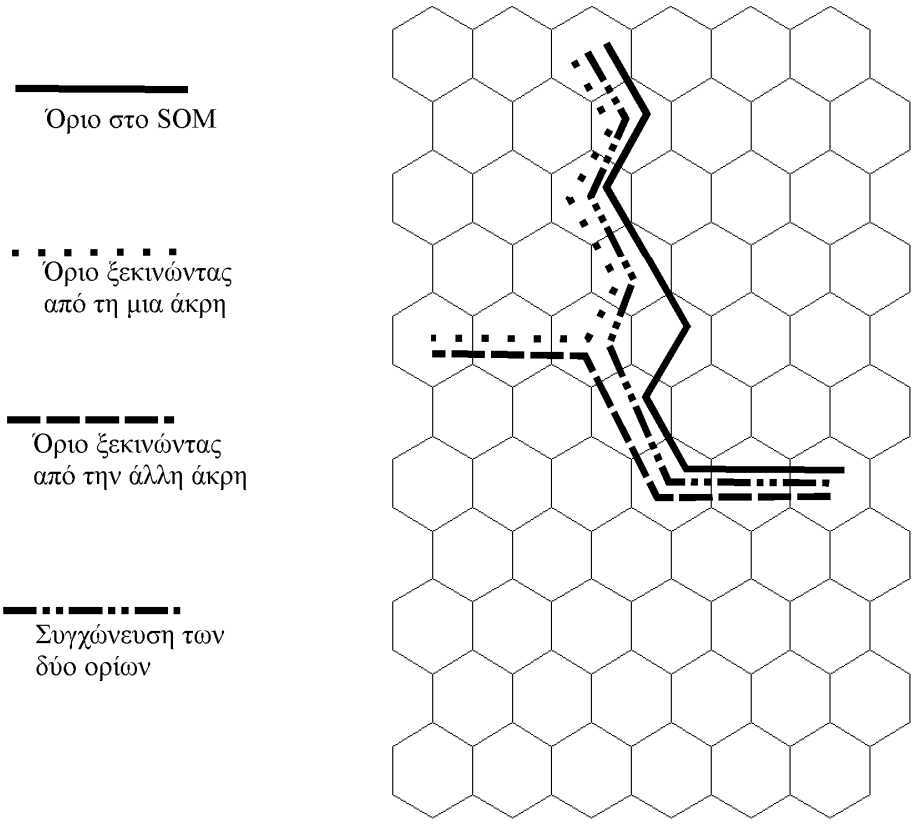
νευρώνες και επιλέγεται ως υποψήφιος για το όριο αυτός με τη μεγαλύτερη τιμή BDV. Αν αυτή η τιμή είναι κατά πολύ μεγαλύτερη από την τιμή του προηγούμενου νευρώνα μπορεί να αγνοηθεί για τους λόγους που εξηγήθηκαν στην προηγούμενη παράγραφο και να επιλεγεί ο επόμενος μεγαλύτερος γειτονικός νευρώνας. Επίσης γίνεται έλεγχος για το αν ο υποψήφιος νευρώνας είναι ήδη μέρος του ορίου. Αν συμβαίνει αυτό σημαίνει ότι ο σχηματισμός του ορίου πηγαίνει ξανά προς τα πίσω οπότε επιλέγεται άλλος υποψήφιος. Τέλος, ελέγχεται αν ο υποψήφιος είναι κατά πολύ μικρότερος από τον προηγούμενο νευρώνα. Σε αυτή την περίπτωση θεωρούμε ότι το όριο που πήγαινε να σχηματιστεί σταματάει αφού πουθενά γύρω του δεν υπάρχει κάποιος νευρώνας με αρκετά υψηλή τιμή BDV για υποψήφιο όριο.

Αν όλες οι παραπάνω προϋποθέσεις ικανοποιούνται τότε ο υποψήφιος νευρώνας θεωρείται ότι συμμετέχει στο όριο και η αναζήτηση συνεχίζεται ψάχνοντας τους γειτονικούς του νέου νευρώνα μέχρι το όριο να φτάσει σε κάποια άκρη του χάρτη και να χωρίσει δυο περιοχές σε αυτόν. Η διαδικασία συνεχίζεται επιλέγοντας τον νευρώνα που βρίσκεται σε άκρη και έχει την επόμενη μεγαλύτερη τιμή BDV και τελειώνει όταν έχουν αρχίσει όρια από όλους τους νευρώνες των άκρων.

Πριν αρχίσει η αναζήτηση του ορίου μηδενίζουμε την τιμή BDV των δυο ακριανών νευρώνων που γειτονεύουν με τον νευρώνα που αρχίζει το όριο. Αυτό γίνεται γιατί πολλές φορές επιλέγεται ως επόμενος νευρώνας ένας από τους δύο ακριανούς και ο αλγόριθμος σταματάει απότομα. Η εύρεση ενός τέτοιου ορίου δεν έχει κάποιο νόημα αφού όρια κατά μήκος της άκρης του χάρτη δε χωρίζουν δυο διαφορετικές περιοχές. Αν παρόλα αυτά αφήσουμε την αναζήτηση να προχωρήσει θα οδηγηθούμε στο σχηματισμό ενός ορίου που έχει ήδη βρεθεί ή θα βρεθεί στη συνέχεια της εκτέλεσης του αλγορίθμου, αφού όλοι οι ακριανοί νευρώνες χρησιμοποιούνται ως αρχικά σημεία για την αναζήτηση ορίων.

Όταν έχει σχηματιστεί κάποιο έγκυρο όριο στο SOM τότε γίνεται αναζήτηση για όμοιο όριο στους U-matrix κάθε μεταβλητής. Η αναζήτηση δε χρειάζεται να γίνει σε ολόκληρο τον χάρτη αλλά αρκεί να εξετασθούν οι περιοχές που βρίσκονται κοντά στους νευρώνες που αποτελούν και το όριο του SOM. Έτσι η αναζήτηση ξεκινάει από τα δυο άκρα του ορίου του SOM και γίνεται με παρόμοιο τρόπο με προηγουμένως. Υπάρχει επίσης προσοχή ώστε τα όρια που σχηματίζονται να μην κινούνται στις άκρες του χάρτη και κάθε φορά ελέγχεται το μήκος και η θέση του ορίου για να βρεθεί αν αυτό είναι έγκυρο ή όχι.

Υπάρχουν περιπτώσεις που τα ενώ η αναζήτηση ορίων στο χάρτη της μεταβλητής ακολουθεί το όριο του SOM αρχίζοντας και από τα δύο άκρα του, από ένα κοινό ή από δύο γειτονικούς νευρώνες και μετά το όριο στο χάρτη της μεταβλητής ακολουθεί τελείως άλλη πορεία όπως φαίνεται στο σχήμα 13. Έτσι ενώ ουσιαστικά υπάρχει όριο που είναι αρκετά όμοιο με αυτό του SOM η παρουσία ενός νευρώνα με μεγάλη τιμή BDV που υποδηλώνει ένα άλλο όριο είναι ικανή να αποπροσανατολίζει την αναζήτηση και ο αλγόριθμος να μη δίνει τα σωστά αποτελέσματα αλλά ένα όριο που ταυτίζεται μόνο κατά ένα μικρότερο μέρος. Για να αντιμετωπιστεί αυτό το πρόβλημα τα δύο όρια που δημιουργούνται με αφετηρία τα δύο άκρα του ορίου στο SOM μπορούν να συγχωνευτούν. Η συγχώνευση γίνεται ακολουθώντας το ένα όριο και μόλις σε κάποιο νευρώνα συναντηθεί το δεύτερο όριο συνεχίζουμε ακολουθώντας το δεύτερο όριο μέχρι την άκρη του χάρτη. Η διαδικασία αυτή χρησιμοποιείται και σε περιπτώσεις που τα δύο όρια ακολουθούν κοινούς νευρώνες.



Σχήμα 13. Συγχώνευση ορίων σε πίνακα μεταβλητής

Όταν βρεθούν κάποια έγκυρα όρια στο χάρτη της μεταβλητής συγκρίνονται με το όριο στο SOM και αν κάποιο από αυτά είναι αρκετά όμοιο θεωρούμε ότι η μεταβλητή αυτή παίζει μεγάλο ρόλο στο σχηματισμού του ορίου στο SOM και προχωράμε στη δημιουργία κανόνα αλλιώς συνεχίζουμε με την επόμενη μεταβλητή. Η σύγκριση των δύο ορίων γίνεται με βάση το μήκος του μεγαλύτερου και εξετάζονται ένας ένας οι νευρώνες για να βρεθεί αν αποτελούν μέρος και του άλλου ορίου. Το ποσοστό ταύτισης από το οποίο και πάνω θεωρείται ότι τα δύο όρια είναι αρκετά όμοιο είναι μια επιλογή του χρήστη. Όσο μεγαλύτερο είναι αυτό το ποσοστό τόσο λιγότερα όρια ταυτίζονται με αποτέλεσμα και λιγότερους κανόνες.

Όταν υπάρχει αυστηρότητα στην ταύτιση των ορίων επιλέγονται μόνο τα όρια που σχηματίζονται σε παρόμοιες περιοχές του χάρτη και του χάρτη της μεταβλητής. Σε σύνολα δεδομένων πολλών μεταβλητών είναι δύσκολο μια μεταβλητή να είναι τόσο σημαντική που τα όρια της να υπάρχουν αυτούσια και στο SOM. Στη γενική περίπτωση τα όρια στο SOM οφείλονται στο συνδυασμό των μεταβλητών. Έτσι, αν τα όρια πρέπει να ταυτίζονται αυστηρά μόνο ελάχιστα από αυτά θα επιλεγούν και δε θα είναι αρκετά για να δημιουργηθούν κανόνες που να περιγράφουν σωστά τα δεδομένα.

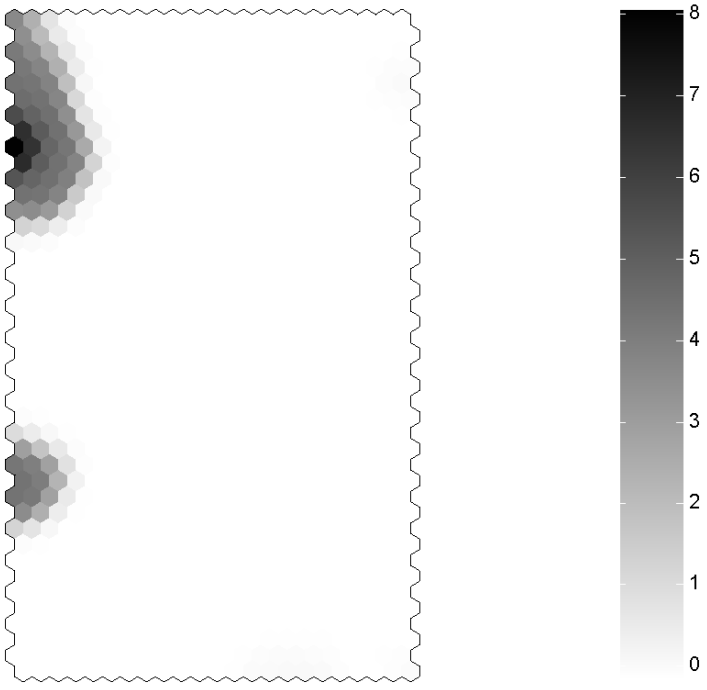
Για να επιλέγονται περισσότεροι κανόνες μπορεί να μειωθεί το ποσοστό της ταύτισης των ορίων. Όσο μειώνεται αυτό το ποσοστό τόσο περισσότερη βαρύτητα δίνεται στα όρια των μεταβλητών σε βάρος των ορίων του SOM. Με αυτό τον τρόπο επιλέγονται όλα τα "ευδιάκριτα" όρια στους χάρτες των μεταβλητών και χρησιμοποιούνται για την παραγωγή κανόνων. Το γεγονός ότι αυτά δεν είναι πάντα σε μεγάλη αντιστοιχία με τα όρια στο SOM δε σημαίνει ότι δεν είναι σημαντικά αφού τα όρια των μεταβλητών μπορεί να ισχύουν κατευθείαν για τα δεδομένα ή ακόμα και να συμμετέχουν στο σχηματισμό δευτερευόντων ορίων στο SOM που η αναζήτηση δεν μπορεί να εντοπίσει γιατί ακολουθεί άλλα πιο ευδιάκριτα όρια.

Η υλοποίηση γίνεται στη συνάρτηση FindGeneralBoundaries για το όριο στο γενικό πίνακα BDV και στη συνάρτηση FindComponentBoundary για το όριο στον πίνακα BDV μίας μεταβλητής.

5.5 Δημιουργία κανόνων

Κάθε όριο που ταυτοποιείται στο SOM και στο χάρτη μιας μεταβλητής μπορεί να αποτελέσει τη βάση για τη δημιουργία ενός κανόνα. Ο κανόνας αυτός θα ισχύει για τις δύο περιοχές που δημιουργούνται στο χάρτη από το όριο. Η τιμή της συνθήκης του κανόνα μπορεί να βρεθεί από τους νευρώνες που βρίσκονται πάνω στο όριο. Ο μέσος όρος των τιμών όλων των νευρώνων κατά μήκος του ορίου στο χάρτη της μεταβλητής είναι η τιμή της συνθήκης στον κανόνα. Η εξαγωγή του μέσου όρου γίνεται στη συνάρτηση BoundaryThrehold.

Μετά την εξαγωγή του μέσου όρου πρέπει να βρεθεί ποιες περιοχές έχουν μικρότερες τιμές από αυτή και ποιες μεγαλύτερη. Σε κάθε περιοχή του χάρτη που ορίζει το όριο λαμβάνεται ο μέσος όρος των τιμών όλων των νευρώνων. Αν αυτός ο μέσος όρος είναι μεγαλύτερος από την τιμή της συνθήκης τότε για την περιοχή αυτή ο κανόνας ισχύει για τιμές μεγαλύτερες της συνθήκης αλλιώς για μικρότερες. Η χρήση του μέσου όρου σε μεγάλη περιοχή του χάρτη και μεγάλο αριθμό νευρώνων κρύβει κινδύνους γιατί μικρές ομάδες νευρώνων μπορεί να έχουν πολύ διαφορετικές τιμές από την υπόλοιπη περιοχή και έτσι ο κανόνας να μην ισχύει για αυτές. Στο σχήμα 14 αν ένα όριο του SOM οδηγήσει στη δημιουργία ορίου στον πίνακα της μεταβλητής για την μικρή περιοχή που βρίσκεται κάτω αριστερά ο κανόνας που θα δημιουργηθεί είναι πιθανό να μην περιγράφει σωστά την περιοχή που βρίσκεται πάνω αριστερά αφού η επίδραση του υπόλοιπου χάρτη που είναι ομοιογενής μπορεί να επηρεάσει πολύ περισσότερο τον μέσο όρο. Τέτοιες περιπτώσεις μπορούν να ανακαλυφθούν και να διορθωθούν στο στάδιο επεξεργασίας των κανόνων.



Σχήμα 14. Επίδραση χρήσης μέσου όρου σε μικρές ομάδες νευρώνων

Στη συνέχεια πρέπει να γίνει αντιστοίχηση των δύο περιοχών που υπάρχουν στο χάρτη με τις κατηγορίες των αρχικών δεδομένων. Η αντιστοίχηση αυτή γίνεται με τη βοήθεια των ετικετών που υπάρχουν σε κάθε νευρώνα. Αρχικά γίνεται καταμέτρηση των ετικετών σε όλο το SOM αλλά και στις δυο περιοχές. Η αντιστοίχηση γίνεται σε δύο περιπτώσεις. Η πρώτη περίπτωση είναι μέσα στην περιοχή του χάρτη να υπάρχουν οι περισσότερες από τις ετικέτες μιας κατηγορίας σύμφωνα με ένα προκαθορισμένο ποσοστό. Η δεύτερη περίπτωση είναι οι ετικέτες μιας κατηγορίας να είναι η πλειοψηφία σε μια περιοχή του χάρτη. Ακόμα και αν οι ετικέτες αυτές είναι πολύ λιγότερες από το σύνολο της κατηγορίας μπορούν να δημιουργηθούν κανόνες που να περιγράφουν καλά ένα μέρος των δεδομένων αυτής της κατηγορίας αποκλείοντας παράλληλα δεδομένα άλλων κατηγοριών.

Το αποτέλεσμα αυτής της διαδικασίας είναι η δημιουργία μεγάλου αριθμού κανόνων για κάθε κατηγορία. Οι κανόνες αυτοί έχουν από ένα κατηγόρημα. Για την απαλοιφή λανθασμένων κανόνων και για το συνδυασμό των καλύτερων κανόνων κάθε κατηγορίας είναι απαραίτητο ένα επιπλέον στάδιο επεξεργασίας κανόνων που είναι το αντικείμενο της επόμενης παραγράφου. Η υλοποίηση της κατασκευής των κανόνων γίνεται στις συναρτήσεις RuleSections και IrisRuleCreation, IonRuleCreation SegRuleCreation αφού για κάθε σύνολο δεδομένων υπάρχει διαφορετικός αριθμός κατηγοριών και ετικετών.

5.6 Επεξεργασία κανόνων

Οι κανόνες που κατασκευάζονται με την παραπάνω διαδικασία σε λίγες μόνος περιπτώσεις καταφέρνουν να περιγράψουν με ακρίβεια και αποκλειστικά μία κατηγορία δεδομένων. Πιο αποδοτική περιγραφή μπορεί να γίνει με συνδυασμό των κανόνων. Ο συνδυασμός των κανόνων μπορεί να γίνει με δύο τρόπους, με σύζευξη (AND) ή διάζευξη (OR). Βρέθηκε ότι οι περισσότεροι κανόνες εκτός από την κατηγορία που περιγράφουν επαληθεύονται και από αρκετά άλλα πρότυπα. Για το λόγο αυτό γίνεται συνδυασμός κανόνων κυρίως με σύζευξη. Κανόνες που δημιουργήθηκαν στην περίπτωση που οι ετικέτες μιας κατηγορίας ήταν η πλειοψηφία σε μια περιοχή του χάρτη μπορούν να συνδυαστούν με διάζευξη αν επαληθεύονται από λίγα λάθος πρότυπα

Κάθε κανόνας εφαρμόζεται στο αρχικό σύνολο δεδομένων για να καταμετρηθούν πόσα και ποιας κατηγορίας πρότυπα τον επαληθεύουν. Αν ο κανόνας δεν περιγράφει σωστά ούτε τα μισά πρότυπα της κατηγορίας η συνθήκη του αντιστρέφεται και θεωρούμε ότι ισχύει πλέον ο νέος κανόνας. Αυτό γίνεται ώστε να διορθωθούν τυχόν λάθη που οφείλονται στη χρήση μέσου όρου για την κατασκευή του κανόνα. Οι ποσότητες που υπολογίζονται είναι οι n_{++} και n_{+-} αφού όπως αναφέρθηκε στο κεφάλαιο 4 αρκούν για να υπολογιστούν τα μέτρα αποδοτικότητας. Το μέτρο που χρησιμοποιήθηκε είναι το S_2.

Ανάμεσα στους απλούς κανόνες υπάρχουν κάποιοι που περιγράφουν μια κατηγορία πλήρως. Οι κανόνες αυτοί συνδέονται με AND και σχηματίζουν τον κανόνα της κατηγορίας αυτής. Αν ο κανόνας που προκύπτει δεν είναι ικανοποιητικός τότε πρέπει να εφαρμοστούν και άλλοι κανόνες. Οι υπόλοιποι κανόνες μπορούν να συνδυαστούν με δύο διαφορετικούς τρόπους που καταλήγουν σε διαφορετικά σύνολα κανόνων.

Στον πρώτο τρόπο οι κανόνες ταξινομούνται με βάση την ποσότητα n_{++} και αν υπάρχει ισότητα και με βάση την n_{+-}. Οι κανόνες με καλύτερη περιγραφή της κατηγορίας, δηλαδή μεγάλο n_{++} και μικρό n_{+-} συνδυάζονται ένας κάθε φορά στον κανόνα της κατηγορίας που έχει κατασκευαστεί μέχρι εκείνη τη στιγμή μέχρι ο κανόνας να μην κατηγοριοποιεί καθόλου λάθος πρότυπα οπότε και δεν υπάρχει η ανάγκη προσθήκης

άλλων κανόνων. Αν η εφαρμογή ενός κανόνα χειροτερεύει τα αποτελέσματα που υπήρχαν μέχρι εκείνη τη στιγμή τότε ο κανόνας αυτός απορρίπτεται. Στο δεύτερο τρόπο οι κανόνες ταξινομούνται με βάση το κριτήριο S_2 και συνδυάζονται όπως και στην πρώτη περίπτωση.

Με αυτή τη διαδικασία κατασκευάζονται δυο διαφορετικά σύνολα κανόνων που έχουν έναν κανόνα για κάθε κατηγορία του αρχικού συνόλου δεδομένων. Οι κανόνες αυτοί αποτελούνται από ένα ή περισσότερα κατηγορήματα το πλήθος των οποίων είναι πάντα κατά πολύ μικρότερο από το πλήθος των κανόνων που είχαν κατασκευαστεί αρχικά. Για κάθε κατηγορία επιλέγεται ο ένας από τους δύο κανόνες που την περιγράφει πιο καλά και έτσι δημιουργείται το τελικό σύνολο κανόνων. Η υλοποίηση γίνεται στις συναρτήσεις RulePostProccessing1 και RulePostProccessing2.

6

Εξαγωγή κανόνων με βάση στατιστικές ιδιότητες του SOM

Βασική ιδιότητα ενός αυτο-οργανούμενου χάρτη είναι ότι όμοια πρότυπα εισόδου ταξινομούνται στον ίδιο ή σε γειτονικούς νευρώνες. Μελετώντας τα βάρη των νευρώνων σε ομάδες γειτονικών νευρώνων μπορούν να υπολογιστούν διάφορες σημαντικές στατιστικές ιδιότητες (π.χ. μέση τιμή, τυπική απόκλιση) που μπορούν να συσχετισθούν με τα πρότυπα που περιγράφονται από αυτούς τους νευρώνες και έτσι να εξαχθούν συμπεράσματα και κανόνες που να περιγράφουν τα αρχικά δεδομένα. Ο αλγόριθμος είναι ο εξής:

1. Δημιουργία και εκπαίδευση SOM
2. Σχηματισμός ομάδων στο SOM (Clustering)
3. Υπολογισμός στατιστικών ιδιοτήτων για κάθε ομάδα
4. Επιλογή σημαντικών μεταβλητών και δημιουργία κανόνων για κάθε ομάδα
5. Αντιστοίχηση ομάδων με τις κατηγορίες των αρχικών δεδομένων
6. Επεξεργασία κανόνων

Καθένα από τα παραπάνω βήματα μπορεί να υλοποιηθεί με παραπάνω από έναν τρόπους οδηγώντας σε διαφορετικά αποτελέσματα. Κάποια γενικά βέλτιστη μέθοδος δεν μπορεί να υπάρξει αφού διαφορετικά σύνολα δεδομένων μπορούν να παρουσιάζουν τέτοιες χαρακτηριστικές ιδιότητες που να διευκολύνουν ή να δυσκολεύουν την επεξεργασία τους με μία συγκεκριμένη μέθοδο. Για το λόγο αυτό κάθε βήμα του αλγορίθμου έχει υλοποιηθεί σε διαφορετική συνάρτηση ώστε να μπορούν να μελετηθούν εύκολα οι διάφορες παραλλαγές του βασικού αλγορίθμου και να συγκριθούν τα αποτελέσματα τους. Παρακάτω αναλύεται κάθε βήμα του αλγορίθμου ξεχωριστά.

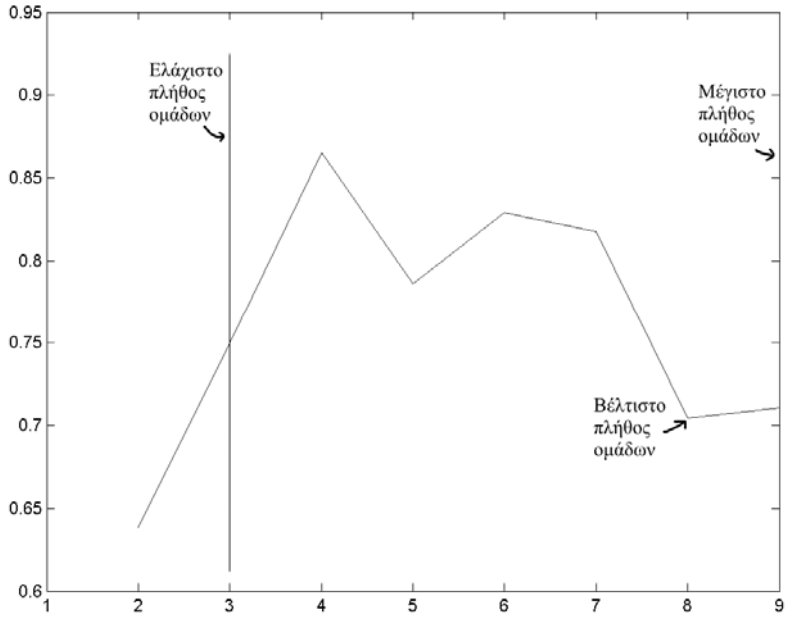
6.1 Δημιουργία και εκπαίδευση SOM

Για την δημιουργία και την εκπαίδευση του SOM ακολουθείται η ίδια διαδικασία με αυτή την προηγούμενης μεθόδου. Τα δεδομένα πριν τη δημιουργία του χάρτη κανονικοποιούνται έτσι ώστε να έχουν μηδενική μέση τιμή και μοναδιαία τυπική απόκλιση. Με αυτόν τον τρόπο οι τιμές για όλες τις μεταβλητές θα κυμαίνονται στο ίδιο εύρος και θα είναι άμεσα συγκρίσιμες μεταξύ τους. Το μέγεθος του χάρτη μπορεί να καθοριστεί αυτόματα ή να ορισθεί από το χρήστη. Σε σύνολα δεδομένων που περιέχουν πολλές διαφορετικές κατηγορίες είναι προτιμότερο το μέγεθος του χάρτη να είναι μεγάλο. Κάθε νευρώνας στο χάρτη παίρνει μια ετικέτα ανάλογα με ποιας κατηγορίας τα πρότυπα κατηγοριοποιούνται περισσότερο σε αυτόν. Τα διανύσματα βαρών των νευρώνων του χάρτη χρησιμοποιούνται στην ομαδοποίηση και την επιλογή των μεταβλητών και μετά από την αποκανονικοποίηση χρησιμοποιούνται στην εξαγωγή των κανόνων.

6.2 Σχηματισμός ομάδων στο SOM (Clustering)

Στο βήμα αυτό υπολογίζονται ομάδες νευρώνων στο χάρτη. Για την ομαδοποίηση χρησιμοποιούνται οι τεχνικές που περιγράφηκαν στο κεφάλαιο 3, πιο συγκεκριμένα ένας συσσωρευτικός ιεραρχικός αλγόριθμος και ο αλγόριθμος των κ-μέσων. Και στις δυο περιπτώσεις ορίζεται άνω και κάτω όριο στο πλήθος των ομάδων. Κάτω όριο είναι ο αριθμός των κατηγοριών που ανήκουν τα αρχικά δεδομένα και άνω όριο η τετραγωνική ρίζα του αριθμού των νευρώνων του χάρτη.

Ο ιεραρχικός αλγόριθμος κατασκευάζει το δενδρόγραμμα με τις συνενώσεις των ομάδων. Διασχίζοντας το δενδρόγραμμα μπορεί να βρεθεί η βάση της ομαδοποίησης σε κάθε βήμα του αλγορίθμου και να υπολογισθεί ο δείκτης αξιολόγησης Davies-Bouldin για το εύρος του πλήθους των ομάδων που μας ενδιαφέρει. Η ελάχιστη τιμή του δείκτη δείχνει και την καλύτερη ομαδοποίηση που γίνεται στο χάρτη όπως φαίνεται στο σχήμα 15.



Σχήμα 15. Διάγραμμα δείκτη αξιολόγησης Davies-Bouldin

Η αξιολόγηση της ομαδοποίησης που προκύπτει από τον αλγόριθμο των κ-μέσων γίνεται με τον ίδιο τρόπο. Ο αλγόριθμος των κ-μέσων εκτελείται συνολικά 20 φορές, με τυχαία αρχικοποίηση, για κάθε κ μέχρι το άνω όριο που έχει ορισθεί και επιλέγεται η καλύτερη σύμφωνα με το τετραγωνικό σφάλμα. Έτσι ελαχιστοποιείται η επίδραση της τυχαίας αρχικοποίησης με παράλληλη όμως αύξηση του υπολογιστικού κόστους, ιδιαίτερα σε μεγάλους χάρτες. Η ομαδοποίηση με τον αλγόριθμο των κ-μέσων υλοποιείται στη συνάρτηση KMeansClustering και του συσσωρευτικού ιεραρχικού αλγορίθμου στη HierarchicalClustering.

6.3 Υπολογισμός στατιστικών ιδιοτήτων για κάθε ομάδα

Μετά το στάδιο της ομαδοποίησης σε κάθε νευρώνα αντιστοιχεί ένας αριθμός που υποδεικνύει σε ποια ομάδα ανήκει. Από τους νευρώνες που ανήκουν στην ίδια ομάδα βρίσκουμε την μέση τιμή της ομάδας, την τυπική απόκλιση, την μέγιστη και την ελάχιστη τιμή. Οι τιμές αυτές χαρακτηρίζουν ολόκληρη την ομάδα και όχι κάθε νευρώνα ξεχωριστά και στη συνέχεια του αλγορίθμου θεωρούμε τον χάρτη σε επίπεδο ομάδων και όχι σε επίπεδο νευρώνων. Οι τιμές υπολογίζονται τόσο σε κανονικοποιημένα όσο και αποκανονικοποιημένα μεγέθη. Οι υπολογισμοί γίνονται στη StatisticsComp.

6.4 Επιλογή σημαντικών μεταβλητών και δημιουργία κανόνων

Σε κάθε ομάδα οι τιμές της κάθε μεταβλητής καταλαμβάνουν διαφορετικό εύρος τιμών από αυτό που καταλαμβάνουν σε όλον τον χάρτη. Υπάρχουν μεταβλητές που έχουν συγκεκριμένο εύρος τιμών σε κάποια ομάδα και ουσιαστικά χαρακτηρίζουν την ομάδα αυτή και την διαχωρίζουν από τις υπόλοιπες. Βρίσκοντας τις σημαντικότερες μεταβλητές που χαρακτηρίζουν μια ομάδα και αγνοώντας τις υπόλοιπες μπορούμε να απλοποιήσουμε την περαιτέρω ανάλυση των δεδομένων, να μειώσουμε τον όγκο των υπολογισμών και να φτάσουμε σε πιο απλά και πιο εύκολα κατανοητά αποτελέσματα. Επίσης για αυτές τις μεταβλητές μπορούμε να κατασκευάσουμε κανόνες που να ισχύουν για τα πρότυπα που κατηγοριοποιούνται σε κάθε ομάδα. Η επιλογή των σημαντικών μεταβλητών μπορεί να γίνει με δυο τρόπους.

Ο πρώτος τρόπος είναι να θεωρηθεί κάθε μεταβλητή ξεχωριστά, να αξιολογηθεί με βάση κάποιο κριτήριο και οι πιο σημαντικές να επιλεγούν. Στον δεύτερο τρόπο αντί για μεμονωμένες μεταβλητές θεωρούνται σύνολα μεταβλητών, τα οποία αξιολογούνται και επιλέγονται τα σημαντικότερα. Ο δεύτερος τρόπος έχει σημαντικά μεγαλύτερο υπολογιστικό κόστος από τον πρώτο, ιδιαίτερα σε σύνολα δεδομένων με πολλές μεταβλητές, αφού χρειάζεται να βρεθούν όλοι οι πιθανοί συνδυασμοί των μεταβλητών μεταξύ τους και για κάθε συνδυασμό να υπολογιστούν τα διάφορα μεγέθη που αποτελούν το κριτήριο επιλογής. Το πλεονέκτημα όμως αυτής της μεθόδου είναι ότι μπορεί να ανακαλύψει τις αλληλεπιδράσεις των μεταβλητών που χαρακτηρίζουν μια ομάδα, κάτι που δεν μπορεί να βρεθεί όταν θεωρούμε κάθε μεταβλητή ξεχωριστά.

6.4.1 Ο αλγόριθμος SIG*

Ο αλγόριθμος sig* αναπτύχθηκε από τον A. Ultsch [13]. Ο αλγόριθμος παίρνει ένα SOM στο οποίο έχουν δημιουργηθεί ομάδες για είσοδο και κατασκευάζει κανόνες που χαρακτηρίζουν τις ομάδες δεδομένων και τις διαχωρίζουν από τις υπόλοιπες. Για την επιλογή των μεταβλητών που χαρακτηρίζουν μια ομάδα κάθε μεταβλητή παίρνει μια "τιμή σπουδαιότητας" η οποία μπορεί να προέλθει από τις στατιστικές ιδιότητες της ομάδας. Για να εξηγηθεί ο αλγόριθμος θεωρούμε ότι έχουμε ένα σύνολο δεδομένων 4 μεταβλητών και στο SOM δημιουργούνται 7 ομάδες. Μπορούμε να κατασκευάσουμε τον "πίνακα σπουδαιότητας".

	Ομάδα 1	Ομάδα 2	Ομάδα 3	Ομάδα 4	Ομάδα 5	Ομάδα 6	Ομάδα 7
Μεταβλητή 1	4.535	2.673	4.2241	2.5619	3.3118	5.4126 *	2.8784
Μεταβλητή 2	2.0027	2.0065	3.4938 *	2.1526	2.4081	2.0771	3.0535
Μεταβλητή 3	10.004	12.986 *	5.1911	5.4576	3.479	7.1688	4.873
Μεταβλητή 4	10.21	13.626 *	3.0457	4.7029	4.4746	6.3346	3.9966

Πίνακας 1. Πίνακας σπουδαιότητας για τον αλγόριθμο sig*

Στον πίνακα σπουδαιότητας μαρκάρουμε την μεγαλύτερη τιμή που παίρνει κάθε μεταβλητή με ένα αστεράκι. Για την επιλογή των πιο σημαντικών μεταβλητών για την περιγραφή κάθε κλάσης σχηματίζουμε έναν νέο πίνακα όπου οι τιμές σπουδαιότητας κάθε μεταβλητής κανονικοποιούνται ως προς το άθροισμα των τιμών σπουδαιότητας για όλη την ομάδα. Οι τιμές αυτές ταξινομούνται από την μεγαλύτερη προς τη μικρότερη και υπολογίζονται οι αθροιστικές τιμές σπουδαιότητας. Για παράδειγμα για τις ομάδες 1 και 6 οι νέοι πίνακες είναι:

	Ομάδα 1	Αθροιστικές τιμές
Μεταβλητή 4	38.1665%	38.1665%
Μεταβλητή 3	37.3952%	75.5617%
Μεταβλητή 1	16.9521%	92.5138%
Μεταβλητή 2	7.4862%	100.00%

	Ομάδα 6	Αθροιστικές τιμές
Μεταβλητή 3	34.1484%	34.1484%
Μεταβλητή 4	30.1748%	64.3231%
Μεταβλητή 1 *	25.7828%	90.1060%
Μεταβλητή 2	9.8940%	100.00%

Πίνακας 2. Παράδειγμα επιλογής μεταβλητών με τον αλγόριθμο sig*

Από τους παραπάνω πίνακες διαλέγουμε τις μεταβλητές με το μεγαλύτερο ποσοστό σπουδαιότητας μέχρι οι αθροιστικές τιμές σπουδαιότητας να είναι ίσες ή να ξεπεράσουν ένα προκαθορισμένο κατώφλι. Μια συνηθισμένη επιλογή για το κατώφλι αυτό είναι το 50%. Με αυτό το κατώφλι στην ομάδα 1 επιλέγονται οι μεταβλητές 4 και 3 ενώ στην ομάδα 6 οι μεταβλητές 3 και 4. Επειδή στην ομάδα 6 η μεταβλητή 1 παίρνει τη μεγαλύτερη τιμή σημαντικότητας και είναι σημαδεμένη με αστεράκι πρέπει να θεωρηθεί ότι είναι χαρακτηριστική για την ομάδα ανεξάρτητα από το γεγονός ότι η αθροιστική τιμή σπουδαιότητας υπερβαίνει το κατώφλι. Γενικά, όλες οι μεταβλητές που έχουν αστεράκι θεωρούνται σημαντικές για την αντίστοιχη ομάδα και συμπεριλαμβάνονται στην περιγραφή.

Με αυτόν τον τρόπο αν μια ή κάποιες λίγες μεταβλητές είναι πολύ σημαντικές για μία ομάδα επιλέγονται μόνο αυτές. Αντίθετα, αν όλες οι μεταβλητές είναι περίπου το ίδιο

σημαντικές θα επιλεγούν πολλές από αυτές. Έτσι η περιγραφή των ομάδων άρα και οι κανόνες που θα προκύψουν παραμένουν αρκετοί απλοί. Αν σε κάποιες ομάδες επιλέγονται οι ίδιες μεταβλητές για την περιγραφή τους και επίσης το εύρος τιμών των μεταβλητών είναι έστω και εν μέρει το ίδιο σε όλες τις ομάδες, τότε ουσιαστικά οι ομάδες αυτές δεν μπορούν να διαχωριστούν μεταξύ τους. Για να γίνει αυτό μπορούν να προστεθούν και άλλες μεταβλητές στην περιγραφή κάθε ομάδας μεγαλώνοντας το κατώφλι του αλγορίθμου για τις ομάδες αυτές μέχρι να επιτευχθεί ικανοποιητικός διαχωρισμός.

Ο αλγόριθμος sig* δεν καθορίζει ποια ποσότητα πρέπει να χρησιμοποιείται ως τιμή σπουδαιότητας. Μια επιλογή για την τιμή σπουδαιότητας είναι η μέση τιμή της μεταβλητής σε κάθε ομάδα. Η μεταβλητή με την μεγαλύτερη μέση τιμή θεωρείται ότι είναι πιο σημαντική. Η επιλογή αυτή δεν οδηγεί όμως σε σωστά αποτελέσματα αφού μια μεγάλη μέση τιμή δε σημαίνει απαραίτητα ότι η μεταβλητή είναι σημαντική ή μπορεί να διαχωρίσει την ομάδα από τις υπόλοιπες.

Άλλη ποσότητα που μπορεί να χρησιμοποιηθεί είναι η τυπική απόκλιση των δεδομένων. Αν οι τιμές μιας μεταβλητής σε μια ομάδα κινούνται μέσα σε ένα στενό εύρος τιμών ή ισοδύναμα η μεταβλητή έχει μικρή τυπική απόκλιση γύρω από μια κεντρική τιμή, η τιμή αυτή είναι χαρακτηριστική και μπορεί να περιγράψει την ομάδα. Ως τιμή σημαντικότητας μπορεί να θεωρηθεί το αντίστροφο της τυπικής απόκλισης κάθε μεταβλητής μέσα στην ομάδα.

Από την εφαρμογή αυτής της μεθόδου βρέθηκε ότι δεν αξιολογούνται σωστά μεταβλητές οι οποίες παίρνουν σχεδόν τις ίδιες τιμές στο μεγαλύτερο μέρος του χάρτη και διαφοροποιούνται έντονα μόνο σε ένα μικρό μέρος τους. Λόγω της μικρής τυπικής απόκλισης που έχουν στο μεγαλύτερο μέρος του χάρτη θεωρούνται σημαντικές στις περισσότερες ομάδες ενώ εκεί που η μεταβλητή παίρνει διαφορετικές τιμές δεν θεωρούνται σημαντικές. Έτσι ουσιαστικά παίρνουμε το αντίθετο αποτέλεσμα από αυτό που θα έπρεπε αφού στο μικρό μέρος του χάρτη που η μεταβλητή όντως διαχωρίζει τις ομάδες δε θεωρείται σημαντική ενώ εκεί που η μεταβλητή παίρνει τις ίδιες τιμές για πολλές ομάδες θεωρείται σημαντική.

Το μειονέκτημα αυτών των επιλογών είναι ότι δεν συνυπολογίζονται οι τιμές της μεταβλητής σε γειτονικές ή και σε όλες τις άλλες ομάδες ώστε να φαίνεται αν οι μεταβλητές που θεωρούνται σημαντικές μπορούν πραγματικά να ξεχωρίσουν τις ομάδες. Μια παραλλαγή που λαμβάνει υπόψη τις τιμές της μεταβλητής σε όλες τις ομάδες είναι να χρησιμοποιηθεί ως τιμή σπουδαιότητας ο λόγος της τυπικής απόκλισης της μεταβλητής σε όλο το χάρτη προς την τυπική απόκλιση της μεταβλητής στην ομάδα.

Ανεξάρτητα από την ποσότητα που θα αποτελέσει το κριτήριο επιλογής αξιολογώντας κάθε μεταβλητή ξεχωριστά δεν υπάρχει καμία ένδειξη για την αλληλεπίδραση των διαφόρων μεταβλητών μέσα στην ομάδα. Υπάρχουν περιπτώσεις που ο συνδυασμός μεταβλητών είναι αυτός που χαρακτηρίζει ορθότερα μια ομάδα. Στην επόμενη παράγραφο θα μελετηθεί μια τεχνική για επιλογή ομάδων μεταβλητών. Η υλοποίηση του αλγορίθμου sig* γίνεται στη SigStar.

6.4.2 Επιλογή ομάδων μεταβλητών

Αν ένα σύνολο δεδομένων αποτελείται από n μεταβλητές και πρέπει να επιλεγούν οι k πιο σημαντικές από αυτές, η βέλτιστη μέθοδος για να γίνει αυτό είναι να βρεθούν όλοι οι πιθανοί συνδυασμοί των k μεταβλητών, να αξιολογηθούν με βάση κάποιο κριτήριο και να επιλεγεί ο καλύτερος. Για να γίνει αυτό θα πρέπει να διερευνηθούν συνολικά

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

συνδυασμοί, αριθμός που να μπορεί να γίνει εξαιρετικά μεγάλος ακόμα και για μικρές τιμές των n και k. Για παράδειγμα για n = 24 και k = 11 υπάρχουν 2496144 συνδυασμοί. Οι υπολογισμοί αυτοί είναι πολλές φορές πρακτικά αδύνατο να γίνουν μέσα σε κάποιο αποδεκτό χρονικό διάστημα.

Για να αποφευχθεί ένας τόσο μεγάλος όγκος υπολογισμών ακολουθήθηκε μια παραλλαγή της βέλτιστης μεθόδου. Ανεξάρτητα από το συνολικό πλήθος των μεταβλητών και το πλήθος του υποσυνόλου των μεταβλητών που θεωρείται ότι περιγράφουν κάθε ομάδα βρίσκουμε τους συνδυασμούς για ένα μικρό αριθμό μεταβλητών. Στη συνέχεια δεν επιλέγουμε μόνο τον συνδυασμό μεταβλητών που μεγιστοποιεί το κριτήριο αλλά όσους συνδυασμούς θεωρείται ότι χρειάζονται ώστε συνολικά να επιλεγούν αρκετές μεταβλητές που να χαρακτηρίζουν τις ομάδες. Πόσες μεταβλητές θα υπάρχουν σε κάθε συνδυασμό και πόσοι συνδυασμοί θα επιλέγονται κάθε φορά είναι μια επιλογή που για κάθε διαφορετικό σύνολο δεδομένων μπορεί να αλλάζει ανάλογα και με το μέγεθος του. Έτσι υπολογίζονται οι σημαντικές μεταβλητές κάθε ομάδας σε πολύ μικρότερο χρόνο και με πολύ λιγότερες πράξεις.

Για το κριτήριο επιλογής υπολογίζονται η συνδιακύμανση των μεταβλητών μέσα στην ομάδα και ανάμεσα σε όλες τις ομάδες. Το κριτήριο μεγιστοποιείται όταν για κάποια ομάδα ο συνδυασμός των μεταβλητών που εξετάζεται έχει μικρή συνδιακύμανση μέσα στην ομάδα αυτή και μεγάλη συνδιακύμανση σχετικά με τις υπόλοιπες ομάδες. Πιο συγκεκριμένα οι ποσότητες που υπολογίζονται είναι [14]:

α) Πίνακας συνδιακύμανσης (covariance matrix) εντός της ομάδας i

$$\mathbf{\Sigma}_i = \frac{1}{n_i}\sum_{j=1}^{n} z_{ij}\left(\mathbf{x}_j - \mathbf{m}_i\right)\left(\mathbf{x}_j - \mathbf{m}_i\right)^T$$

όπου n_i είναι το πλήθος των στοιχείων της ομάδας Q_i, το z_{ij} υποδεικνύει τα στοιχεία της ομάδας Q_i

$$z_{ij} = \begin{cases} 1\ \alpha\nu\ \mathbf{x}_j \in Q_j \\ 0\ \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

και $\mathbf{m}_i$ είναι η μέση τιμή των στοιχείων της ομάδας Q_i.

β) Πίνακας συνδιακύμανσης μεταξύ των ομάδων

$$\mathbf{S}_B = \frac{n_i}{n}(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

όπου **m** είναι η μέση τιμή των στοιχείων όλων των ομάδων.

Το κριτήριο επιλογής βασίζεται στους δύο αυτούς πίνακες και επιλέγεται ως πιο σημαντικός ο συνδυασμός μεταβλητών που μεγιστοποιεί την ποσότητα

$$J = Tr\{\mathbf{\Sigma}_i^{-1}\mathbf{S}_B\}$$

όπου Tr είναι το ίχνος του γινομένου των δυο πινάκων.

Η υλοποίησης αυτής της μεθόδου γίνεται στη VariableSelection.

6.4.3 Κατασκευή κανόνων

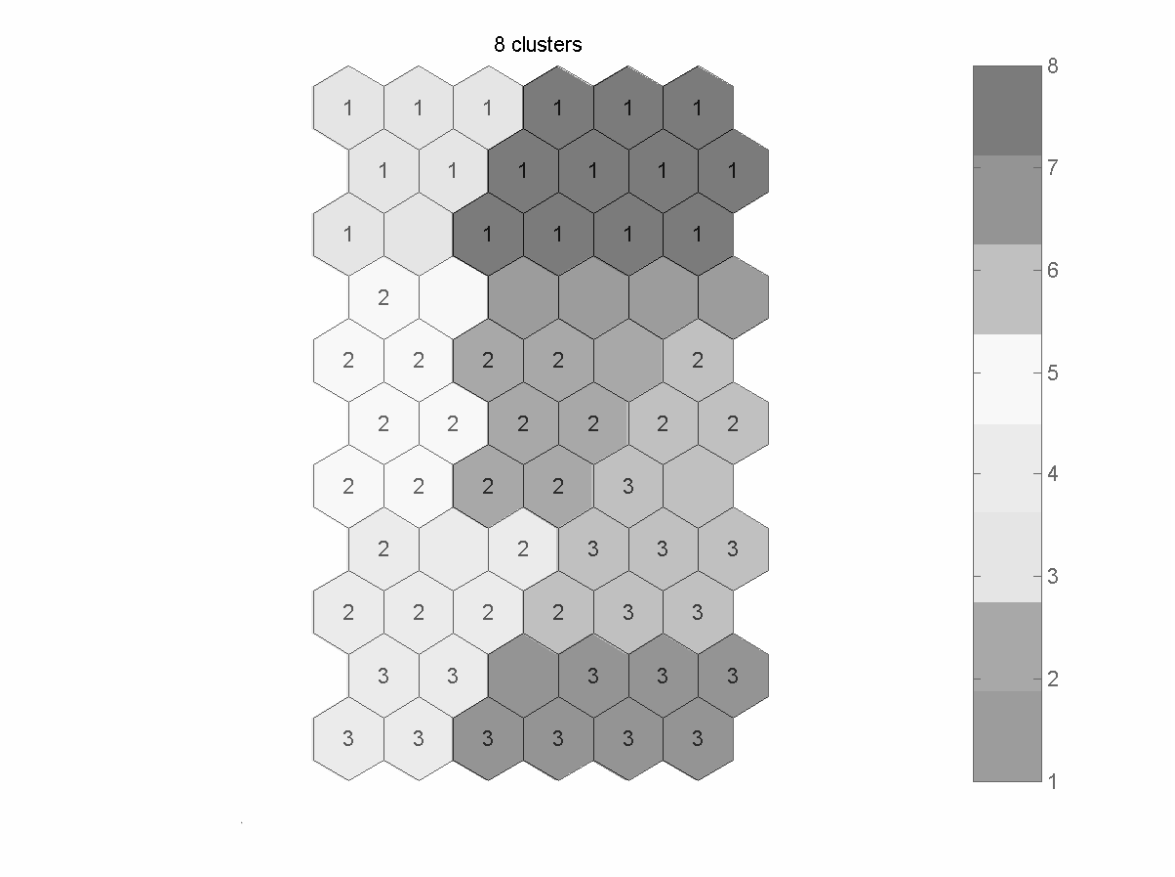
Από τη στιγμή που έχουν επιλεγεί οι μεταβλητές που περιγράφουν ικανοποιητικά κάθε ομάδα μπορούν να χρησιμοποιηθούν για την κατασκευή κανόνων. Αν οι συνθήκες των κανόνων είναι πολύ αυστηρές τότε πολλά πρότυπα που ανήκουν στην ομάδα μπορεί να ικανοποιούν τους κανόνες αυτούς. Αντίθετα, αν οι συνθήκες των κανόνων είναι πολύ χαλαρές τότε θα τις ικανοποιούν πρότυπα που κανονικά δεν ανήκουν σε αυτή την ομάδα οδηγώντας και πάλι σε λανθασμένη κατηγοριοποίηση. Για να κατασκευαστούν σωστοί κανόνες πρέπει να γίνει μια καλή εκτίμηση για την κατανομή των δεδομένων μέσα στην ομάδα.

Οι πιο απλοί κανόνες κατασκευάζονται αν υπολογίσουμε την ελάχιστη και την μέγιστη τιμή κάθε μεταβλητής και θεωρήσουμε αυτές τις τιμές ως κάτω και άνω όριο του κανόνα αντίστοιχα. Έτσι καταλήγουμε για κάθε σημαντική μεταβλητή j σε κανόνες της μορφής μεταβλητή$_j$ ανήκει [min$_j$, max$_j$]. Ουσιαστικά με τέτοιου τύπου κανόνες δε γίνεται κάποια υπόθεση για την κατανομή των δεδομένων και μπορεί να κατασκευαστούν κανόνες μικρής ακρίβειας αφού αρκεί ένα μόνο πρότυπο που να έχει τιμή αρκετά μικρότερη ή αρκετά μεγαλύτερη από όλα τα υπόλοιπα της ομάδας για να αλλάξει τελείως η συνθήκη του κανόνα με αποτέλεσμα χειρότερη κατηγοριοποίηση.

Μια υπόθεση που γίνεται συχνά και εφαρμόστηκε σε αυτήν την εργασία είναι ότι τα δεδομένα ακολουθούν την κανονική κατανομή. Μια σημαντική ιδιότητα της κανονικής κατανομής που είναι γνωστή από την στατιστική είναι ότι το 95% των δεδομένων περιέχονται στο διάστημα μ±2σ όπου μ είναι η μέση τιμή των δεδομένων και σ η τυπική απόκλιση των δεδομένων. Έτσι για κάθε σημαντική μεταβλητή j μπορούμε να κατασκευάσουμε έναν κανόνα της μορφής μεταβλητή$_j$ ανήκει [μ_j - 2σ_j, μ_j + 2σ_j] που να περιγράφει ικανοποιητικά την ομάδα. Η μέση τιμή και η τυπική απόκλιση κάθε ομάδας υπολογίζονται από τα αποκανονικοποιημένα βάρη των νευρώνων έτσι ώστε οι κανόνες να ισχύουν απευθείας για το αρχικό σύνολο δεδομένων και να έχουν μεγαλύτερη φυσική σημασία.

6.5 Αντιστοίχηση ομάδων με τις κατηγορίες των αρχικών δεδομένων

Οι κανόνες που έχουν δημιουργηθεί στο προηγούμενο βήμα του αλγορίθμου ισχύουν για το αρχικό σύνολο δεδομένων αλλά όχι και για τις κατηγορίες των δεδομένων αφού περιγράφουν τις ομάδες που έχουν σχηματισθεί το SOM. Οι κατηγορίες των δεδομένων που αντιστοιχούν σε κάθε νευρώνα έχουν βρεθεί στο πρώτο βήμα του αλγορίθμου άρα μπορεί να βρεθεί μια αντιστοιχία μεταξύ ομάδων και κατηγοριών δεδομένων (σχήμα 16).



Σχήμα 16. Αντιστοίχηση ομάδων με τις αρχικές κατηγορίες δεδομένων

Αρχικά γίνεται καταμέτρηση σε όλες τις ομάδες του χάρτη του πλήθους των νευρώνων που περιγράφουν κάθε κατηγορία δεδομένων. Κατά κανόνα οι ομάδες νευρώνων που σχηματίζονται στο SOM αποτελούνται από λίγους νευρώνες ενώ οι περιοχές που καταλαμβάνουν οι κατηγορίες των δεδομένων στο χάρτη είναι μεγαλύτερες. Αν ένας αρκετά μεγάλος αριθμός νευρώνων μέσα σε μια ομάδα περιγράφει την ίδια κατηγορία τότε θεωρούμε ότι υπάρχει αντιστοιχία μεταξύ ομάδας και κατηγορίας. Ο κανόνας που ίσχυε για την ομάδα ισχύει πλέον και για την κατηγορία των αρχικών δεδομένων. Για την εφαρμογή θεωρήθηκε ότι αν το 40% τουλάχιστον των νευρώνων μιας ομάδας περιγράφει μια κατηγορία δεδομένων τότε υπάρχει αντιστοιχία. Με αυτόν τον τρόπο επιλέγονται για κάθε ομάδα μία ή το πολύ δύο κυρίαρχες κατηγορίες ενώ σε ομάδες που κατατάσσονται πρότυπα από πολλές διαφορετικές κατηγορίες δε γίνεται καμία αντιστοίχηση αφού αυτό θα οδηγούσε σε κανόνες χαμηλής ακρίβειας. Η υλοποίηση γίνεται στις συναρτήσεις RuleSections και IrisRuleCreation, IonRuleCreation SegRuleCreation αφού για κάθε σύνολο δεδομένων υπάρχει διαφορετικός αριθμός κατηγοριών και ετικετών.

6.6 Επεξεργασία κανόνων

Οι κανόνες είναι πλέον έτοιμοι να εφαρμοστούν στο σύνολο δεδομένων. Αρχικά εφαρμόζεται κάθε κανόνας σε κάθε πρότυπο ξεχωριστά. Έτσι καταμετρούνται τα πρότυπα που κατηγοριοποιούνται σωστά από τον κανόνα και πόσα πρότυπα ενώ επαληθεύουν τον κανόνα δεν ανήκουν στην κατηγορία που υποδεικνύει ο κανόνας. Επίσης υπολογίζεται και

η σπουδαιότητα του κανόνα σύμφωνα με το μέτρο S_2 που αναφέρθηκε στο κεφάλαιο 4 και οι κανόνες για κάθε κατηγορία δεδομένων ταξινομούνται από τον περισσότερο προς το λιγότερο σημαντικό.

Στη συνέχεια γίνεται συνδυασμός των πιο επιτυχημένων κανόνων για πληρέστερη περιγραφή των δεδομένων. Αρχίζοντας από τον πιο επιτυχημένο κανόνα προσθέτουμε κανόνες που αυξάνουν την αποτελεσματικότητα του κανόνα στη συγκεκριμένη κατηγορία ή γενικά σε ολόκληρο το σύνολο δεδομένων μέχρι να υπάρξει ικανοποιητική κατηγοριοποίηση. Ο συνδυασμός των κανόνων μπορεί να γίνει με δύο τρόπους, με σύζευξη (AND) ή διάζευξη (OR). Αν ο κανόνας κατηγοριοποιεί σωστά την πλειοψηφία των προτύπων της κατηγορίας χρησιμοποιείται σύζευξη ανεξάρτητα από πόσα πρότυπα ακόμα ικανοποιούν τον κανόνα ενώ δε θα έπρεπε. Αν ο κανόνας επαληθεύεται μόνο από πρότυπα τις σωστής κατηγορίας και λίγα πρότυπα από άλλες κατηγορίες χρησιμοποιείται διάζευξη.

Η εφαρμογή κάποιων κανόνων, ανεξάρτητα από την ατομική σπουδαιότητα τους, μπορεί να μην βελτιώνει καθόλου ή ακόμα και να μειώνει την απόδοση του συνόλου των κανόνων. Αυτό συμβαίνει γιατί τα πρότυπα που κατηγοριοποιεί μπορεί να έχουν ήδη κατηγοριοποιηθεί από πιο επιτυχημένους κανόνες. Επίσης, σε περιπτώσεις που οι κανόνες επικαλύπτονται, δηλαδή υπάρχουν πρότυπα που ικανοποιούν δύο ή και περισσότερους κανόνες, υπάρχει περίπτωση ο συνδυασμός ενός νέου κανόνα να μειώνει την απόδοση σε μια κατηγορία αλλά να αυξάνεται η απόδοση συνολικά από όλους τους κανόνες οπότε ο κανόνας αυτός εντάσσεται στο σύνολο κανόνων. Κάποιοι κανόνες δε χρειάζεται να μπουν στο σύνολο κανόνων είτε γιατί έχει ήδη επιτευχθεί πολύ καλή περιγραφή είτε επειδή οι κανόνες δεν περιγράφουν ικανοποιητικά την κατηγορία λόγω των μη βέλτιστων διαδικασιών στην εξαγωγή τους. Η υλοποίηση της επεξεργασίας των κανόνων γίνεται στη μέθοδο RuleOutput.

7

Πειραματικά αποτελέσματα

7.1 Σύνολα δεδομένων

Η εφαρμογή και η αξιολόγηση των μεθόδων εξαγωγής κανόνων έγινε σε τρία διαφορετικά σύνολα δεδομένων που παρουσιάζουν διαφορετικά χαρακτηριστικά ώστε να εκτιμηθούν οι δυνατότητες και οι αδυναμίες των μεθόδων. Όλα τα σύνολα δεδομένων αποτελούνται από συνεχείς αριθμητικές τιμές. Τα σύνολα δεδομένων μπορούν να βρεθούν στη βάση δεδομένων μηχανικής μάθησης UCI [15].

α) Ionosphere

Το σύνολο δεδομένων Ionosphere αποτελείται από δεδομένα που προέρχονται από ένα σύστημα ραντάρ και περιγράφουν σήματα που είτε βρήκαν κάποιο σωματίδιο στην ιονόσφαιρα και χαρακτηρίζονται "καλά" ή δεν βρήκαν τίποτα και χαρακτηρίζονται "άσχημα", δηλαδή υπάρχουν 2 κατηγορίες. Τα δεδομένα αποτελούνται από 34 μεταβλητές που προέρχονται από την επεξεργασία των σημάτων. Υπάρχουν 351 δείγματα από τα οποία τα 225 είναι "καλά" και τα υπόλοιπα 126 "άσχημα".

β) Iris

Το σύνολο δεδομένων Iris αποτελείται από δεδομένα που περιγράφουν 3 διαφορετικούς τύπους λουλουδιών του είδους Iris. Οι τρεις κατηγορίες είναι Iris-Setosa, Iris-Versicolor και Iris-Virginica. Τα δεδομένα αποτελούνται από 4 μεταβλητές που είναι το μήκος και το πλάτος των σέπαλων και των πετάλων των λουλουδιών σε εκατοστά. Υπάρχουν συνολικά 150 δείγματα από τα οποία σε κάθε κατηγορία ανήκουν 50 δείγματα. Η κατηγορία Iris-Setosa είναι γραμμικά διαχωρίσιμη από τις δύο άλλες κατηγορίες οι οποίες δεν είναι γραμμικά διαχωρίσιμες.

γ) Image Segmentation

Το σύνολο δεδομένων Image Segmentation αποτελείται από δεδομένα που προέρχονται από 7 εικόνες εξωτερικών χώρων. Οι εικόνες κατατμήθηκαν έτσι ώστε να δημιουργηθεί κατηγοριοποίηση για κάθε pixel. Υπάρχουν 7 κατηγορίες που είναι τούβλα, ουρανός, φύλλωμα, τσιμέντο, παράθυρο, μονοπάτι και γρασίδι. Κάθε δείγμα είναι μια περιοχή 3x3 και αποτελείται από 19 μεταβλητές που προέρχονται από την επεξεργασία των εικόνων. Υπάρχουν συνολικά 2310 δείγματα από τα οποία σε κάθε κατηγορία ανήκουν 330 δείγματα.

7.2 Αποτελέσματα

Η εφαρμογή των αλγορίθμων στα δεδομένα έγινε πολλές φορές με διαφορετικές παραμέτρους για να μελετηθεί ποια μέθοδος βγάζει καλύτερα αποτελέσματα και υπό ποιες προϋποθέσεις. Τα αποτελέσματα που παραθέτουμε αφορούν ολόκληρο το σύνολο δεδομένων και εκφράζονται σε ποσοστά επί του συνολικού πλήθους προτύπων. Η μετρήσεις είναι για πρότυπα που:

1. Κατηγοριοποιούνται στη σωστή κατηγορία και μόνο αυτή
   (Σωστή Κατηγοριοποίηση - ΣΚ)
2. Κατηγοριοποιούνται σε λάθος κατηγορία
   (Λάθος Κατηγοριοποίηση – ΛΚ)
3. Δεν κατηγοριοποιούνται καθόλου (Άγνωστα Πρότυπα – ΑΠ)
4. Κατηγοριοποιούνται σε παραπάνω από μια κατηγορίες
   (Πολλαπλή Κατηγοριοποίηση – ΠΚ)

Όταν υπάρχουν πολλά πρότυπα που κατηγοριοποιούνται σε παραπάνω από μια κατηγορίες εντάσσονται στην κατηγορία που έχει την μικρότερη πιθανότητα λάθους όπως εξηγήθηκε στο κεφάλαιο 4. Η τεχνική αυτή εφαρμόστηκε στο σύνολο δεδομένων Image Segmentation όπου βρέθηκε ότι υπάρχουν πολλά τέτοια πρότυπα.

7.3 Αποτελέσματα μεθόδου εξαγωγής κανόνων με βάση τον U-matrix

Η μέθοδος αυτή εφαρμόστηκε στα τρία σύνολα δεδομένων με διαφορετικούς συνδυασμούς κάποιων παραμέτρων ώστε να αξιολογηθεί η επίδραση τους στα αποτελέσματα. Ονομάζουμε "μέθοδο 1α" τη μέθοδο που βασίζεται στον U-matrix και ο υπολογισμός του πίνακα BDV γίνεται με χρήση του τύπου $BDV = (M_L - M_O) / R_O$ ενώ "μέθοδος 1β" είναι αυτή που γίνεται χρήση του τύπου $BDV = M_L - M_O$. Οι παράμετροι που διαφοροποιούνται είναι:

- Τρόπος υπολογισμού των τιμών BDV
- Μέγεθος χάρτη που μπορεί να κανονιστεί σε κανονικό ή μεγάλο και υπολογίζεται αυτόματα από την εφαρμογή
- Όριο εύρους τιμών BDV που μπορεί να είναι μικρό ώστε μεγάλες αποκλίσεις να θεωρούνται λάθη και να μη λαμβάνονται υπόψη στη δημιουργία ορίων ή μεγάλο ώστε να επιτρέπονται μεγάλες αποκλίσεις

- Ελάχιστο ποσοστό ταυτοποίησης ορίων που όσο μεγαλύτερο είναι τόσο περισσότερο πρέπει να ταυτίζονται τα όρια του SOM και του πίνακα της μεταβλητής

7.3.1 Σύνολο δεδομένων Ionosphere

Το σύνολο δεδομένων Ionosphere αποτελείται από δύο μόνο κλάσεις από τις οποίες η μία έχει πολύ περισσότερα πρότυπα. Βρέθηκε ότι οι κανόνες είναι πολύ πιο ακριβείς για την πρώτη και μεγαλύτερη κατηγορία δεδομένων. Για το λόγο αυτό δε δημιουργείται κανόνας για τη δεύτερη κατηγορία αλλά θεωρείται ότι τα πρότυπα που δεν ικανοποιούν τον κανόνα της πρώτης κατηγορίας ανήκουν στη δεύτερη. Τα αποτελέσματα που προκύπτουν είναι:

Μέθοδος	Μέγεθος χάρτη	Όριο εύρους τιμών BDV	Ελάχιστη ταυτοποίηση ορίων (%)	ΣΚ (%)	ΛΚ (%)
Μέθοδος 1α	Κανονικό	Μικρό	50	92.02	7.98
Μέθοδος 1α	Κανονικό	Μεγάλο	50	92.59	7.41
Μέθοδος 1β	Κανονικό	Μικρό	50	90.88	9.12
Μέθοδος 1β	Κανονικό	Μεγάλο	50	90.88	9.12
Μέθοδος 1α	Μεγάλο	Μικρό	50	93.73	6.27
Μέθοδος 1α	Μεγάλο	Μεγάλο	50	94.59	5.41
Μέθοδος 1β	Μεγάλο	Μικρό	50	94.30	5.70
Μέθοδος 1β	Μεγάλο	Μεγάλο	50	94.30	5.70

Πίνακας 3. Αποτελέσματα μεθόδου 1 στα δεδομένα Ionosphere με ελάχιστη ταυτοποίηση ορίων 50%

Από τον παραπάνω πίνακα παρατηρούμε ότι για μεγάλο μέγεθος χάρτη έχουμε αρκετά καλύτερα αποτελέσματα. Για το λόγο αυτό η εφαρμογή με άλλα ποσοστά ταυτοποίησης ορίων έγινε μόνο σε μεγάλο χάρτη.

Μέθοδος	Μέγεθος χάρτη	Όριο εύρους τιμών BDV	Ελάχιστη ταυτοποίηση ορίων (%)	ΣΚ (%)	ΛΚ (%)
Μέθοδος 1α	Μεγάλο	Μικρό	25	94.59	5.41
Μέθοδος 1α	Μεγάλο	Μεγάλο	25	**94.87**	5.13
Μέθοδος 1β	Μεγάλο	Μικρό	25	**94.87**	5.13
Μέθοδος 1β	Μεγάλο	Μεγάλο	25	**94.87**	5.13
Μέθοδος 1α	Μεγάλο	Μικρό	75	93.45	6.55
Μέθοδος 1α	Μεγάλο	Μεγάλο	75	94.59	5.41
Μέθοδος 1β	Μεγάλο	Μικρό	75	94.02	5.98
Μέθοδος 1β	Μεγάλο	Μεγάλο	75	94.02	5.98

Πίνακας 4. Αποτελέσματα μεθόδου 1 στα δεδομένα Ionosphere με ελάχιστο ποσοστό ταυτοποίησης ορίων 25% και 75%

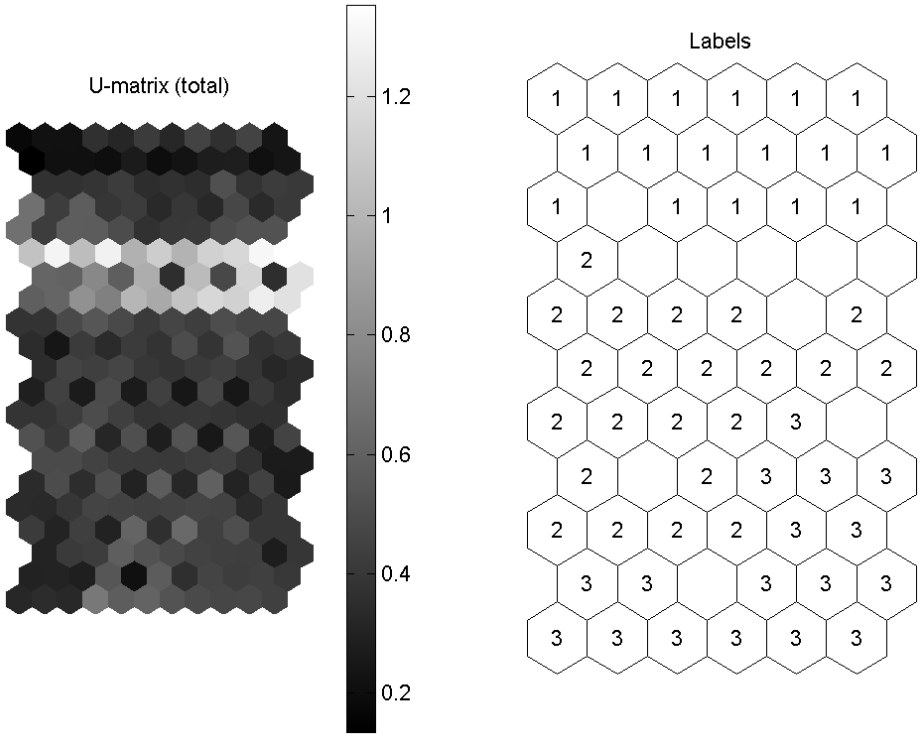
Παρατηρούμε ότι καθώς αυξάνεται το ελάχιστο ποσοστό ταυτοποίησης ορίων υπάρχει μια μικρή μόνο μείωση της ακρίβειας των κανόνων που σημαίνει ότι τα όρια που σχηματίζονται στο χάρτη και ταυτοποιούνται από τον αλγόριθμο είναι αρκετά ευδιάκριτα. Επίσης, το όριο εύρος τιμών BDV δε διαφοροποιεί σημαντικά τα αποτελέσματα αφού με

μεγάλο όριο η αύξηση στην ακρίβεια είναι μικρή. Το καλύτερο σύνολο κανόνων δημιουργείται με ελάχιστο ποσοστό ταυτοποίησης ορίων 25% και είναι το:

IF Var3>0.09771574
AND Var1>0.5799758
AND Var5>0.05921073
AND Var4>-0.7077920
AND Var16>-0.6837564
AND Var6>-0.3947427
AND Var8>-0.5933991 THEN class is 1 (“good”)
ELSE class is 2 (“bad”)

7.3.2 Σύνολο δεδομένων Iris

Για το σύνολο δεδομένων Iris παρατηρούμε ότι στον U-matrix υπάρχει ένα ευδιάκριτο όριο που διαχωρίζει τα δεδομένα της κατηγορίας 1 από τα υπόλοιπα δεδομένα (σχήμα 17). Αναμένουμε λοιπόν η κατηγορία Iris-Setosa να περιγράφεται τέλεια από έναν κανόνα αφού είναι γραμμικά διαχωρίσιμη από τις υπόλοιπες. Στις άλλες δύο κατηγορίες θα υπάρχει μια μικρή επικάλυψη. Το σύνολο δεδομένων Iris είναι αρκετά απλό οπότε για την περιγραφή του αρκεί ένας κανονικού μεγέθους χάρτης. Τα αποτελέσματα από την εφαρμογή του αλγορίθμου είναι:



Σχήμα 17. U-matrix και ετικέτες στο SOM για τα δεδομένα Iris

Μέθοδος	Μέγεθος χάρτη	Όριο εύρους τιμών BDV	Ελάχιστη ταυτοποίηση ορίων (%)	ΣΚ (%)	ΛΚ (%)	ΑΠ (%)	ΠΚ (%)
Μέθοδος 1α	Κανονικό	Μικρό	25	**96.67**	3.33	0.00	0.00
Μέθοδος 1α	Κανονικό	Μεγάλο	25	96.00	2.00	0.67	1.33
Μέθοδος 1β	Κανονικό	Μικρό	25	95.33	2.00	1.33	1.33
Μέθοδος 1β	Κανονικό	Μεγάλο	25	95.33	2.00	1.33	1.33
Μέθοδος 1α	Κανονικό	Μικρό	50	**96.67**	3.33	0.00	0.00
Μέθοδος 1α	Κανονικό	Μεγάλο	50	**96.67**	2.67	0.67	0.00
Μέθοδος 1β	Κανονικό	Μικρό	50	95.33	3.33	0.00	1.33
Μέθοδος 1β	Κανονικό	Μεγάλο	50	95.33	3.33	0.00	1.33
Μέθοδος 1α	Κανονικό	Μικρό	75	**96.67**	3.33	0.00	0.00
Μέθοδος 1α	Κανονικό	Μεγάλο	75	**96.67**	2.67	0.67	0.00
Μέθοδος 1β	Κανονικό	Μικρό	75	95.33	2.00	1.33	1.33
Μέθοδος 1β	Κανονικό	Μεγάλο	75	95.33	2.00	1.33	1.33

Πίνακας 5. Αποτελέσματα μεθόδου 1 στα δεδομένα Iris

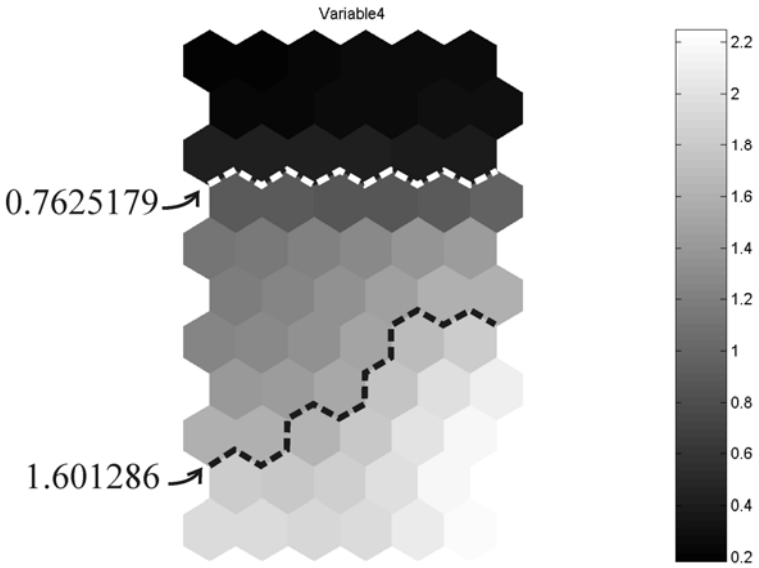
Από τον παραπάνω πίνακα βλέπουμε ότι για το συγκεκριμένο απλό σύνολο δεδομένων δεν έχει σημασία ποιες παράμετροι θα επιλεγούν κατά την εκτέλεση του προγράμματος. Με διάφορες παραμέτρους έχει επιτευχθεί το ίδιο καλή κατηγοριοποίηση, το σύνολο κανόνων που βρέθηκε με τη μέθοδο 1α και μεγάλο ποσοστό ταυτοποίησης ορίων της τάξης του 75% είναι:

IF Var4<0.7625179 THEN class is 1 (Setosa)

IF Var4>0.7625179
AND Var4<1.601286
AND Var1<7.044481 THEN class is 2 (Versicolor)

IF Var4>1.601286
OR Var1>7.044481 THEN class is 3 (Virginica)

Τα όρια στα χάρτη που δημιουργούνται από τους κανόνες της μεταβλητής 4 φαίνοντα στο σχήμα 18 και ταυτίζονται σε μεγάλο βαθμό με τις περιοχές που καλύπτουν οι ετικέτες στο χάρτη.



Σχήμα 18. Όρια που δημιουργούν οι κανόνες στο χάρτη

7.3.3 Σύνολο δεδομένων Image Segmentation

Το σύνολο δεδομένων Image Segmentation είναι πιο μεγάλο και πολύπλοκο από τα δύο προηγούμενα αφού έχει 7 διαφορετικές κατηγορίες. Τα αποτελέσματα για κανονικό και μεγάλο μέγεθος χάρτη φαίνονται στους πίνακες 6 και 7 αντίστοιχα.

Μέθοδος	Μέγεθος χάρτη	Όριο εύρους τιμών BDV	Ελάχιστη ταυτοποίηση ορίων (%)	ΣΚ (%)	ΛΚ (%)	ΑΠ (%)	ΠΚ (%)
Μέθοδος 1α	Κανονικό	Μικρό	25	85.02	2.55	11.56	0.87
Μέθοδος 1α	Κανονικό	Μεγάλο	25	87.45	1.95	8.61	1.99
Μέθοδος 1β	Κανονικό	Μικρό	25	84.20	1.26	14.03	0.52
Μέθοδος 1β	Κανονικό	Μεγάλο	25	85.97	1.90	10.91	1.21
Μέθοδος 1α	Κανονικό	Μικρό	50	84.37	2.55	12.51	0.56
Μέθοδος 1α	Κανονικό	Μεγάλο	50	83.59	1.26	14.94	0.22
Μέθοδος 1β	Κανονικό	Μικρό	50	84.37	1.21	13.77	0.65
Μέθοδος 1β	Κανονικό	Μεγάλο	50	84.55	1.08	13.64	0.74

Πίνακας 6. Αποτελέσματα μεθόδου 1 στα δεδομένα Image Segmentation (κανονικό μέγεθος χάρτη)

Μέθοδος	Μέγεθος χάρτη	Όριο εύρους τιμών BDV	Ελάχιστη ταυτοποίηση ορίων (%)	ΣΚ (%)	ΛΚ (%)	ΑΠ (%)	ΠΚ (%)
Μέθοδος 1α	Μεγάλο	Μικρό	10	85.11	1.60	12.73	0.56
Μέθοδος 1α	Μεγάλο	Μεγάλο	10	86.93	3.12	3.42	6.54
Μέθοδος 1β	Μεγάλο	Μικρό	10	86.49	2.12	4.46	6.93
Μέθοδος 1β	Μεγάλο	Μεγάλο	10	85.11	2.21	4.68	6.71
Μέθοδος 1α	Μεγάλο	Μικρό	25	82.86	1.77	14.55	0.82
Μέθοδος 1α	Μεγάλο	Μεγάλο	25	86.62	3.03	3.51	6.84
Μέθοδος 1β	Μεγάλο	Μικρό	25	86.36	2.12	4.42	7.10
Μέθοδος 1β	Μεγάλο	Μεγάλο	25	86.28	2.21	4.63	6.88
Μέθοδος 1α	Μεγάλο	Μικρό	50	81.60	2.77	10.52	5.11
Μέθοδος 1α	Μεγάλο	Μεγάλο	50	87.97	2.42	6.67	2.94
Μέθοδος 1β	Μεγάλο	Μικρό	50	82.86	3.25	12.42	1.47
Μέθοδος 1β	Μεγάλο	Μεγάλο	50	82.81	2.21	4.85	10.13

Πίνακας 7. Αποτελέσματα μεθόδου 1 στα δεδομένα Image Segmentation (μεγάλο μέγεθος χάρτη)

Σε πολλές από τις παραπάνω περιπτώσεις βρίσκουμε ότι το μεγαλύτερο ποσοστό των προτύπων που δεν έχουν κατηγοριοποιηθεί σωστά είναι τα πρότυπα που επαληθεύουν παραπάνω από έναν κανόνες. Για να ενταχθούν τα πρότυπα αυτά σε μία μόνο κατηγορία χρησιμοποιήθηκε το κριτήριο της ελάχιστης πιθανότητας λάθους. Αφού όλα τα πρότυπα δοκιμαστούν στο σύνολο κανόνων υπολογίζεται η πιθανότητα λάθους για κάθε κατηγορία. Για κάθε ένα από τα πρότυπα που επαληθεύουν παραπάνω από έναν κανόνες βρίσκεται ποιος κανόνας έχει τη μικρότερη πιθανότητα λάθους και θεωρείται ότι ανήκει στην κατηγορία αυτή. Με εφαρμογή αυτής της διαδικασίας τα αποτελέσματα είναι:

Μέθοδος	Μέγεθος χάρτη	Όριο εύρους τιμών BDV	Ελάχιστη ταυτοποίηση ορίων (%)	ΣΚ (%)	ΛΚ (%)	ΑΠ (%)
Μέθοδος 1α	Κανονικό	Μικρό	25	85.50	2.94	11.56
Μέθοδος 1α	Κανονικό	Μεγάλο	25	88.48	2.90	8.61
Μέθοδος 1β	Κανονικό	Μικρό	25	84.42	1.56	14.03
Μέθοδος 1β	Κανονικό	Μεγάλο	25	86.45	2.64	10.91
Μέθοδος 1α	Κανονικό	Μικρό	50	84.81	2.68	12.51
Μέθοδος 1α	Κανονικό	Μεγάλο	50	83.59	1.47	14.94
Μέθοδος 1β	Κανονικό	Μικρό	50	84.63	1.60	13.77
Μέθοδος 1β	Κανονικό	Μεγάλο	50	84.89	1.47	13.64

Πίνακας 8. Αποτελέσματα μεθόδου 1 στα δεδομένα Image Segmentation μετά την εφαρμογή του κριτηρίου ελάχιστης πιθανότητας λάθους (κανονικό μέγεθος χάρτη)

Μέθοδος	Μέγεθος χάρτη	Όριο εύρους τιμών BDV	Ελάχιστη ταυτοποίηση ορίων (%)	ΣΚ (%)	ΛΚ (%)	ΑΠ (%)
Μέθοδος 1α	Μεγάλο	Μικρό	10	85.45	1.82	12.73
Μέθοδος 1α	Μεγάλο	Μεγάλο	10	**91.73**	4.85	3.42
Μέθοδος 1β	Μεγάλο	Μικρό	10	91.52	4.03	4.46
Μέθοδος 1β	Μεγάλο	Μεγάλο	10	91.34	3.98	4.68
Μέθοδος 1α	Μεγάλο	Μικρό	25	83.46	1.99	14.55
Μέθοδος 1α	Μεγάλο	Μεγάλο	25	91.52	4.98	3.51
Μέθοδος 1β	Μεγάλο	Μικρό	25	91.56	4.03	4.42
Μέθοδος 1β	Μεγάλο	Μεγάλο	25	91.39	3.98	4.63
Μέθοδος 1α	Μεγάλο	Μικρό	50	85.45	4.03	10.52
Μέθοδος 1α	Μεγάλο	Μεγάλο	50	89.26	4.07	6.67
Μέθοδος 1β	Μεγάλο	Μικρό	50	83.64	3.94	12.42
Μέθοδος 1β	Μεγάλο	Μεγάλο	50	90.26	4.89	4.85

Πίνακας 9. Αποτελέσματα μεθόδου 1 στα δεδομένα Image Segmentation μετά την εφαρμογή του κριτηρίου ελάχιστης πιθανότητας λάθους (μεγάλο μέγεθος χάρτη)

Παρατηρούμε ότι στις περισσότερες περιπτώσεις έχουμε μεγαλύτερη αύξηση των σωστά κατηγοριοποιημένων προτύπων από την αύξηση των λανθασμένα κατηγοριοποιημένων προτύπων μετά την εφαρμογή του κριτηρίου της ελάχιστης πιθανότητας λάθους. Τα αποτελέσματα για μεγάλο μέγεθος χάρτη είναι καλύτερα από αυτά για μικρό μέγεθος που σημαίνει ότι οι κατηγορίες διαχωρίζονται καλύτερα σε μεγαλύτερο χάρτη.

Σε αυτό το σύνολο δεδομένων φαίνεται επίσης και η επίδραση των υπολοίπων παραμέτρων. Παρατηρούμε ότι όταν επιτρέπεται να υπάρχει μεγάλο εύρος τιμών υπάρχει μια σημαντική αύξηση στην ακρίβεια των κανόνων. Αυτό δείχνει ότι ακόμα και αν υπάρχουν τιμές BDV στις άκρες του χάρτη που είναι πολύ μεγαλύτερες από τον μέσο όρο δεν οδηγούν σε λάθος όρια αλλά αντίθετα επιτρέπουν τη δημιουργία ορίων που αλλιώς θα είχαν απορριφθεί και έτσι η αναζήτηση παράγει καλύτερα αποτελέσματα.

Στους πίνακες 8 και 9 παρατηρούμε επίσης ότι όσο μειώνεται το ελάχιστο ποσοστό ταυτοποίησης μεταξύ του ορίου στο SOM και του πίνακα της μεταβλητής τόσο καλύτερα αποτελέσματα παίρνουμε. Αυτό συμβαίνει διότι είναι δύσκολο ένα όριο μιας μεταβλητής να είναι το ίδιο "ξεκάθαρο" και στο SOM αφού εκεί συνδυάζονται και οι 19 μεταβλητές του συνόλου δεδομένων. Το καλύτερο σύνολο κανόνων δημιουργείται για μεγάλο μέγεθος χάρτη, μεγάλο όριο εύρους τιμών BDV και ελάχιστο ποσοστό ταυτοποίησης ορίων 10% και είναι:

IF Var19>-2.032394
AND Var18>0.3398911
AND Var13<25.08626
AND Var14>-13.10291
AND Var2<149.5705
AND Var16<-3.926524
AND Var11>3.817628 THEN class is 1

IF Var15>5.337083
AND Var16<-3.926524
AND Var2<162.7840

AND Var19<-1.508973
AND Var12>7.937839
AND Var17>8.095234
AND Var10<75.75962
AND Var14>-36.36754
AND Var6>0.1929839
AND Var10>27.52540
AND Var11>25.96549
AND Var18<0.3708840
AND Var19>-2.169582 THEN class is 2

IF Var10<104.0393
AND Var14<1.912578
AND Var15<71.41415
AND Var4<0.1224296
AND Var12<80.87550
AND Var2>69.08885
AND Var11<38.12885
AND Var2<143.9782
AND Var19<-2.089521
AND Var6>0.1929839
AND Var16>-9.206002 THEN class is 3

IF Var2>143.9782
AND Var15<11.84375
AND Var16>-7.328961
AND Var13>6.445298
AND Var18>0.1619028 THEN class is 4

IF Var2>149.5705
AND Var18<0.3596714
AND Var10>20.27115
AND Var12>22.68713
AND Var15>11.84375
AND Var4<0.1224296 THEN class is 5

IF Var10>75.75962
AND Var6<3.029512 THEN class is 6

IF Var13<30.99179
AND Var12<45.18722
AND Var2<161.9073
AND Var15<36.41600
AND Var19<0.3649871
AND Var7<145.3272
AND Var14<1.912578
AND Var11<25.96549
AND Var19>-2.237582
AND Var16>-13.44761 THEN class is 7

7.4 Αποτελέσματα μεθόδου εξαγωγής κανόνων με βάση στατιστικές ιδιότητες του SOM

Η εφαρμογή αυτή δοκιμάστηκε με διαφορετικούς τρόπους επιλογής μεταβλητών που περιγράφουν κάθε ομάδα και με διαφορετικές τεχνικές ομαδοποίησης. Ονομάζουμε "μέθοδο 2α" τη μέθοδο που βασίζεται στις στατιστικές ιδιότητες του SOM και γίνεται χρήση του αλγορίθμου sig* για την επιλογή των σημαντικών μεταβλητών με πίνακα σπουδαιότητας το αντίστροφο της τυπικής απόκλισης της μεταβλητής στην ομάδα, "μέθοδο 2β" τη μέθοδο όπου πίνακας σπουδαιότητας είναι ο λόγος της τυπικής απόκλισης της μεταβλητής σε όλο το χάρτη προς την τυπική απόκλιση της μεταβλητής στην ομάδα ενώ "μέθοδος 2γ" είναι αυτή που γίνεται επιλογή ομάδων μεταβλητών. Στον αλγόριθμο sig* επιλέγονται μεταβλητές μέχρι οι αθροιστικές τιμές σπουδαιότητας να ξεπεράσουν το 50%.

Για την ομαδοποίηση εφαρμόστηκε ο αλγόριθμος των κ-μέσων και ομαδοποίηση με ιεραρχικό συσσωρευτικό αλγόριθμο. Στον ιεραρχικό αλγόριθμο δοκιμάστηκαν διαφορετικοί τρόποι υπολογισμού αποστάσεων ομάδων όπως απλή, πλήρης, μέση, κεντροειδής και Ward. Κάποιοι τρόποι υπολογισμού αποστάσεων, όπως η απλή σύνδεση, δε δημιούργησαν ικανοποιητική περιγραφή του χάρτη και στις περιπτώσεις αυτές δεν αναφέρονται καθόλου τα σχετικά αποτελέσματα αφού είχαν πολύ μικρή ακρίβεια. Στα αποτελέσματα που ακολουθούν αναφέρεται αν έχει χρησιμοποιηθεί ο αλγόριθμος των κ-μέσων ή δίνεται το όνομα των αποστάσεων του ιεραρχικού αλγορίθμου.

7.4.1 Σύνολο δεδομένων Ionosphere

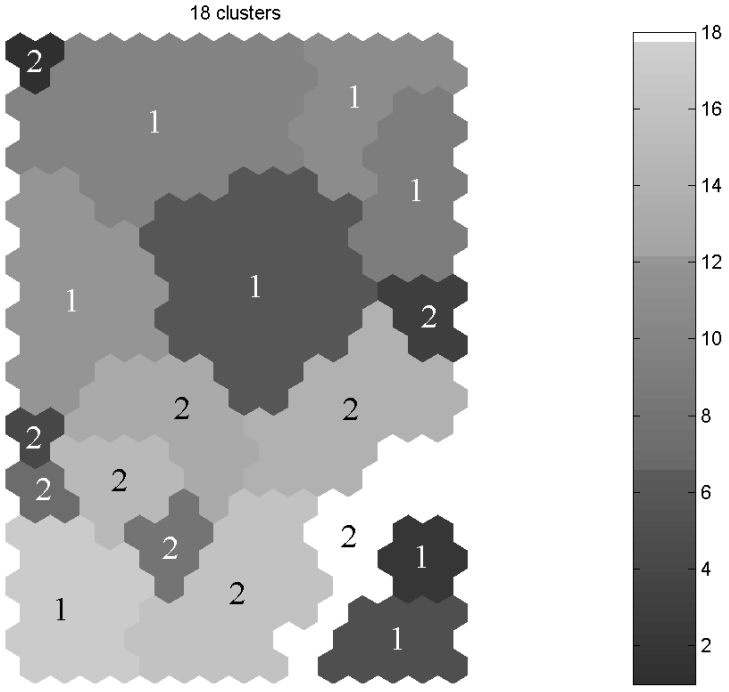
Για το σύνολο δεδομένων Ionosphere βρέθηκε ότι όπως και στην πρώτη μέθοδο οι κανόνες είναι πολύ πιο ακριβείς για την πρώτη και μεγαλύτερη κατηγορία δεδομένων οπότε είναι προτιμότερο να θεωρείται ότι τα πρότυπα που δεν ικανοποιούν τον κανόνα της πρώτης κατηγορίας ανήκουν στη δεύτερη. Τα αποτελέσματα για μεγάλο μέγεθος χάρτη είναι καλύτερα οπότε οι περισσότερες δοκιμές είναι σε τέτοιο χάρτη. Στη μέθοδο επιλογής ομάδων μεταβλητών αρχικά τέθηκε να επιλέγονται 5 ομάδες που να περιέχουν από 4 μεταβλητές. Η επιλογή αυτή οδήγησε σε μεγάλο χρόνο υπολογισμού και μεγάλο αριθμό κανόνων. Για το λόγο αυτό η επιλογή μειώθηκε σε 5 ομάδες που να περιέχουν από 3 μεταβλητές μειώνοντας έτσι κατά πολύ τη διάρκεια των υπολογισμών χωρίς κάποιο αρνητικό αντίκτυπο στην ακρίβεια των κανόνων. Επίσης για περαιτέρω μείωση του χρόνου υπολογισμού αποκλείστηκε από την αναζήτηση η μεταβλητή 2 του συνόλου δεδομένων αφού έχει μηδενική τιμή για όλα τα πρότυπα άρα δεν χρησιμεύει καθόλου στην κατηγοριοποίηση. Τα αποτελέσματα που προκύπτουν είναι:

Μέθοδος	Μέγεθος χάρτη	Ομαδοποίηση	Πλήθος ομάδων	ΣΚ (%)	ΛΚ (%)
Μέθοδος 2α	Κανονικό	κ-μέσων	10	77.78	22.22
Μέθοδος 2β	Κανονικό	κ-μέσων	10	86.32	13.68
Μέθοδος 2γ	Κανονικό	κ-μέσων	10	90.31	9.69
Μέθοδος 2α	Μεγάλο	κ-μέσων	19	81.48	18.52
Μέθοδος 2β	Μεγάλο	κ-μέσων	19	88.32	11.68
Μέθοδος 2γ	Μεγάλο	κ-μέσων	17	88.89	11.11
Μέθοδος 2β	Μεγάλο	Κεντροειδής	18	87.18	12.82
Μέθοδος 2γ	Μεγάλο	Κεντροειδής	18	83.76	16.24
Μέθοδος 2β	Μεγάλο	Πλήρης	18	**90.60**	9.40
Μέθοδος 2γ	Μεγάλο	Πλήρης	18	88.60	11.40
Μέθοδος 2β	Μεγάλο	Ward	19	**90.60**	9.40
Μέθοδος 2γ	Μεγάλο	Ward	19	84.33	15.67
Μέθοδος 2β	Μεγάλο	Μέση	19	88.32	11.68
Μέθοδος 2γ	Μεγάλο	Μέση	19	83.19	16.81

Πίνακας 10. Αποτελέσματα μεθόδου 2 στα δεδομένα Ionosphere

Από τα παραπάνω παρατηρούμε ότι η μέθοδος όπου επιλέγονται μεταβλητές μόνο με βάση την τυπική απόκλιση μέσα στην ομάδα (μέθοδος 2α) δίνει αρκετά χειρότερα αποτελέσματα από τις υπόλοιπες. Οι δύο άλλες μέθοδοι δίνουν περίπου ίδια αποτελέσματα με τη μέθοδο 2β να υπερτερεί ελαφρώς. Το καλύτερο σύνολο κανόνων παράγεται από τη μέθοδο 2β ενώ έχει προηγηθεί ιεραρχική ομαδοποίηση με απόσταση Ward ή πλήρη. Η ομαδοποίηση και η αντιστοίχηση κατηγοριών στις ομάδες φαίνεται στο σχήμα 23. Ενδεικτικά οι κανόνες για πλήρη απόσταση είναι:

IF Var1>0.8063240 AND Var1<1.109379
AND Var5>0.1431831 AND Var5<1.091216
AND Var3>0.2600765 AND Var3<1.117424 THEN class is 1 (“good”)
ELSE class is 2 (“bad”)



Σχήμα 19. Αντιστοίχηση ομάδων και κατηγοριών στα δεδομένα Ionosphere

7.4.2 Σύνολο δεδομένων Iris

Η εφαρμογή της μεθόδου στο σύνολο δεδομένων Iris έγινε μόνο σε χάρτη κανονικού μεγέθους αφού όπως βρέθηκε και στην πρώτη μέθοδο είναι κατάλληλο μέγεθος για να δώσει καλή κατηγοριοποιήση. Στη μέθοδο επιλογής ομάδων μεταβλητών τέθηκε να επιλέγεται μία ομάδα που να περιέχει δύο μεταβλητές. Βρέθηκε σε πολλές περιπτώσεις ότι οι κανόνες διαχωρίζουν τέλεια την πρώτη κατηγορία αλλά δεν περιγράφουν καλά την τρίτη. Σε αυτές τις περιπτώσεις θεωρείται ότι στην τρίτη κατηγορία ανήκουν τα πρότυπα που δεν επαληθεύουν κανέναν από τους κανόνες των δύο πρώτων κατηγοριών . Τα αποτελέσματα είναι:

Μέθοδος	Μέγεθος χάρτη	Ομαδοποίηση	Πλήθος ομάδων	ΣΚ (%)	ΛΚ (%)	ΑΠ (%)	ΠΚ (%)
Μέθοδος 2α	Κανονικό	κ-μέσων	7	88.00	1.33	10.00	0.67
Μέθοδος 2β	Κανονικό	κ-μέσων	7	**95.33**	4.67	0.00	0.00
Μέθοδος 2γ	Κανονικό	κ-μέσων	8	92.00	8.00	0.00	0.00
Μέθοδος 2β	Κανονικό	Κεντροειδής	3	94.00	6.00	0.00	0.00
Μέθοδος 2γ	Κανονικό	Κεντροειδής	3	92.00	8.00	0.00	0.00
Μέθοδος 2β	Κανονικό	Πλήρης	8	90.67	0.67	2.67	6.00
Μέθοδος 2γ	Κανονικό	Πλήρης	8	84.00	2.67	7.33	6.00
Μέθοδος 2β	Κανονικό	Ward	8	92.67	2.00	2.67	2.67
Μέθοδος 2γ	Κανονικό	Ward	8	**95.33**	4.67	0.00	0.00
Μέθοδος 2β	Κανονικό	Μέση	3	94.00	6.00	0.00	0.00
Μέθοδος 2γ	Κανονικό	Μέση	3	92.00	8.00	0.00	0.00

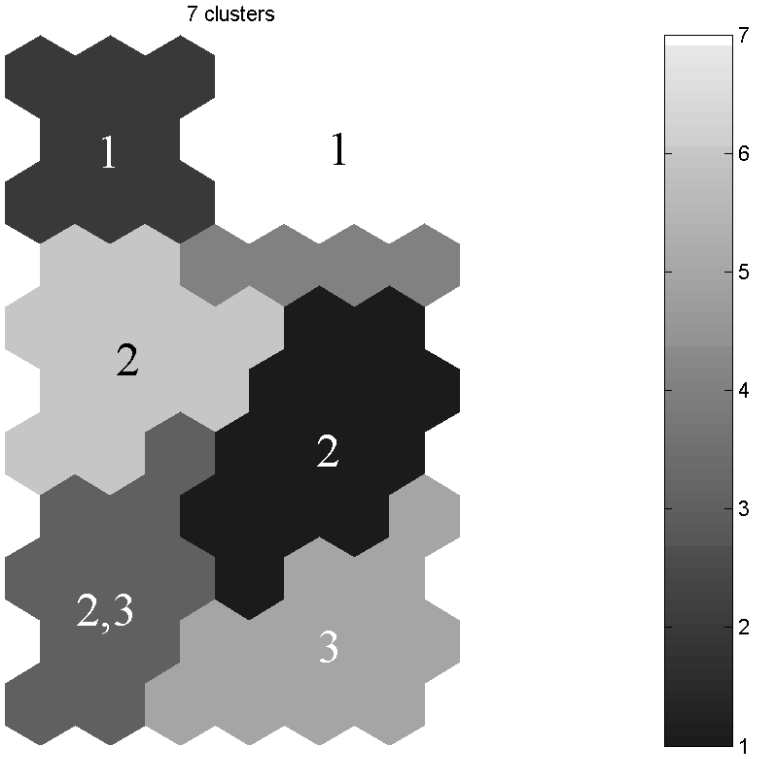
Πίνακας 11. Αποτελέσματα μεθόδου 2 στα δεδομένα Iris

Καλύτερη κατηγοριοποίηση παίρνουμε με τη μέθοδο 2β και ομαδοποίηση κ-μέσων και με τη μέθοδο 2γ και απόσταση Ward. Η ομαδοποίηση και η αντιστοίχηση κατηγοριών στις ομάδες φαίνεται στο σχήμα 24. Ενδεικτικά οι κανόνες από τη μέθοδο 2β με ομαδοποίηση κ-μέσων είναι:

IF (Var4>0.07005605 AND Var4<0.5160947)
OR (Var3>1.162711 AND Var3<2.085595) THEN class is 1 (Setosa)

ELSE IF (Var3>2.954204 AND Var3<4.708537)
OR (Var4>0.8688886 AND Var4<1.469599) THEN class is 2 (Versicolor)

ELSE class is 3 (Virginica)



Σχήμα 20. Αντιστοίχηση ομάδων και κατηγοριών στα δεδομένα Iris

7.4.3 Σύνολο δεδομένων Image Segmentation

Στο σύνολο δεδομένων Image Segmentation παρατηρήθηκε ότι μερικές φορές για κάποια κατηγορία δε δημιουργείται κανένας κανόνας. Στις περιπτώσεις αυτές στη κατηγορία αυτή εντάσσονται όλα τα πρότυπα που δεν έχουν κατηγοριοποιηθεί σε καμία άλλη. Σε περιπτώσεις που αυτό συμβαίνει για δύο ή παραπάνω κατηγορίες δεν είναι δυνατό να σχηματιστεί σύνολο κανόνων. Στη μέθοδο επιλογής ομάδων μεταβλητών τέθηκε να επιλέγονται δύο ομάδες που να περιέχουν 7 μεταβλητές. Επίσης από την αναζήτηση εξαιρέθηκε η μεταβλητή 3 γιατί έχει την ίδια τιμή για όλα τα πρότυπα. Τα αποτελέσματα που προκύπτουν είναι:

Μέθοδος	Μέγεθος χάρτη	Ομαδοποίηση	Πλήθος ομάδων	ΣΚ (%)	ΛΚ (%)	ΑΠ (%)	ΠΚ (%)
Μέθοδος 2β	Κανονικό	κ-μέσων	16	76.23	21.56	0.00	2.21
Μέθοδος 2γ	Κανονικό	κ-μέσων	15	61.95	27.06	0.00	11.00
Μέθοδος 2α	Μεγάλο	κ-μέσων	27	6.75	31.00	1.73	60.52
Μέθοδος 2β	Μεγάλο	κ-μέσων	31	59.31	5.63	19.22	15.84
Μέθοδος 2γ	Μεγάλο	κ-μέσων	31	65.93	5.50	22.38	6.19
Μέθοδος 2β	Κανονικό	Ward	15	69.74	7.14	20.35	2.77
Μέθοδος 2β	Μεγάλο	Ward	26	58.35	4.20	16.28	21.17
Μέθοδος 2γ	Μεγάλο	Ward	26	52.94	6.80	20.22	20.04
Μέθοδος 2β	Μεγάλο	Μέση	28	60.13	27.01	0.00	12.86
Μέθοδος 2γ	Μεγάλο	Μέση	28	58.44	26.32	0.00	15.24

Πίνακας 12. Αποτελέσματα μεθόδου 2 στα δεδομένα Image Segmentation

Τα αποτελέσματα μετά την εφαρμογή του κριτηρίου της ελάχιστης πιθανότητας λάθους για τα πρότυπα που επαληθεύουν παραπάνω από έναν κανόνες:

Μέθοδος	Μέγεθος χάρτη	Ομαδοποίηση	Πλήθος ομάδων	ΣΚ (%)	ΛΚ (%)	ΑΠ (%)
Μέθοδος 2β	Κανονικό	κ-μέσων	16	**77.23**	22.77	0.00
Μέθοδος 2γ	Κανονικό	κ-μέσων	15	70.30	29.70	0.00
Μέθοδος 2α	Μεγάλο	κ-μέσων	27	40.48	57.79	1.73
Μέθοδος 2β	Μεγάλο	κ-μέσων	31	74.20	6.58	19.22
Μέθοδος 2γ	Μεγάλο	κ-μέσων	31	71.73	5.89	22.38
Μέθοδος 2β	Κανονικό	Ward	15	71.95	7.71	20.35
Μέθοδος 2β	Μεγάλο	Ward	26	76.93	6.80	16.28
Μέθοδος 2γ	Μεγάλο	Ward	26	66.15	13.64	20.22
Μέθοδος 2β	Μεγάλο	Μέση	28	70.17	29.83	0.00
Μέθοδος 2γ	Μεγάλο	Μέση	28	68.48	31.52	0.00

Πίνακας 13. Αποτελέσματα μεθόδου 2 στα δεδομένα Image Segmentation μετά την εφαρμογή του κριτηρίου ελάχιστης πιθανότητας λάθους

Από τον παραπάνω πίνακα παρατηρούμε ότι και σε αυτή την περίπτωση το κριτήριο της τυπικής απόκλισης δε δίνει καθόλου ικανοποιητικά αποτελέσματα. Το καλύτερο σύνολο κανόνων δημιουργήθηκε με τη μέθοδο 2β και τον αλγόριθμο των κ-μέσων. Η ομαδοποίηση και η αντιστοίχηση κατηγοριών στις ομάδες φαίνεται στο σχήμα 25 και οι κανόνες είναι:

IF Var14>-5.678044 AND Var14<4.558214
AND Var19>-2.187392 AND Var19<-0.02675546
AND Var18>0.2554172 AND Var18<0.6971410 THEN class is 1

IF Var7>-8.714179 AND Var7<79.31937
AND Var5>-8.910676e-007 AND Var5<1.186567e-006
AND Var19>-2.183330 AND Var19<-1.795701
AND Var2>59.72355 AND Var2<159.6405
AND Var18>0.2428823 AND Var18<0.4456318 THEN class is 2

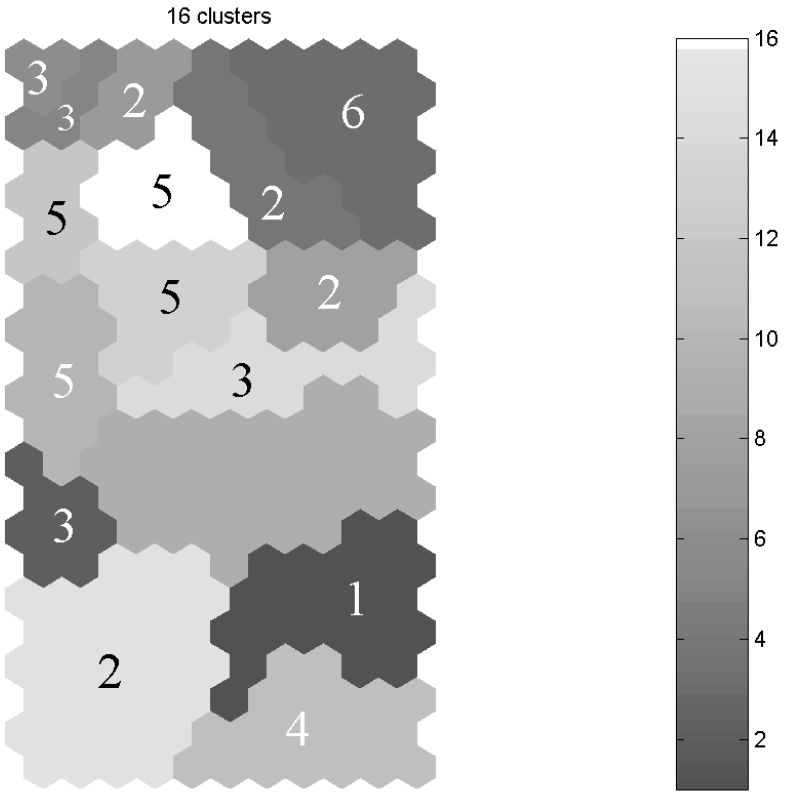
IF Var18>0.6878433 AND Var18<1.030352 THEN class is 3

IF Var19>0.3459164 AND Var19<2.937254 THEN class is 4

IF Var2>126.0921 AND Var2<209.3252
AND Var18>0.2417755 AND Var18<0.3801989
AND Var7>0.6830344 AND Var7<8.635406
AND Var6>1.256731 AND Var6<4.554008 THEN class is 5

IF Var17>111.2423 AND Var17<150.0378 THEN class is 6

IF UNCLASSIFIED THEN class is 7



Σχήμα 21. Αντιστοίχηση ομάδων και κατηγοριών στα δεδομένα Image Segmentation

7.5 Συμπεράσματα

Η μεθοδολογία εξαγωγής κανόνων με βάση τον U-matrix εφαρμόστηκε με επιτυχία και στα τρία διαφορετικά σύνολα δεδομένων. Δεν είναι όμως δυνατό να βρεθεί θεωρητικά, και ούτε πειραματικά όπως αποδείχθηκε, ένας βέλτιστος συνδυασμός παραμέτρων που να δίνει σε όλες τις περιπτώσεις πολύ καλά αποτελέσματα. Αρχικά πρέπει να καθορίζεται το μέγεθος του SOM που μπορεί να προσφέρει μια καλή κατηγοριοποίηση των δεδομένων. Στη συνέχεια μπορεί να εφαρμοστεί ο αλγόριθμος με διαφορετικές παραμέτρους και να εξεταστεί ποιες δίνουν τα καλύτερα αποτελέσματα. Αν η περιγραφή που επιτυγχάνεται με τους κανόνες δεν είναι ικανοποιητική πρέπει να μειώνεται το ποσοστό ταυτοποίησης ορίων ώστε να δημιουργούνται περισσότεροι κανόνες που πιθανώς θα οδηγήσουν και σε καλύτερα αποτελέσματα.

Η τακτική αυτή δοκιμάστηκε και στα τρία σύνολα δεδομένων του πειράματος. Στο Iris βρέθηκε ότι οι τιμές των παραμέτρων δεν επηρεάζουν ουσιαστικά το αποτέλεσμα. Η επίδραση των παραμέτρων ήταν εμφανής στα άλλα σύνολα δεδομένων και ιδιαίτερα στο Image Segmentation που είναι και το πιο πολύπλοκο από τα τρία. Η μείωση του ελάχιστου ποσοστού ταυτοποίησης ορίων οδήγησε σε ελαφρά πιο ακριβείς κανόνες στο Ionosphere και αρκετά πιο ακριβείς κανόνες στο Image Segmentation. Το γεγονός αυτό επιβεβαιώνει την υπόθεση ότι τα σημαντικά όρια στους πίνακες των μεταβλητών, ακόμα και αν δεν είναι εμφανή στο SOM, μπορούν να παράγουν έγκυρους κανόνες. Ανάλογη αύξηση στην ακρίβεια των κανόνων είχαμε και στα δύο αυτά σύνολα δεδομένων με αύξηση του επιτρεπόμενου εύρους ορίων για τις τιμές BDV. Αυτό είναι περισσότερο φανερό στη μέθοδο 1α αφού στη μέθοδο 1β οι τιμές BDV δεν έχουν έτσι και αλλιώς μεγάλο εύρος επειδή υπολογίζονται με διαφορετικό τρόπο.

Η μεθοδολογία εξαγωγής κανόνων με βάση τις στατιστικές ιδιότητες του SOM δεν οδηγεί σε τόσο καλά αποτελέσματα όσο η προηγούμενη. Και από τα τρία σύνολα δεδομένων είναι φανερό ότι η χρήση της τυπικής απόκλισης της μεταβλητής μέσα στην ομάδα ως κριτήριο επιλογής σημαντικών μεταβλητών δεν οδηγεί σε καλά αποτελέσματα, όπως ήταν αναμενόμενο αφού δε λαμβάνονται καθόλου υπόψη οι υπόλοιπες ομάδες. Η απόκλιση ακόμα και για το απλό σύνολο Iris είναι μεγάλη ενώ στα υπόλοιπα σύνολα δεδομένων η διαφορά από τα άλλα κριτήρια είναι τόσο μεγάλη που συμπεραίνουμε ότι η χρήση του κριτηρίου αυτού θα πρέπει να αποφεύγεται.

Η χρήση του λόγου της τυπικής απόκλισης σε όλο το χάρτη προς την τυπική απόκλιση της ομάδας οδηγεί σε πολύ καλύτερα σύνολα κανόνων και αποδεικνύεται ότι σε όλα τα σύνολα δεδομένων αποτελεί το καλύτερο κριτήριο επιλογής μεταβλητών από αυτά που δοκιμάστηκαν. Η επιλογή ομάδων μεταβλητών με αναζήτηση όλων των συνδυασμών οδηγεί σε λίγο χειρότερα αποτελέσματα από το λόγο των αποκλίσεων στα σύνολα Iris και Ionosphere. Στο Image Segmentation υπάρχουν και περιπτώσεις που δεν έχει καλή απόδοση σε σχέση με το λόγο των τυπικών αποκλίσεων. Επιπλέον η μέθοδος αυτή έχει το μειονέκτημα της πολύ μεγαλύτερης διάρκειας εκτέλεσης ιδαίτερα στα σύνολα Ionosphere και Image Segmentation που έχουν πολλές μεταβλητές και στο χάρτη τους δημιουργείται μεγάλος αριθμός ομάδων οπότε ο όγκος των υπολογισμών αυξάνεται ακόμα περισσότερο.

Η μειωμένη ακρίβεια της δεύτερης μεθόδου μπορεί να οφείλεται σε διάφορους λόγους. Όποια τεχνική ομαδοποίησης και αν χρησιμοποιηθεί εισάγει αναπόφευκτα λάθη στην περιγραφή. Σε ομάδες που αντιστοιχίζονται δύο ή παραπάνω κατηγορίες δεδομένων θα πρέπει να γίνεται εκ νέου ομαδοποίηση ώστε να υπάρχει καλύτερος διαχωρισμός. Η χρήση της κανονικής κατανομής για την περιγραφή των τιμών μιας μεταβλητής σε μία ομάδα είναι μία απλοποιημένη υπόθεση αλλά η εύρεση της πραγματικής κατανομής των δεδομένων είναι μία επίσης πολύπλοκη διαδικασία. Άλλος παράγοντας που μπορεί να βελτιωθεί είναι ο τρόπος επιλογής μεταβλητών για την περιγραφή μιας ομάδας.

Από την εφαρμογή των μεθόδων αποδείχθηκε ότι είναι εφικτό να γίνει ορθή εξαγωγή κανόνων από ένα SOM και να περιγραφούν τα δεδομένα με ακρίβεια από ένα μικρό σύνολο κανόνων. Σε κάθε περίπτωση το σύνολο κανόνων αποτελείται από έναν απλό κανόνα για κάθε κατηγορία. Κάθε κανόνας έχει ένα μικρό πλήθος κατηγορημάτων που είναι πάντα κατά πολύ μικρότερος από το πλήθος των μεταβλητών του συνόλου δεδομένων. Η απλότητα και το μικρό μέγεθος των κανόνων τους καθιστά εύκολα κατανοητούς και ένας χρήστης μπορεί γρήγορα να καταλάβει ποιες μεταβλητές χαρακτηρίζουν κάθε κατηγορία και σε ποιες περιοχές κινούνται οι τιμές τους. Οι κανόνες αυτοί συνδυασμένοι με τα πλεονεκτήματα του SOM μπορούν να βοηθήσουν σημαντικά στην εύκολη κατανόηση ενός συνόλου δεδομένων.

Παραρτήματα

Παράρτημα Α. Πηγαίος κώδικας μεθόδου 1

Η υλοποίηση των προγραμμάτων έγινε στη γλώσσα προγραμματισμού Matlab® έκδοση 6.5 κάνοντας χρήση της ανοιχτού κώδικα βιβλιοθήκης συναρτήσεων SOM Toolbox [16].

- Κεντρικό αρχείο της εφαρμογής για τα δεδομένα Iris. Η μετατροπή του για τα υπόλοιπα δεδομένα γίνεται με απλή αλλαγή των δεδομένων εισόδου, του πλήθους των κατηγοριών και του πλήθους των προτύπων σε κάθε κατηγορία.

```
% Iris data
warning off MATLAB:divideByZero
clear all;
clc;
close all;

global neighborUnits mapSize labelThreshold numOfVars bdvThreshold
alreadySelected numClasses classLabels ruleCell alternateRules
currentVariable

% Load data
startTime = clock;
load('iris.mat');

% Label Data
for i = 1:size(DataIn,1)
    s = sprintf('%1.0d',OutClass(i));
    c(i) = cellstr(s);
end
OutClassStr = c';
sD = som_data_struct(DataIn,'labels',OutClassStr,'name','IRIS');
sD = som_normalize(sD,'var');
clear c OutClassStr s

% Make SOM
sM = som_make(sD);
sM = som_autolabel(sM,sD,'vote');

% Specifications
numOfVars = size(DataIn,2);
mapSize = sM.topol.msize;
numClasses = 3; % (Dataset specific)
instancesPerClass = [50 50 50]; % (Dataset specific)
bdvThreshold = 4; % Threshold for BDV relative values
boundThreshold = 25; % Threshold for matching boundaries
labelThreshold = 0.75; % Threshold for associating labels with classes

% Basic Visualization
%
som_show(sM,'umat','all','comp',1:numOfVars,'empty','Labels','norm','d');
% som_show_add('label',sM,'subplot',numOfVars+2);
% figure;som_show(sM,'umat',{'all','U-matrix (total)'},'empty','Labels');
```

```
% som_show_add('label',sM,'subplot',2);

% U-matrix calculation
U = som_umat(sM,'mode','mean');
e = eye(numOfVars);
% U-matrix calculation for each variable
for i = 1:numOfVars
    UVar(:,:,i) = som_umat(sM,'mask',e(:,i),'mode','mean');
end

% BDV Matrices calculation
bdvMatrix = BDVMatrixcalcRatio(U);
% bdvMatrix = BDVMatrixcalcDiff(U);
for i = 1:numOfVars
    bdvVar(:,:,i) = BDVMatrixcalcRatio(UVar(:,:,i));
%     bdvVar(:,:,i) = BDVMatrixcalcDiff(UVar(:,:,i));
end

% Map codebook (denormalized)
sM = som_denormalize(sM);
mcd = sM.codebook;

% Component matrices
var = reshape(mcd,[mapSize(1) mapSize(2) numOfVars]);

% Map labels
classLabels = sM.labels;
classLabels = reshape(classLabels,[mapSize(1) mapSize(2)]);

% Find neighboring units on the map
neighborUnits = som_unit_neighs(sM.topol);

% Initialization
alreadySelected = zeros(mapSize(1),mapSize(2));
boundary = zeros(mapSize(1),mapSize(2));
ruleOperands = zeros(2,1);
ruleCell = cell(numClasses,1);
bestRules = cell(numClasses,1);
alternateRules = cell(numClasses,1);
secondaryRules = cell(numClasses,1);
entered = zeros(numClasses,size(DataIn,2),2);

[C,I] = min(bdvMatrix);
[C2,I2] = min(C);
minIndex = [I(I2) I2];
boundarySize = 1;
while boundarySize>0
    % Find boundary on the SOM
    [boundary,noBoundary] = FindGeneralBoundaries(bdvMatrix,minIndex);
    alreadySelected = alreadySelected+boundary;
    boundarySize = find(boundary);
    while length(boundarySize)<2 || noBoundary==1
        if length(boundarySize)==0 % No boundary formed
            break;
        end
        [boundary,noBoundary] =
FindGeneralBoundaries(bdvMatrix,minIndex);
        alreadySelected = alreadySelected+boundary;
        boundarySize = find(boundary);
    end
```

```
    if length(boundarySize)==0 % No boundaries left
        break;
    end

    for currentVariable = 1:numOfVars
        bdvTestComp = bdvVar(:,:,currentVariable);
        currentVariableMatrix = var(:,:,currentVariable);
        % Find boundary on component matrix
        [componentBoundary,matchPercentage] =
FindComponentBoundary(boundary,bdvTestComp);
        if matchPercentage>boundThreshold % If there is a match
            % Extract rule threshold from boundary
            ruleThreshold =
BoundaryThrehold(componentBoundary,currentVariableMatrix);
            % Find sections on the map
            [section1,section2,ruleOperands] =
RuleSections(componentBoundary,currentVariableMatrix,ruleThreshold);
            % Create rules
            [ruleCell,alternateRules] =
IrisRuleCreation(section1,section2,ruleThreshold,ruleOperands);
        end
    end
end

% Rule post-processing
run RulePostProccessing

fid = fopen('./results/sxolia21.txt','w');
fprintf(fid,sprintf('Iris Data\n'));
fprintf(fid,sprintf('Map size is normal\n'));
fprintf(fid,sprintf('BDVThres = %d\n',bdvThreshold));
fprintf(fid,sprintf('BoundThreshold = %d\n',boundThreshold));
fprintf(fid,sprintf('labelThreshold = %d\n',labelThreshold));

run RuleOutput

endTime = clock;
fprintf(fid,sprintf('\n Start Time Hour=%d Min=%d
Sec=%d\n',startTime(4),startTime(5),startTime(6)));
fprintf(fid,sprintf('\n End Time   Hour=%d Min=%d
Sec=%d\n',endTime(4),endTime(5),endTime(6)));
fclose(fid);
```

- BDVMatrixcalcRatio.m

```
function bdvMatrix = BDVMatrixcalcRatio(U)
% Calculate BDV matrix
% Input arguments: U-matrix
% Output arguments: BDV matrix

global mapSize

y = mapSize(1);
x = mapSize(2);

bdvMatrix = zeros(mapSize(1),mapSize(2));

for j = 1:mapSize(1)
```

```
uj = 2*j-1;
for i=1:mapSize(2)
    ui = 2*i-1;
    bdvCandidate = zeros(3,1);
    if j==1 && i==1    %top left corner
        Ml = U(uj,ui+1);
        Mo = U(uj+1,ui);
        Ro = Mo;
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj+1,ui);
        Mo = U(uj,ui+1);
        Ro = Mo;
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
    elseif j==1 && i>1 && i<mapSize(2)     % Upper edge
        Ml = (U(uj,ui-1)+U(uj,ui+1))/2;
        tmp = [U(uj+1,ui) U(uj+1,ui-1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj+1,ui);
        tmp = [U(uj,ui-1) U(uj+1,ui-1) U(uj,ui+1)];
        Mo = (sum(tmp))/3;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj+1,ui-1);
        tmp = [U(uj+1,ui) U(uj,ui-1) U(uj,ui+1)];
        Mo = (sum(tmp))/3;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
    elseif j==1 && i==mapSize(2)      % Top right corner
        Ml = U(uj,ui-1);
        tmp = [U(uj+1,ui) U(uj+1,ui-1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj+1,ui);
        tmp = [U(uj,ui-1) U(uj+1,ui-1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj+1,ui-1);
        tmp = [U(uj+1,ui) U(uj,ui-1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
    elseif j>1 && j<mapSize(1) && i==mapSize(2)    % Right edge
        if rem(j,2)==0
            Ml = U(uj,ui-1);
            tmp = [U(uj+1,ui) U(uj-1,ui)];
            Mo = (sum(tmp))/2;
            Ro = max(tmp)-min(tmp);
            bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
            Ml = U(uj+1,ui);
            tmp = [U(uj,ui-1) U(uj-1,ui)];
            Mo = (sum(tmp))/2;
            Ro = max(tmp)-min(tmp);
            bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
            Ml = U(uj-1,ui);
            tmp = [U(uj+1,ui) U(uj,ui-1)];
            Mo = (sum(tmp))/2;
            Ro = max(tmp)-min(tmp);
```

```
            bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
        else
            Ml = (U(uj-1,ui)+U(uj+1,ui-1))/2;
            tmp = [U(uj-1,ui-1) U(uj+1,ui) U(uj,ui-1)];
            Mo = (sum(tmp))/3;
            Ro = max(tmp)-min(tmp);
            bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
            Ml = (U(uj-1,ui-1)+U(uj+1,ui))/2;
            tmp = [U(uj-1,ui) U(uj+1,ui-1) U(uj,ui-1)];
            Mo = (sum(tmp))/3;
            Ro = max(tmp)-min(tmp);
            bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
            Ml = U(uj,ui-1);
            tmp = [U(uj-1,ui-1) U(uj+1,ui) U(uj-1,ui) U(uj+1,ui-1)];
            Mo = (sum(tmp))/4;
            Ro = max(tmp)-min(tmp);
            bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
        end
    elseif j==mapSize(1) && i==mapSize(2)     % Bottom right corner
        Ml = U(uj,ui-1);
        tmp = [U(uj-1,ui) U(uj-1,ui-1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj-1,ui);
        tmp = [U(uj,ui-1) U(uj-1,ui-1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj-1,ui-1);
        tmp = [U(uj-1,ui) U(uj,ui-1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
    elseif j==mapSize(1) && i>1 && i<mapSize(2)     % Lower edge
        Ml = (U(uj,ui-1)+U(uj,ui+1))/2;
        tmp = [U(uj-1,ui) U(uj-1,ui-1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj-1,ui);
        tmp = [U(uj,ui-1) U(uj-1,ui-1) U(uj,ui+1)];
        Mo = (sum(tmp))/3;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj-1,ui-1);
        tmp = [U(uj-1,ui) U(uj,ui-1) U(uj,ui+1)];
        Mo = (sum(tmp))/3;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
    elseif j==mapSize(1) && i==1     % Bottom left corner
        Ml = U(uj,ui+1);
        %tmp = [];
        Mo = U(uj-1,ui);
        Ro = Mo;
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj-1,ui);
        %tmp = [];
        Mo = U(uj,ui+1);
        Ro = Mo;
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
```

```
elseif j>1 && j<mapSize(1) && i==1    % Left edge
    if rem(j,2)==0
        Ml = (U(uj-1,ui)+U(uj+1,ui+1))/2;
        tmp = [U(uj-1,ui+1) U(uj+1,ui) U(uj,ui+1)];
        Mo = (sum(tmp))/3;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = (U(uj-1,ui+1)+U(uj+1,ui))/2;
        tmp = [U(uj-1,ui) U(uj+1,ui+1) U(uj,ui+1)];
        Mo = (sum(tmp))/3;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
        Ro = 4;
        Ml = U(uj,ui+1);
        tmp = [U(uj-1,ui+1) U(uj+1,ui) U(uj-1,ui) U(uj+1,ui+1)];
        Mo = (sum(tmp))/4;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
    else
        Ml = U(uj,ui+1);
        tmp = [U(uj+1,ui) U(uj-1,ui)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj+1,ui);
        tmp = [U(uj,ui+1) U(uj-1,ui)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
        Ml = U(uj-1,ui);
        tmp = [U(uj+1,ui) U(uj,ui+1)];
        Mo = (sum(tmp))/2;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
    end
else                    % Middle part
    if rem(j,2)==0
        Ml = (U(uj-1,ui)+U(uj+1,ui+1))/2;
        tmp = [U(uj-1,ui+1) U(uj+1,ui) U(uj,ui+1) U(uj,ui-1)];
        Mo = (sum(tmp))/4;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = (U(uj-1,ui+1)+U(uj+1,ui))/2;
        tmp = [U(uj-1,ui) U(uj+1,ui+1) U(uj,ui+1) U(uj,ui-1)];
        Mo = (sum(tmp))/4;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
        Ml = (U(uj,ui+1)+U(uj,ui-1))/2;
        tmp = [U(uj-1,ui+1) U(uj+1,ui) U(uj-1,ui) U(uj+1,ui+1)];
        Mo = (sum(tmp))/4;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
    else
        Ml = (U(uj-1,ui)+U(uj+1,ui-1))/2;
        tmp = [U(uj-1,ui-1) U(uj+1,ui) U(uj,ui+1) U(uj,ui-1)];
        Mo = (sum(tmp))/4;
        Ro = max(tmp)-min(tmp);
        bdvCandidate(1) = BDVCalc(Ml,Mo,Ro);
        Ml = (U(uj-1,ui-1)+U(uj+1,ui))/2;
        tmp = [U(uj-1,ui) U(uj+1,ui-1) U(uj,ui+1) U(uj,ui-1)];
        Mo = (sum(tmp))/4;
```

```
                Ro = max(tmp)-min(tmp);
                bdvCandidate(2) = BDVCalc(Ml,Mo,Ro);
                Ml = (U(uj,ui+1)+U(uj,ui-1))/2;
                tmp = [U(uj-1,ui-1) U(uj+1,ui) U(uj-1,ui) U(uj+1,ui-1)];
                Mo = (sum(tmp))/4;
                Ro = max(tmp)-min(tmp);
                bdvCandidate(3) = BDVCalc(Ml,Mo,Ro);
            end
        end
        bdvMatrix(j,i) = max(bdvCandidate);
    end
end
ret = bdvMatrix;
```

- BDVCalc.m

```
function bdv = BDVCalc(Ml,Mo,Ro)
bdv = (Ml-Mo)/Ro;
```

- BDVMatrixcalcDiff.m

```
function bdvMatrix = BDVMatrixcalcDiff(U)
% Calculate BDV matrix
% Input arguments: U-matrix
% Output arguments: BDV matrix

global mapSize

y = mapSize(1);
x = mapSize(2);

bdvMatrix = zeros(mapSize(1),mapSize(2));

for j = 1:mapSize(1)
    uj = 2*j-1;
    for i=1:mapSize(2)
        ui = 2*i-1;
        bdvCandidate = zeros(3,1);
        if j==1 && i==1    % Top left corner
            Ml = U(uj,ui+1);
            Mo = U(uj+1,ui);
            bdvCandidate(1) = Ml-Mo;
            Ml = U(uj+1,ui);
            Mo = U(uj,ui+1);
            bdvCandidate(2) = Ml-Mo;
        elseif j==1 && i>1 && i<mapSize(2)     % Upper edge
            Ml = (U(uj,ui-1)+U(uj,ui+1))/2;
            Mo = (U(uj+1,ui)+U(uj+1,ui-1))/2;
            bdvCandidate(1) = Ml-Mo;
            Ml = U(uj+1,ui);
            Mo = (U(uj,ui-1) +U(uj+1,ui-1) + U(uj,ui+1))/3;
            bdvCandidate(2) = Ml-Mo;
            Ml = U(uj+1,ui-1);
            Mo = (U(uj+1,ui) +U(uj,ui-1)+ U(uj,ui+1))/3;
            bdvCandidate(3) = Ml-Mo;
```

```
        elseif j==1 && i==mapSize(2)      % Top right corner
            Ml = U(uj,ui-1);
            Mo = (U(uj+1,ui)+ U(uj+1,ui-1))/2;
            bdvCandidate(1) = Ml-Mo;
            Ml = U(uj+1,ui);
            Mo = (U(uj,ui-1)+ U(uj+1,ui-1))/2;
            bdvCandidate(2) = Ml-Mo;
            Ml = U(uj+1,ui-1);
            Mo = (U(uj+1,ui) +U(uj,ui-1))/2;
            bdvCandidate(3) = Ml-Mo;
        elseif j>1 && j<mapSize(1) && i==mapSize(2)    % Right edge
            if rem(j,2)==0
                Ml = U(uj,ui-1);
                Mo = (U(uj+1,ui)+ U(uj-1,ui))/2;
                bdvCandidate(1) = Ml-Mo;
                Ml = U(uj+1,ui);
                Mo = (U(uj,ui-1)+ U(uj-1,ui))/2;
                bdvCandidate(2) = Ml-Mo;
                Ml = U(uj-1,ui);
                Mo = (U(uj+1,ui)+ U(uj,ui-1))/2;
                bdvCandidate(3) = Ml-Mo;
            else
                Ml = (U(uj-1,ui)+U(uj+1,ui-1))/2;
                Mo = (U(uj-1,ui-1)+ U(uj+1,ui) +U(uj,ui-1))/3;
                bdvCandidate(1) = Ml-Mo;
                Ml = (U(uj-1,ui-1)+U(uj+1,ui))/2;
                Mo = (U(uj-1,ui) +U(uj+1,ui-1)+ U(uj,ui-1))/3;
                bdvCandidate(2) = Ml-Mo;
                Ml = U(uj,ui-1);
                Mo = (U(uj-1,ui-1) +U(uj+1,ui) +U(uj-1,ui) +U(uj+1,ui-
1))/4;
                bdvCandidate(3) = Ml-Mo;
            end
        elseif j==mapSize(1) && i==mapSize(2)     % Bottom right corner
            Ml = U(uj,ui-1);
            Mo = (U(uj-1,ui)+ U(uj-1,ui-1))/2;
            bdvCandidate(1) = Ml-Mo;
            Ml = U(uj-1,ui);
            Mo = (U(uj,ui-1) +U(uj-1,ui-1))/2;
            bdvCandidate(2) = Ml-Mo;
            Ml = U(uj-1,ui-1);
            Mo = (U(uj-1,ui)+ U(uj,ui-1))/2;
            bdvCandidate(3) = Ml-Mo;
        elseif j==mapSize(1) && i>1 && i<mapSize(2)     % Lower edge
            Ml = (U(uj,ui-1)+U(uj,ui+1))/2;
            Mo = (U(uj-1,ui)+ U(uj-1,ui-1))/2;
            bdvCandidate(1) = Ml-Mo;
            Ml = U(uj-1,ui);
            Mo = (U(uj,ui-1) +U(uj-1,ui-1) +U(uj,ui+1))/3;
            bdvCandidate(2) = Ml-Mo;
            Ml = U(uj-1,ui-1);
            Mo = (U(uj-1,ui)+ U(uj,ui-1) +U(uj,ui+1))/3;
            bdvCandidate(3) = Ml-Mo;
        elseif j==mapSize(1) && i==1     % Bottom left corner
            Ml = U(uj,ui+1);
            Mo = U(uj-1,ui);
            bdvCandidate(1) = Ml-Mo;
            Ml = U(uj-1,ui);
            Mo = U(uj,ui+1);
            bdvCandidate(2) = Ml-Mo;
        elseif j>1 && j<mapSize(1) && i==1    % Left edge
```

```
            if rem(j,2)==0
                Ml = (U(uj-1,ui)+U(uj+1,ui+1))/2;
                Mo = (U(uj-1,ui+1)+ U(uj+1,ui)+ U(uj,ui+1))/3;
                bdvCandidate(1) = Ml-Mo;
                Ml = (U(uj-1,ui+1)+U(uj+1,ui))/2;
                Mo = (U(uj-1,ui)+ U(uj+1,ui+1) +U(uj,ui+1))/3;
                bdvCandidate(2) = Ml-Mo;
                Ml = U(uj,ui+1);
                Mo = (U(uj-1,ui+1)+ U(uj+1,ui) +U(uj-1,ui)
+U(uj+1,ui+1))/4;
                bdvCandidate(3) = Ml-Mo;
            else
                Ml = U(uj,ui+1);
                Mo = (U(uj+1,ui)+ U(uj-1,ui))/2;
                bdvCandidate(1) = Ml-Mo;
                Ml = U(uj+1,ui);
                Mo = (U(uj,ui+1)+ U(uj-1,ui))/2;
                bdvCandidate(2) = Ml-Mo;
                Ml = U(uj-1,ui);
                Mo = (U(uj+1,ui) +U(uj,ui+1))/2;
                bdvCandidate(3) = Ml-Mo;
            end
        else                    % Middle part
            if rem(j,2)==0
                Ml = (U(uj-1,ui)+U(uj+1,ui+1))/2;
                Mo = (U(uj-1,ui+1) +U(uj+1,ui) +U(uj,ui+1) +U(uj,ui-
1))/4;
                bdvCandidate(1) = Ml-Mo;
                Ml = (U(uj-1,ui+1)+U(uj+1,ui))/2;
                Mo = (U(uj-1,ui)+ U(uj+1,ui+1) +U(uj,ui+1)+ U(uj,ui-
1))/4;
                bdvCandidate(2) = Ml-Mo;
                Ml = (U(uj,ui+1)+U(uj,ui-1))/2;
                Mo = (U(uj-1,ui+1) +U(uj+1,ui) +U(uj-1,ui)+
U(uj+1,ui+1))/4;
                bdvCandidate(3) = Ml-Mo;
            else
                Ml = (U(uj-1,ui)+U(uj+1,ui-1))/2;
                Mo = (U(uj-1,ui-1)+ U(uj+1,ui)+ U(uj,ui+1) +U(uj,ui-
1))/4;
                bdvCandidate(1) = Ml-Mo;
                Ml = (U(uj-1,ui-1)+U(uj+1,ui))/2;
                Mo = (U(uj-1,ui)+ U(uj+1,ui-1) +U(uj,ui+1)+ U(uj,ui-
1))/4;
                bdvCandidate(2) = Ml-Mo;
                Ml = (U(uj,ui+1)+U(uj,ui-1))/2;
                Mo = (U(uj-1,ui-1)+ U(uj+1,ui) +U(uj-1,ui) +U(uj+1,ui-
1))/4;
                bdvCandidate(3) = Ml-Mo;
            end
        end
        bdvMatrix(j,i) = max(bdvCandidate);
    end
end
ret = bdvMatrix;
```

- FindGeneralBoundaries.m

```
function [ret1,ret2] = FindGeneralBoundaries(generalBDV,minIndex)
% Find boundary
% Input arguments: BDV matrix
%                  index of minimum value on BDV matrix
% Output arguments: boundary
%                   binary variable, 1 means there is no boundary

startNeuron = FindStart(minIndex,generalBDV);

if startNeuron(1)==0; % Finished selecting neurons from the edge of the
map
    ret1 = 0;
    ret2 = 1;
    return;
else
    [ret1,ret2] = BoundariesCreation(startNeuron,generalBDV);
end
```

- FindStart.m

```
function startNeuron = FindStart(min_ind,generalBDV)
% Find biggest value that has not been selected yet
% Input arguments: index of minimum value on BDV matrix
%                  BDV matrix
% Output arguments: index of the neuron to begin searching for a boundary

global mapSize alreadySelected

ind = [0 0];
maxValue = generalBDV(min_ind(1),min_ind(2));
for i = 2:mapSize(2)
    if generalBDV(1,i)>maxValue && alreadySelected(1,i)==0
        ind = [1 i];
        maxValue = generalBDV(1,i);
    end
end
for j = 2:mapSize(1)
    if generalBDV(j,mapSize(2))>maxValue &&
alreadySelected(j,mapSize(2))==0
        ind = [j mapSize(2)];
        maxValue = generalBDV(j,mapSize(2));
    end
end
for i = 2:(mapSize(2)-1)
    if generalBDV(mapSize(1),i)>maxValue &&
alreadySelected(mapSize(1),i)==0
        ind = [mapSize(1) i];
        maxValue = generalBDV(mapSize(1),i);
    end
end
for j = 2:(mapSize(1)-1)
    if generalBDV(j,1)>maxValue && alreadySelected(j,1)==0
        ind = [j 1];
        maxValue = generalBDV(j,1);
```

```
    end
end
startNeuron = ind;
```

- BoundariesCreation.m

```
function [newBoundary,noBoundary] =
BoundariesCreation(startNeuron,currentBDV)
% Create boundary
% Input arguments: index of the neuron to begin searching for a boundary
%                  BDV matrix
% Output arguments: boundary
%                   binary variable, 1 means there is no boundary

global neighborUnits bdvThreshold mapSize

% Mark neighboring units of the startNeuron on the edge as zeros
% so that the boundary does not stuck on the edge of the map

if startNeuron(1)==1 && startNeuron(2)~=mapSize(2)
    currentBDV(startNeuron(1),startNeuron(2)-1) = 0;
    currentBDV(startNeuron(1),startNeuron(2)+1) = 0;
elseif startNeuron(1)==1 && startNeuron(2)==mapSize(2)
    currentBDV(startNeuron(1),startNeuron(2)-1) = 0;
    currentBDV(startNeuron(1)+1,startNeuron(2)) = 0;
elseif startNeuron(1)==mapSize(1) && startNeuron(2)~=mapSize(2)
    currentBDV(startNeuron(1),startNeuron(2)-1) = 0;
    currentBDV(startNeuron(1),startNeuron(2)+1) = 0;
elseif startNeuron(1)==mapSize(1) && startNeuron(2)==mapSize(2)
    currentBDV(startNeuron(1),startNeuron(2)-1) = 0;
    currentBDV(startNeuron(1)-1,startNeuron(2)) = 0;
elseif startNeuron(2)==1
    currentBDV(startNeuron(1)-1,startNeuron(2)) = 0;
    currentBDV(startNeuron(1)+1,startNeuron(2)) = 0;
elseif startNeuron(2)==mapSize(2) && startNeuron(1)~=1 &&
startNeuron(1)~=mapSize(1)
    currentBDV(startNeuron(1)-1,startNeuron(2)) = 0;
    currentBDV(startNeuron(1)+1,startNeuron(2)) = 0;
end

noBoundary = 0;
newBoundary = zeros(mapSize(1),mapSize(2));

newBoundary(startNeuron(1),startNeuron(2)) = 1;
J = startNeuron(1);
I = startNeuron(2);

% Search for boundary
while newBoundary(J,I)==1;
    % Find next neuron on the boundary
    temp1 = find(neighborUnits(:,(J+mapSize(1)*(I-1))));
    ind = [mod(temp1,mapSize(1)) ceil(temp1/mapSize(1))];
    ind(find(ind==0)) = mapSize(1);
    currentNeighbors = [];
    % Fing neighbor with the highest BDV value
    for i = 1:length(ind)
        currentNeighbors(i,:) = [ind(i,1) ind(i,2)
currentBDV(ind(i,1),ind(i,2))];
```

```
    end
    [dummy1,temp2] = max(currentNeighbors);
    candidate = currentNeighbors(temp2(3),:);
    % If the BDV value is too high try next candidate
    while currentBDV(J,I)/candidate(3)<(1/bdvThreshold)
        currentNeighbors(temp2(3),:) = 0;
        [dummy1,temp2] = max(currentNeighbors);
        candidate = currentNeighbors(temp2(3),:);
    end

    % If it is already part of the boundary try next candidate
    while candidate(1)~=0 && newBoundary(candidate(1),candidate(2))~=0
        currentNeighbors(temp2(3),:) = 0;
        [dummy1,temp2] = max(currentNeighbors);
        candidate = currentNeighbors(temp2(3),:);
    end

    if candidate(3)~=0 && currentBDV(J,I)/candidate(3)<bdvThreshold
        % If the BDV value is not too low then candidate is part
        % of the boundary
        newBoundary(candidate(1),candidate(2)) = 1;
    else % A boundary can not be formed
        noBoundary = 1;
        break;
    end

    J = candidate(1);
    I = candidate(2);

    % If the boundary has reaches an edge of the map return
    % Or else search for next neuron
    if J==1 || J==mapSize(1) || I==1 || I==mapSize(2)
        break;
    end
end
```

- FindComponentBoundary.m

```
function [componentBoundary,matchPercentage] =
FindComponentBoundary(generalBoundary,BDVTest)
% Find boundary on component matrices
% Input arguments: general boundary
%                  BDV matrix
% Output arguments: component boundary
%                   percentage of similarity between general and
component boundary

global neighborUnits mapSize olaTaCmpbnd %currentVariable

componentBoundary = zeros(mapSize(1),mapSize(2));
compBoundaries = zeros(mapSize(1),mapSize(2),3);
compparted = zeros(mapSize(1),mapSize(2));
successMatrix = zeros(3,1);
noBoundary = 0;

% Find the edges of the general boundary
twoEdges = [];
for i = 1:mapSize(2)
```

```
    if generalBoundary(1,i)==1
        twoEdges = [twoEdges; 1 i];
        found = 1;
    end
end
for j = 2:mapSize(1)
    if generalBoundary(j,mapSize(2))==1
        twoEdges = [twoEdges; j mapSize(2)];
        found = 1;
    end
end
for i = 1:(mapSize(2)-1)
    if generalBoundary(mapSize(1),i)==1
        twoEdges = [twoEdges; mapSize(1) i];
        found = 1;
    end
end
for j = 2:(mapSize(1)-1)
    if generalBoundary(j,1)==1
        twoEdges = [twoEdges; j 1];
        found = 1;
    end
end

% Begin from the first edge
edgeIndex = twoEdges(1,:);
componentBoundary(edgeIndex(1),edgeIndex(2)) = 1;
[componentBoundary,noBoundary] =
ComponentBoundaryCreation(edgeIndex,componentBoundary,compparted,BDVTest)
;
compparted = compparted+componentBoundary;
boundarySize = find(componentBoundary);
while length(boundarySize)<2 || (length(boundarySize)==2 &&
(abs(boundarySize(1)-boundarySize(2))==1 || abs(boundarySize(1)-
boundarySize(2))==mapSize(1)))
    componentBoundary = zeros(mapSize(1),mapSize(2));
    componentBoundary(edgeIndex(1),edgeIndex(2)) = 1;
    [componentBoundary,noBoundary] =
ComponentBoundaryCreation(edgeIndex,componentBoundary,compparted,BDVTest)
;
    if noBoundary==1
        break;
    end
    compparted = compparted+componentBoundary;
    boundarySize = find(componentBoundary);
end
noBoundary = 0;
compBoundaries(:,:,1) = componentBoundary;
% Compare boundaries
successMatrix(1) =
MatchBoundaries(generalBoundary,compBoundaries(:,:,1));

% Begin from the other edge
componentBoundary = zeros(mapSize(1),mapSize(2));
compparted = zeros(mapSize(1),mapSize(2));
edgeIndex = twoEdges(2,:);
componentBoundary(edgeIndex(1),edgeIndex(2)) = 1;
[componentBoundary,noBoundary] =
ComponentBoundaryCreation(edgeIndex,componentBoundary,compparted,BDVTest)
;
compparted = compparted+componentBoundary;
```

```
boundarySize = find(componentBoundary);
while length(boundarySize)<2 || (length(boundarySize)==2 &&
(abs(boundarySize(1)-boundarySize(2))==1 || abs(boundarySize(1)-
boundarySize(2))==mapSize(1)))
    componentBoundary = zeros(mapSize(1),mapSize(2));
    componentBoundary(edgeIndex(1),edgeIndex(2)) = 1;
    [componentBoundary,noBoundary] =
ComponentBoundaryCreation(edgeIndex,componentBoundary,compparted,BDVTest)
;
    if noBoundary==1
        break;
    end
    compparted = compparted+componentBoundary;
    boundarySize = find(componentBoundary);
end
compBoundaries(:,:,2) = componentBoundary;
% Compare boundaries
successMatrix(2) =
MatchBoundaries(generalBoundary,compBoundaries(:,:,2));

% Check if there is a perfect match
if successMatrix(2)==100
    componentBoundary = compBoundaries(:,:,2);
elseif successMatrix(1)==100
    componentBoundary = compBoundaries(:,:,1);
else
    % Merge the two previous boundaries
    mergedBoundary =
compBoundaries(:,:,1)+compBoundaries(:,:,2)+compBoundaries(:,:,2);
    componentBoundary = zeros(mapSize(1),mapSize(2));
    noNeighborBoundary = 0;
    edgeIndex = twoEdges(1,:);
    J = edgeIndex(1);
    I = edgeIndex(2);
    componentBoundary(J,I) = 1;
    mergedBoundary(J,I) = 0;
    while componentBoundary(J,I)==1 && noNeighborBoundary==0
        % Follow first boundary
        noNeighborBoundary = 1;
        temp = find(neighborUnits(:,(J+mapSize(1)*(I-1))));
        ind = [mod(temp,mapSize(1)) ceil(temp/mapSize(1))];
        ind(find(ind==0)) = mapSize(1);
        for i = 1:length(ind)
            if mergedBoundary(ind(i,1),ind(i,2))==3 &&
generalBoundary(ind(i,1),ind(i,2))==1
                % Follow the common boundary
                componentBoundary(ind(i,1),ind(i,2)) = 1;
                J = ind(i,1);
                I = ind(i,2);
                mergedBoundary(ind(i,1),ind(i,2)) = 0;
                noNeighborBoundary = 0;
                break;
            end
        end
        if noNeighborBoundary==1
            for i = 1:length(ind)
                if mergedBoundary(ind(i,1),ind(i,2))==2 &&
generalBoundary(ind(i,1),ind(i,2))==1
                    %  Else follow the second boundary if there is also
the
                    %  general boundary
```

```
                componentBoundary(ind(i,1),ind(i,2)) = 1;
                J = ind(i,1);
                I = ind(i,2);
                mergedBoundary(ind(i,1),ind(i,2)) = 0;
                noNeighborBoundary = 0;
                break;
            end
        end
    end
    if noNeighborBoundary==1
        for i = 1:length(ind)
            if mergedBoundary(ind(i,1),ind(i,2))==1 &&
generalBoundary(ind(i,1),ind(i,2))==1
                % Else follow the first boundary if there is also the
                % general boundary
                componentBoundary(ind(i,1),ind(i,2)) = 1;
                J = ind(i,1);
                I = ind(i,2);
                mergedBoundary(ind(i,1),ind(i,2)) = 0;
                noNeighborBoundary = 0;
                break;
            end
        end
    end
    if noNeighborBoundary==1
        for i = 1:length(ind)
            if mergedBoundary(ind(i,1),ind(i,2))==2
                % Else follow the second boundary
                componentBoundary(ind(i,1),ind(i,2)) = 1;
                J = ind(i,1);
                I = ind(i,2);
                mergedBoundary(ind(i,1),ind(i,2)) = 0;
                noNeighborBoundary = 0;
                break;
            end
        end
    end
end
compBoundaries(:,:,3) = componentBoundary;
% Compare boundaries
successMatrix(3) =
MatchBoundaries(generalBoundary,compBoundaries(:,:,3));
end

% Find the boundary that is the most similar to the general boundary
[matchPercentage winnerIndex] = max(successMatrix);
componentBoundary = compBoundaries(:,:,winnerIndex);
```

- ComponentBoundaryCreation.m

```
function [newCompBoundary,noBoundary] =
ComponentBoundaryCreation(edgeIndex,newCompBoundary,compparted,bdvTest)
% Create boundary starting from a given neuron
% Input arguments: index of the neuron to begin searching for a boundary
%                  component boundary
%                  already visited neurons
%                  BDV matrix
% Output arguments: new component boundary
```

```
%                   binary variable, 1 means there is no boundary

global neighborUnits mapSize bdvThreshold

J = edgeIndex(1);
I = edgeIndex(2);

noBoundary = 0;
while newCompBoundary(J,I)==1;
    % Find next neuron on the boundary
    temp1 = find(neighborUnits(:,(J+mapSize(1)*(I-1))));
    ind = [mod(temp1,mapSize(1)) ceil(temp1/mapSize(1))];
    ind(find(ind==0)) = mapSize(1);
    currentNeighbors = [];
    % Fing neighbor with the highest BDV value
    for i = 1:length(ind)
        currentNeighbors(i,:) = [ind(i,1) ind(i,2)
bdvTest(ind(i,1),ind(i,2))];
    end
    [dummy1,temp2] = max(currentNeighbors);
    candidate = currentNeighbors(temp2(3),:);
    % If the BDV value is too high try next candidate
    while (bdvTest(J,I)/candidate(3))<(1/bdvThreshold)
        currentNeighbors(temp2(3),:) = 0;
        [dummy1,temp2] = max(currentNeighbors);
        candidate = currentNeighbors(temp2(3),:);
    end

    % If it is already part of the boundary try next candidate
    while candidate(1)~=0 &&
(newCompBoundary(candidate(1),candidate(2))~=0 ||
compparted(candidate(1),candidate(2))~=0)
        currentNeighbors(temp2(3),:) = 0;
        [dummy1,temp2] = max(currentNeighbors);
        candidate = currentNeighbors(temp2(3),:);
    end

    if candidate(3)~=0 && bdvTest(J,I)/candidate(3)<bdvThreshold
        % If the BDV value is not too low then candidate is part
        % of the boundary
        newCompBoundary(candidate(1),candidate(2)) = 1;
    else % A boundary can not be formed
        noBoundary = 1;
        break;
    end

    J = candidate(1);
    I = candidate(2);

    % If the boundary has reaches an edge of the map return
    % Or else search for next neuron
    if J==1 || J==mapSize(1) || I==1 || I==mapSize(2)
        break;
    end
end
```

- MatchBoundaries.m

```
function success = MatchBoundaries(matrix1,matrix2)
% Match two boundaries
% Input arguments: boundary 1
%                  boundary 1
% Output arguments: percentage of similarity between general and
component boundary

hits = 0;
checks = 0;

num1 = length(find(matrix1));
num2 = length(find(matrix2));
% Calculate match based on the largest boundary
if num1<num2
    tmp = matrix1;
    matrix1 = matrix2;
    matrix2 = tmp;
end
for j = 1:size(matrix1,1)
    for i = 1:size(matrix1,2)
        if matrix1(j,i)==1
            if matrix2(j,i)==1
                hits = hits+1;
                checks = checks+1;
            else
                checks = checks+1;
            end
        end
    end
end
success = hits*100/checks;
```

- BoundaryThrehold.m

```
function ruleThreshold =
BoundaryThrehold(componentBoundary,currentVariableMatrix)
% Calculate rule threshold
% Input arguments: component boundary
%                  component matrix
% Output arguments: rule threshold

global mapSize neighborUnits

a = [];
next = [0 0];
visitedUnit = zeros(mapSize(1),mapSize(2));

% Find an edge of the boundary
ind = [0 0];
found = 0;
for i = 1:mapSize(2)
    if componentBoundary(1,i)==1
        ind = [1 i];
        found = 1;
    end
end
```

```
for j = 2:mapSize(1)
    if found==0 && componentBoundary(j,mapSize(2))==1
        ind = [j mapSize(2)];
        found = 1;
    end
end
for i = 1:(mapSize(2)-1)
    if found==0 && componentBoundary(mapSize(1),i)==1
        ind = [mapSize(1) i];
        found = 1;
    end
end
for j = 2:(mapSize(1)-1)
    if found==0 && componentBoundary(j,1)==1
        ind = [j 1];
        found = 1;
    end
end
next = ind;

% Visit every neuron on the boundary and calculate mean value
while next~=0
    visitedUnit(next(1),next(2)) = 1;
    temp = find(neighborUnits(:,(next(1)+mapSize(1)*(next(2)-1))));
    ind = [mod(temp,mapSize(1)) ceil(temp/mapSize(1))];
    ind(find(ind==0)) = mapSize(1);
    a = [a currentVariableMatrix(next(1),next(2))];
    next = [0 0];
    for i = 1:length(ind)
        if componentBoundary(ind(i,1),ind(i,2))==1 &&
visitedUnit(ind(i,1),ind(i,2))==0
            next = [ind(i,1) ind(i,2)];
            break;
        end
    end
end
ruleThreshold = mean(a);
```

- RuleSections.m

```
function [section1,section2,ruleOperands] =
RuleSections(componentBoundary,currentVariableMatrix,ruleThres)
% Find sections created by the boundary
% Input arguments: component boundary
%                  component matrix
%                  rule threshold
% Output arguments: section1
%                   section2
%                   rule operand for each section
global mapSize

componentBoundary = componentBoundary;
section = 2;


% Mark the two differnt sections with odd and even numbers
for i = 1:mapSize(2)
    j = 1;
```

```
    if componentBoundary(1,i)==1
        section = section+1;
    else
        while j<=mapSize(1) && componentBoundary(j,i)~=1
            componentBoundary(j,i) = section;
            j = j+1;
        end
    end
end
for j = 2:mapSize(1)
    i = mapSize(2);
    if componentBoundary(j,mapSize(2))==1
        section = section+1;
    else
        while i>=1 && componentBoundary(j,i)~=1
            componentBoundary(j,i) = section;
            i = i-1;
        end
    end
end
for i = (mapSize(2)-1):-1:1
    j = mapSize(1);
    if componentBoundary(mapSize(1),i)==1
        section = section+1;
    else
        while j>=1 && componentBoundary(j,i)~=1
            componentBoundary(j,i) = section;
            j = j-1;
        end
    end
end
for j = (mapSize(1)-1):-1:2
    i = 1;
    if componentBoundary(j,1)==1
        section = section+1;
    else
        while i<=mapSize(2) && componentBoundary(j,i)~=1
            componentBoundary(j,i) = section;
            i = i+1;
        end
    end
end

% Find neurons that belong to each section
section1 = zeros(mapSize(1),mapSize(2));
section2 = zeros(mapSize(1),mapSize(2));
for j = 1:mapSize(1)
    for i = 1:mapSize(2)
        if componentBoundary(j,i)==1
        elseif mod(componentBoundary(j,i),2)==1
            section1(j,i) = 1;
        elseif mod(componentBoundary(j,i),2)==0
            section2(j,i) = 1;
        end
    end
end

% Calculate mean value of neurons for each section
section1VariableMatrix = section1.*currentVariableMatrix;
section2VariableMatrix = section2.*currentVariableMatrix;
meanSection1 = 0;
```

```
meanSection2 = 0;
count1 = 0;
count2 = 0;

for j = 1:mapSize(1)
    for i = 1:mapSize(2)
        if section1VariableMatrix(j,i)~=0
            meanSection1 = meanSection1+section1VariableMatrix(j,i);
            count1 = count1+1;
        end
        if section2VariableMatrix(j,i)~=0
            meanSection2 = meanSection2+section2VariableMatrix(j,i);
            count2 = count2+1;
        end
    end
end
meanSection1 = meanSection1/count1;
meanSection2 = meanSection2/count2;

% Find rule operand for the two sections
if meanSection1>ruleThres
    ruleOperands(1) = 2;
else
    ruleOperands(1) = 1;
end
if meanSection2>ruleThres
    ruleOperands(2) = 2;
else
    ruleOperands(2) = 1;
end
```

- IrisRuleCreation.m (Οι συναρτήσεις για άλλα σύνολα δεδομένων είναι απλές μετατροπές του παρακάτω αρχείου)

```
function [ruleCell,alternateRules] =
IrisRuleCreation(section1,section2,ruleThres,ruleOperands)
% Rule Creation
% Input arguments: section1
%                  section2
%                  rule threshold
%                   rule operand for each section
% Output arguments: cell with rules for each data class
%                   cell with alternate rules for each data class

global numClasses classLabels mapSize labelThreshold currentVariable
ruleCell alternateRules

% Count labels on the SOM and on every section
numlabels = zeros(numClasses,1);
numSection1 = zeros(numClasses,1);
numSection2 = zeros(numClasses,1);

for j = 1:mapSize(1) %(Dataset specific)
    for i = 1:mapSize(2)
        if strcmp(classLabels(j,i),'1')
                numlabels(1) = numlabels(1)+1;
        elseif strcmp(classLabels(j,i),'2')
                numlabels(2) = numlabels(2)+1;
```

```
        elseif strcmp(classLabels(j,i),'3')
                numlabels(3) = numlabels(3)+1;
        end
        if section1(j,i)==1
            if strcmp(classLabels(j,i),'1')
                   numSection1(1) = numSection1(1)+1;
            elseif strcmp(classLabels(j,i),'2')
                   numSection1(2) = numSection1(2)+1;
            elseif strcmp(classLabels(j,i),'3')
                   numSection1(3) = numSection1(3)+1;
            end
        end
        if section2(j,i)==1
            if strcmp(classLabels(j,i),'1')
                   numSection2(1) = numSection2(1)+1;
            elseif strcmp(classLabels(j,i),'2')
                   numSection2(2) = numSection2(2)+1;
            elseif strcmp(classLabels(j,i),'3')
                   numSection2(3) = numSection2(3)+1;
            end
        end
    end
end

for i = 1:numClasses
    check1 = [];
    check2 = [];
    check3 = [];
    check4 = [];
    % Create rule when the majority of a class labels belongs to one
    % section
    if numSection1(i)/numlabels(i)>labelThreshold
        ruleCell{i} = [ruleCell{i};currentVariable ruleOperands(1)
ruleThres 0 0 0];
    end
    if numSection2(i)/numlabels(i)>labelThreshold
        ruleCell{i} = [ruleCell{i};currentVariable ruleOperands(2)
ruleThres 0 0 0];
    end
    % Add rule in alternateRule only if it the first time
    if ~isempty(alternateRules{i})
        check1 = find(alternateRules{i}(:,3)==ruleThres);
        check2 = find(alternateRules{i}(:,1)==currentVariable);
        check3 = find(alternateRules{i}(:,2)==ruleOperands(1));
        check4 = find(alternateRules{i}(:,2)==ruleOperands(2));
    end
    % Create rule when a class labels dominate one section
    if numSection1(i)>(0.8*sum(numSection1)) && ~isnan(ruleThres) &&
(isempty(check1) || isempty(check2) || isempty(check3))
        alternateRules{i} = [alternateRules{i};currentVariable
ruleOperands(1) ruleThres 0 0 0];
    end
    if numSection2(i)>(0.8*sum(numSection2)) && ~isnan(ruleThres) &&
(isempty(check1) || isempty(check2) || isempty(check4))
        alternateRules{i} = [alternateRules{i};currentVariable
ruleOperands(2) ruleThres 0 0 0];
    end
end
```

- RulePostProccessing1.m

```
% Rule post processing (stage 1)
% Evaluation and selection of best rules

% Count how many tuplres are correctly classified for each rule and how
% many are badly classified
for i = 1:numClasses
    for k = 1:length(DataIn)
        for j = 1:size(ruleCell{i},1)
            if ruleCell{i}(j,2)==1
                if DataIn(k,ruleCell{i}(j,1))<ruleCell{i}(j,3)
                    if OutClass(k)==i
                        ruleCell{i}(j,4) = ruleCell{i}(j,4)+1;
                    else
                        ruleCell{i}(j,5) = ruleCell{i}(j,5)+1;
                    end
                end
            elseif ruleCell{i}(j,2)==2
                if DataIn(k,ruleCell{i}(j,1))>ruleCell{i}(j,3)
                    if OutClass(k)==i
                        ruleCell{i}(j,4) = ruleCell{i}(j,4)+1;
                    else
                        ruleCell{i}(j,5) = ruleCell{i}(j,5)+1;
                    end
                end
            end
            if k==length(DataIn)
                % Calculate rule significance
                ruleCell{i}(j,6) =
(ruleCell{i}(j,4)*ruleCell{i}(j,4))/((ruleCell{i}(j,5)+ruleCell{i}(j,4))*
(instancesPerClass(i)));
            end
        end
        for j = 1:size(alternateRules{i},1)
            if alternateRules{i}(j,2)==1
                if
DataIn(k,alternateRules{i}(j,1))<alternateRules{i}(j,3)
                    if OutClass(k)==i
                        alternateRules{i}(j,4) =
alternateRules{i}(j,4)+1;
                    else
                        alternateRules{i}(j,5) =
alternateRules{i}(j,5)+1;
                    end
                end
            elseif alternateRules{i}(j,2)==2
                if
DataIn(k,alternateRules{i}(j,1))>alternateRules{i}(j,3)
                    if OutClass(k)==i
                        alternateRules{i}(j,4) =
alternateRules{i}(j,4)+1;
                    else
                        alternateRules{i}(j,5) =
alternateRules{i}(j,5)+1;
                    end
                end
            end
            if k==length(DataIn)
                % Calculate rule significance
```

```
                alternateRules{i}(j,6) =
(alternateRules{i}(j,4)*alternateRules{i}(j,4))/((alternateRules{i}(j,5)+
alternateRules{i}(j,4))*(instancesPerClass(i)));
            end
        end
    end
end

for i = 1:numClasses
    for j = 1:size(ruleCell{i},1)
        % If the rule does not classify half of the class take opposite
        % rule unless correct tuples are more than badly classified
tuples
        if ruleCell{i}(j,4)<(instancesPerClass(i)/2) &&
ruleCell{i}(j,4)<ruleCell{i}(j,5)
            if ruleCell{i}(j,2)==1
                ruleCell{i}(j,2) = 2;
            else
                ruleCell{i}(j,2) = 1;
            end
            ruleCell{i}(j,4) = instancesPerClass(i)-ruleCell{i}(j,4);
            ruleCell{i}(j,5) = length(DataIn)-instancesPerClass(i)-
ruleCell{i}(j,5);
            ruleCell{i}(j,6) =
(ruleCell{i}(j,4)*ruleCell{i}(j,4))/((ruleCell{i}(j,5)+ruleCell{i}(j,4))*
(instancesPerClass(i)));
        end
        %  Discover best rules for every class and every variable
        if (ruleCell{i}(j,4)+ruleCell{i}(j,5))==length(DataIn)
        elseif entered(i,ruleCell{i}(j,1),ruleCell{i}(j,2))
            for k = 1:size(bestRules{i},1)
                if bestRules{i}(k,1)==ruleCell{i}(j,1) &&
bestRules{i}(k,2)==ruleCell{i}(j,2)
                    if bestRules{i}(k,4)<ruleCell{i}(j,4) ||
(bestRules{i}(k,4)==ruleCell{i}(j,4) &&
bestRules{i}(k,5)>ruleCell{i}(j,5))
                        secondaryRules{i} =
[secondaryRules{i};bestRules{i}(k,:)];
                        bestRules{i}(k,:) = ruleCell{i}(j,:);
                    elseif (bestRules{i}(k,4)-
bestRules{i}(k,5))<=(ruleCell{i}(j,4)-ruleCell{i}(j,5)) ||
bestRules{i}(k,6)<=ruleCell{i}(k,6)
                        secondaryRules{i} =
[secondaryRules{i};ruleCell{i}(j,:)];
                    end
                end
            end
        else
            bestRules{i} = [bestRules{i};ruleCell{i}(j,:)];
            entered(i,ruleCell{i}(j,1),ruleCell{i}(j,2)) = 1;
        end
    end
end


% Sort rules
for i = 1:numClasses
    bestRules{i} = sortrows(bestRules{i},[4 5]);
    if ~isempty(alternateRules{i})
        alternateRules{i} = sortrows(alternateRules{i},[6 4 5]);
    end
```

```
    if ~isempty(secondaryRules{i})
        secondaryRules{i} = sortrows(secondaryRules{i},[6 4 5]);
    end
end
```

- RulePostProccessing2.m

```
% Rule post processing (stage 2)
% Combination of rules
clc;

suc = zeros(numClasses,1);
fail = zeros(numClasses,1);
remainingRules = cell(numClasses,1);
tmpBestRules = bestRules;

for i = 1:numClasses
    perfectDescription = 0;
    tmpRules = [];
    tmpDataIn = DataIn;
    tmpOutClass = OutClass;
    fprintf(fid,sprintf('-------- Klash %d --------------\n',i));
    j = size(tmpBestRules{i},1);
    % Select rules with the same number of correctly classified tuples
    tmp = tmpBestRules{i}(j,4);
    while tmp==tmpBestRules{i}(j,4)
        tmpRules = [tmpRules;tmpBestRules{i}(j,:)];
        tmpBestRules{i}(j,:) = [];
        j = j-1;
    end
    oldSuc = -1;
    oldFail = -1;

    % Apply rules
    for j = size(tmpRules,1):-1:1
        suc(i) = 0;
        fail(i) = 0;
        excludedTuples = [];
        for k = 1:size(tmpDataIn,1)
            if tmpRules(j,2)==1 &&
tmpDataIn(k,tmpRules(j,1))<tmpRules(j,3)
                if tmpOutClass(k)==i
                    suc(i) = suc(i)+1;
                else
                    fail(i) = fail(i)+1;
                end
            elseif tmpRules(j,2)==2 &&
tmpDataIn(k,tmpRules(j,1))>tmpRules(j,3)
                if tmpOutClass(k)==i
                    suc(i) = suc(i)+1;
                else
                    fail(i) = fail(i)+1;
                end
            else
                excludedTuples = [excludedTuples;k];
            end
        end
```

```
        % If there are better results delete the tuples that do not meet
the rule
        if fail(i)~=oldFail
            for k = 1:length(excludedTuples)
                tmpDataIn(excludedTuples(k)-k+1,:) = [];
                tmpOutClass(excludedTuples(k)-k+1) = [];
            end
            fprintf(fid,sprintf('if Var%d %d %d then class=%d   suc=%d
fail=%d
sf=%d\n',tmpRules(j,1),tmpRules(j,2),tmpRules(j,3),i,suc(i),fail(i),tmpRu
les(j,6)));
            oldSuc = suc(i);
            oldFail = fail(i);
        end
        if suc(i)==instancesPerClass(i) && fail(i)==0
            perfectDescription = 1;
            break;
        end
    end
    if ~perfectDescription
    % Apply secondary rules
        fprintf(fid,'Telos ths oloklhrhs perigrafhs\n');
        remainingRules{i} = [secondaryRules{i};tmpBestRules{i}];
        % Sort in acsending coverage
        remainingRules{i} = sortrows(remainingRules{i},[4 6 5]);
        % Sort in acsending significance
%         remainingRules{i} = sortrows(remainingRules{i},[6 4 5]);
        tmpRules = [];
        for j = size(remainingRules{i},1):-1:1
            if ~isnan(remainingRules{i}(j,6))
            % Apply rule
            suc(i) = 0;
            fail(i) = 0;
            excludedTuples = [];
            for k = 1:size(tmpDataIn,1)
                if remainingRules{i}(j,2)==1 &&
tmpDataIn(k,remainingRules{i}(j,1))<remainingRules{i}(j,3)
                    if tmpOutClass(k)==i
                        suc(i) = suc(i)+1;
                    else
                        fail(i) = fail(i)+1;
                    end
                elseif remainingRules{i}(j,2)==2 &&
tmpDataIn(k,remainingRules{i}(j,1))>remainingRules{i}(j,3)
                    if tmpOutClass(k)==i
                        suc(i) = suc(i)+1;
                    else
                        fail(i) = fail(i)+1;
                    end
                else
                    excludedTuples = [excludedTuples;k];
                end
            end
        % If there are better results delete the tuples that do not meet
the rule
            if (oldSuc-suc(i))<(oldFail-fail(i))
                for k = 1:length(excludedTuples)
                    tmpDataIn(excludedTuples(k)-k+1,:) = [];
                    tmpOutClass(excludedTuples(k)-k+1) = [];
                end
```

```
                fprintf(fid,sprintf('if Var%d %d %d then class=%d
suc=%d  fail=%d\t
sf=%3.6f\n',remainingRules{i}(j,1),remainingRules{i}(j,2),remainingRules{
i}(j,3),i,suc(i),fail(i),remainingRules{i}(j,6)));
                oldSuc = suc(i);
                oldFail = fail(i);
            end
            if fail(i)==0
                break;
            end
        end
        end
        % Show alternateRules
        fprintf(fid,'alternate rules\n');
        for j = 1:size(alternateRules{i},1)
            fprintf(fid,'%d %d %d %d %d %d\n',alternateRules{i}(j,:));
        end
    end
end
```

- rulesetEvaluation.m (Πρόγραμμα δοκιμής και αξιολόγησης του συνόλου κανόνων με χρήση του κριτηρίου της ελάχιστης πιθανότητας λάθους, μπορεί να χρησιμοποιηθεί και για τη δεύτερη μεθοδολογία χωρίς αλλαγές)

```
% Evaluation of rule set
clc;

load('iris.mat'); % (dataset specific)
numClasses=3; % (dataset specific)
instancesPerClass=[50 50 50]; % (dataset specific)

suc=zeros(numClasses,1);
fail=zeros(numClasses,1);
finsuc=zeros(numClasses,1);
finfail=zeros(numClasses,1);
ruleS2=zeros(numClasses,1);
falsePositives=zeros(numClasses,1);
dcTuples=[];
dcCell=cell(length(DataIn),1);
classSign=zeros(2);

for k=1:length(DataIn)
    cl=0;
    cor=0; % Correctly classified by the rule
    mis=0; % Badly classified by the rule
    dd=[];

    % Rule for class 1
    if DataIn(k,4)<0.7625179
        cl=cl+1;
        dd=[dd;1];
        if OutClass(k)==1
            suc(1)=suc(1)+1;
            cor=cor+1;
        else
            fail(1)=fail(1)+1;
            mis=mis+1;
        end
```

```
    end
    % Rule for class 4
    if DataIn(k,4)>0.7625179 && DataIn(k,4)<1.601286 &&
DataIn(k,1)<7.044481
        cl=cl+1;
        dd=[dd;2];
        if OutClass(k)==2
            suc(2)=suc(2)+1;
            cor=cor+1;
        else
            fail(2)=fail(2)+1;
            mis=mis+1;
        end
    end
    % Rule for class 3
    if DataIn(k,4)>1.601286 || DataIn(k,1)>7.044481
        cl=cl+1;
        dd=[dd;3];
        if OutClass(k)==3
            suc(3)=suc(3)+1;
            cor=cor+1;
        else
            fail(3)=fail(3)+1;
            mis=mis+1;
        end
    end

    if cl==0 classSign(2,2)=classSign(2,2)+1; % Unclassified
    elseif cl>1 % Classified in more than one class
        dcTuples=[dcTuples;k];
        dcCell{k}=dd;
    elseif (cl==1 && cor==1) classSign(1,1)=classSign(1,1)+1; % Correctly
classified by the rule set
    elseif (cl==1 && mis==1) classSign(1,2)=classSign(1,2)+1; % Badly
classified by the rule set
    end
end

% Calculate false positive rate
for j=1:numClasses
    falsePositives(j)=fail(j)/(length(DataIn)-330);
end

% Classify tuples that belong in more than one class to the class with
the
% lowest false positive rate
for i=1:length(dcTuples)
    dd2=[];
    for j=1:length(dcCell{dcTuples(i)})
        dd2=[dd2;falsePositives(dcCell{dcTuples(i)}(j))];
    end
    [dummy ind]=min(dd2);
    if OutClass(dcTuples(i))==dcCell{dcTuples(i)}(ind)
        classSign(1,1)=classSign(1,1)+1;

finsuc(dcCell{dcTuples(i)}(ind))=finsuc(dcCell{dcTuples(i)}(ind))+1;
    else
        classSign(1,2)=classSign(1,2)+1;

finfail(dcCell{dcTuples(i)}(ind))=finfail(dcCell{dcTuples(i)}(ind))+1;
    end
```

```
end

% Calculate significance measures for each rule
tmp1=suc.*suc;
tmp2=suc+fail;
tmp2=tmp2*instancesPerClass(1);
ruleS2=tmp1./tmp2;

% Calculate significance measures for the rule set
rulesetS2=(classSign(1,1)*classSign(1,1))/((classSign(1,1)+classSign(2,2)
)*length(DataIn));

% Calculate accuracy
accuracy=classSign(1,1)*100/length(DataIn);
```

Παράρτημα Β. Πηγαίος κώδικας μεθόδου 2

- Κεντρικό αρχείο της εφαρμογής για τα δεδομένα Iris. Η μετατροπή του για τα υπόλοιπα δεδομένα γίνεται με απλή αλλαγή των δεδομένων εισόδου, του πλήθους των κατηγοριών και του πλήθους των προτύπων σε κάθε κατηγορία.

```
clear all;
clc;
close all;

% global mapSize numOfVars numClasses maxClusters
startTime = clock;

% Load data
load('iris.mat');

% Label Data
for i = 1:size(DataIn,1)
    s = sprintf('%1.0d',OutClass(i));
    c(i) = cellstr(s);
end
OutClassStr = c';
sD = som_data_struct(DataIn,'labels',OutClassStr,'name','IRIS');
sD = som_normalize(sD,'var'); % Normalize data
clear c OutClassStr

% Make SOM
sM = som_make(sD);
sM = som_autolabel(sM,sD,'vote');

% Specifications
numOfVars = size(DataIn,2);
mapSize = sM.topol.msize;
numClasses = 3; % (Dataset specific)
instancesPerClass = [50 50 50]; % (Dataset specific)
maxClusters = ceil(sqrt(mapSize(1)*mapSize(2)));
method = 'ward'; % Method for hierarchical clustering

% Basic Visualization
som_show(sM,'umat','all','comp',1:numOfVars,'empty','Labels','norm','d');
som_show_add('label',sM,'subplot',numOfVars+2);
figure;som_show(sM,'umat',{'all','U-matrix (total)'},'empty','Labels');
som_show_add('label',sM,'subplot',2);

% Map codebook
sMD = som_denormalize(sM);
mcdD = sMD.codebook; % Denormalized codebook
mcdN = sM.codebook; % Normalized codebook

% Map labels
classLabels = sM.labels;

% Hierarchical clustsering
[numOfClusters,clusteringBase] =
HierarchicalClustering(sM,numClasses,maxClusters,method);

% K-means clustering
% [numOfClusters,clusteringBase] =
KMeansClustering(sM,numClasses,maxClusters);
```

```
% Cluster statistics
run StatisticsComp

% Sig* algorithm
run SigStar2

% Selection of sets of variables
% run VariableSelection

str = 'presiris09';
print('-f3','-dpng',sprintf('./results/%sa',str));
print('-f4','-dpng',sprintf('./results/%sb',str));
print('-f5','-dpng',sprintf('./results/%sc',str));
print('-f6','-dpng',sprintf('./results/%sd',str));
fid = fopen(sprintf('./results/%s.txt',str),'w');

% fprintf(fid,sprintf('K-means %d clusters\n',numOfClusters));
fprintf(fid,sprintf('Hierarchical %d clusters
method=%s\n',numOfClusters,method));

fprintf(fid,sprintf('sig %d\n',sigThres));
% fprintf(fid,sprintf('Variables per set=%d\n',reducedVars));
% fprintf(fid,sprintf('Number of sets=%d\n',howManyPairs));

run IrisRuleCreation % (Dataset specific)
run RuleOutput

endTime = clock;
fprintf(fid,sprintf('\n Start Time Hour=%d Min=%d
Sec=%d\n',startTime(4),startTime(5),startTime(6)));
fprintf(fid,sprintf('\n End Time   Hour=%d Min=%d
Sec=%d\n',endTime(4),endTime(5),endTime(6)));

fclose(fid);
```

- HierarchicalClustering.m

```
function [retNum,retBase] =
HierarchicalClustering(sM,numClasses,maxClusters,method)
% Perform hierarchical clustering on the SOM
% Input arguments: SOM map
%                  number of classes in the data
%                  maximum number of clusters
%                  distance method
% Output arguments: number of clusters
%                   clustering base

sC = som_cllinkage(sM,method);

base = sC.base;
treeInfo = sC.tree;
numNeurons = length(base);

baseMatrix = zeros(numNeurons); % Matrix for base of every clustering
baseMatrix(numNeurons,:) = base;

DBs = zeros(maxClusters,1);
```

```
for i = 1:size(treeInfo,1)
    base(find(base==treeInfo(i,1))) = numNeurons+i;
    base(find(base==treeInfo(i,2))) = numNeurons+i;
    % Rescale base range to 1..numOfClusters
    baseMatrix(numNeurons-i,:) = BaseRescale(base);
    % Compute DB index when numOfClusters<maxClusters
    if numNeurons-i<=maxClusters
        [DBs(numNeurons-i),r] = db_index(sM,baseMatrix(numNeurons-i,:));
    end
end

% Select cluster with smallest index
[dum ind] = min(DBs(numClasses:length(DBs)));
retNum = ind+numClasses-1; % Number of clusters
retBase = baseMatrix(retNum,:)'; % Clustering base

figure; plot(DBs); % Plot DB indeces
figure; som_show(sM,'color',{retBase,sprintf('%d clusters',retNum)}); %
Visualize clustering
colormap(jet(retNum)), som_recolorbar % Change colormap
figure; som_show(sM,'color',{retBase,sprintf('%d clusters',retNum)}); %
Visualize clustering
colormap(gray(retNum)), som_recolorbar % Change colormap
figure; som_show(sM,'color',{retBase,sprintf('%d clusters',retNum)}); %
Visualize clustering
colormap(contrast(hsv(retNum))), som_recolorbar % Change colormap
return;
```

- BaseRescale.m

```
function base = BaseRescale(base)
% Rescale base range to 1..numOfClusters

ignore=[];
% if isempty(base), base = 1:dlen; end
if ~isempty(ignore), base(ignore) = NaN; end
cid = unique(base(isfinite(base)));
nc = length(cid);
if max(cid)>nc | min(cid)<1,
  b = base; for i=1:nc, base(find(b==cid(i))) = i; end
end
```

- KMeansClustering.m

```
function [retNum,retBase] = KMeansClustering(sM,numClasses,maxClusters)
% Perform K-means clustering on the SOM
% Input arguments: SOM map
%                  number of classes in the data
%                  maximum number of clusters
% Output arguments: number of clusters
%                   clustering base

[c, p, err, ind] = kmeans_clusters(sM,maxClusters,20,0); % Find
clustering for different k
```

```
[dummy,telikoK] = min(ind(numClasses:length(ind))); % Select the one with
smallest index
retNum = telikoK+numClasses-1; % Number of clusters
retBase = p{retNum}; % Clustering base

figure; plot(ind); % Plot DB indeces
figure; som_show(sM,'color',{p{retNum},sprintf('%d clusters',retNum)}); %
Visualize clustering
colormap(jet(retNum)), som_recolorbar % Change colormap
figure; som_show(sM,'color',{p{retNum},sprintf('%d clusters',retNum)}); %
Visualize clustering
colormap(gray(retNum)), som_recolorbar % Change colormap
figure; som_show(sM,'color',{p{retNum},sprintf('%d clusters',retNum)}); %
Visualize clustering
colormap(contrast(hsv(retNum))), som_recolorbar % Change colormap
return;
```

- StatisticsComp.m

```
% Cluster statistics

% Map units in each cluster
cl_cellD = cell(numOfClusters,1); % Denormalized codebook
cl_cellN = cell(numOfClusters,1); % Normalized codebook

for i = 1:mapSize(1)*mapSize(2)
    cl_cellD{clusteringBase(i)} =
[cl_cellD{clusteringBase(i)};mcdD(i,:)];
    cl_cellN{clusteringBase(i)} =
[cl_cellN{clusteringBase(i)};mcdN(i,:)];
end

meanMatrD = zeros(numOfVars,numOfClusters);
stdMatrD = zeros(numOfVars,numOfClusters);
meanMatrN = zeros(numOfVars,numOfClusters);
stdMatrN = zeros(numOfVars,numOfClusters);
maxMatr = zeros(numOfVars,numOfClusters);
minMatr = zeros(numOfVars,numOfClusters);

% Mean value, standard deviation, maximum, minimum for every variable
meanAllN = mean(mcdN);
stdAllN = std(mcdN);
maxAllN = max(mcdN);
minAllN = min(mcdN);
meanAllD = mean(mcdD);
stdAllD = std(mcdD);
maxAllD = max(mcdD);
minAllD = min(mcdD);

% Mean value, standard deviation, maximum, minimum for every variable in
% each cluster
for i = 1:numOfClusters
    meanMatrD(:,i) = (mean(cl_cellD{i},1))';
    stdMatrD(:,i) = (std(cl_cellD{i},0,1))';
    meanMatrN(:,i) = (mean(cl_cellN{i},1))';
    stdMatrN(:,i) = (std(cl_cellN{i},0,1))';
    maxMatr(:,i) = (max(cl_cellD{i},[],1))';
    minMatr(:,i) = (min(cl_cellD{i},[],1))';
```

```
end
```

- SigStar.m

```
% sig* algorithm

meansD = meanMatrD;
stdsD = stdMatrD;
sigThres = 50; % Sig* threshold value

numOfVariables = size(meansD,1);
numOfClusters = size(meansD,2);
choiceMatrix = zeros(numOfVariables,4,numOfClusters);
clusterRuleCell = cell(numOfClusters,1);

% Choose significance matrix
% significanceMatrix = 1./stdMatrN;
% significanceMatrix(find(significanceMatrix==Inf)) = 0;

for i = 1:numOfClusters
    significanceMatrix(:,i) = (stdAllN')./stdMatrN(:,i);
end
significanceMatrix(find(significanceMatrix==Inf)) = 0;


% Calculate maximum value for each variable
[Y,I] = max(significanceMatrix,[],2);
for i = 1:size(I,1)
    choiceMatrix(i,2,I(i)) = 1;
end
clear Y I

for i = 1:numOfClusters
    % Calculate percentages
    choiceMatrix(:,1,i) = 1:numOfVariables;
    choiceMatrix(:,3,i) =
(significanceMatrix(:,i)*100)/sum(significanceMatrix(:,i));
    choiceMatrix(:,:,i) = sortrows(choiceMatrix(:,:,i),[3]);
    % Calculate cumulative percentages
    choiceMatrix(numOfVariables,4,i) = choiceMatrix(numOfVariables,3,i);
    for j = numOfVariables-1:-1:1
        choiceMatrix(j,4,i) = choiceMatrix(j,3,i)+choiceMatrix(j+1,4,i);
    end
    choiceMatrix(:,:,i) = sortrows(choiceMatrix(:,:,i),[4]);
    % Select significant variables
    exceeds = 0;
    for j = 1:numOfVariables
        if (exceeds==0 || choiceMatrix(j,2,i)==1) &&
stdsD(choiceMatrix(j,1,i),i)~=0
            % Form rule
            clusterRuleCell{i} = [clusterRuleCell{i};choiceMatrix(j,1,i)
meansD(choiceMatrix(j,1,i),i)-2*stdsD(choiceMatrix(j,1,i),i)
meansD(choiceMatrix(j,1,i),i)+2*stdsD(choiceMatrix(j,1,i),i)];
            % Stop adding rules when cumulative percentage exceeds Sig*
threshold value
            if choiceMatrix(j,4,i)>sigThres
                exceeds = 1;
            end
```

```
        end
    end
end
```

- VariableSelection.m

```
% Select sets of variables based on covariance matrix

reducedVars = 2; % Number of variables in each set (Dataset specific)
howManyPairs = 1; % Number of sets to be chosen (Dataset specific)
combinationCell = cell(2,1);

% Find all possible combinations
vect = [1:numOfVars]; % (Dataset specific)
combinationCell{1} = nchoosek(vect,reducedVars);
combinationCell{2} = zeros(nchoosek(numOfVars-
1,reducedVars),numOfClusters);

clusterRuleCell = cell(numOfClusters,1);

for i = 1:numOfClusters
    for j = 1:size(combinationCell{1},1)
        tmpCl = [];
        meanTmpAll = [];
        % Select variables for the current combination
        for k = 1:reducedVars
            tmpCl = [tmpCl cl_cellN{i}(:,combinationCell{1}(j,k))];
            meanTmpAll = [meanTmpAll meanAllN(k)];
        end
        meanTmpCl = mean(tmpCl,1);
        % Calculate between-cluster covariance matrix
        SB = [];
        SB = (size(tmpCl,1)/size(mcdN,1))*(meanTmpCl'-
meanTmpAll')*(meanTmpCl-meanTmpAll);
        % Calcute within-cluster spread
        SI = zeros(reducedVars);
        for k = 1:size(tmpCl,1)
            SI = SI+(tmpCl(k,:)'-meanTmpCl')*(tmpCl(k,:)-meanTmpCl);
        end
        SI = SI./size(tmpCl,1);
        % Calculate measure
        invSI = inv(SI);
        tmp = invSI*SB;
        combinationCell{2}(j,i) = trace(tmp);
    end
    % Select the most significant variable sets
    tmpHowManyPairs = howManyPairs;
    while tmpHowManyPairs>0
        [dum1,dum2] = max(combinationCell{2}(:,i));
        combinationCell{2}(dum2,i) = 0;
        for k = 1:reducedVars
            entered = 0;
            if ~isempty(clusterRuleCell{i})
                for j = 1:size(clusterRuleCell{i},1)
                    % Check if rule is already present in the rule set
                    if
clusterRuleCell{i}(j,1)==combinationCell{1}(dum2,k) &&
```

```
clusterRuleCell{i}(j,2)==meanMatrD(combinationCell{1}(dum2,k),i)-
2*stdMatrD(combinationCell{1}(dum2,k),i)
                        entered = 1;
                        break;
                    end
                end
            end
            if ~entered
                % Form rules
                clusterRuleCell{i} =
[clusterRuleCell{i};combinationCell{1}(dum2,k)
meanMatrD(combinationCell{1}(dum2,k),i)-
2*stdMatrD(combinationCell{1}(dum2,k),i)
meanMatrD(combinationCell{1}(dum2,k),i)+2*stdMatrD(combinationCell{1}(dum
2,k),i)];
            end
        end
        tmpHowManyPairs = tmpHowManyPairs-1;
    end
end
```

- IrisRuleCreation.m (Οι συναρτήσεις για άλλα σύνολα δεδομένων είναι απλές μετατροπές του παρακάτω αρχείου)

```
labelThres = 0.4;
ruleCell = cell(numClasses,1);

fprintf(fid,sprintf('label %d\n\n',labelThres));

% Count map labels in each cluster
numClusters = zeros(numOfClusters,numClasses);
for j = 1:mapSize(1)*mapSize(2)
    if strcmp(classLabels(j),'1')
        numClusters(clusteringBase(j),1) =
numClusters(clusteringBase(j),1)+1;
    elseif strcmp(classLabels(j),'2')
        numClusters(clusteringBase(j),2) =
numClusters(clusteringBase(j),2)+1;
    elseif strcmp(classLabels(j),'3')
        numClusters(clusteringBase(j),3) =
numClusters(clusteringBase(j),3)+1;
    end
end

% Associate clusters with data classes
for j = 1:numOfClusters
    for i = 1:numClasses
        if numClusters(j,i)>(labelThres*sum(numClusters(j,:)))
            disp(sprintf('cluster %d contains labels %d',j,i));
            fprintf(fid,sprintf('cluster %d contains labels %d\n',j,i));
        end
    end
end

% Form rules for every class
for i = 1:numClasses
%     disp(sprintf('KLASH %d',i));
    for j = 1:numOfClusters
```

```
        if numClusters(j,i)>(labelThres*sum(numClusters(j,:)))
%             disp(clusterRuleCell{j});
            for k = 1:size(clusterRuleCell{j},1)
                ruleCell{i} = [ruleCell{i};clusterRuleCell{j}(k,1)
clusterRuleCell{j}(k,2) clusterRuleCell{j}(k,3) 0 0 0];
            end
        end
    end
end
```

- RuleOutput.m

```
% Output rules in a text file

% Count successfully and not successfully classified tuples for every
rule
for i = 1:numClasses
    for j = 1:size(ruleCell{i},1)
        for k = 1:length(DataIn)
            if DataIn(k,ruleCell{i}(j,1))>ruleCell{i}(j,2) &&
DataIn(k,ruleCell{i}(j,1))<ruleCell{i}(j,3)
                if OutClass(k)==i
                    ruleCell{i}(j,4) = ruleCell{i}(j,4)+1;
                else
                    ruleCell{i}(j,5) = ruleCell{i}(j,5)+1;
                end
            end
        end
        % Calculate rule significance
        ruleCell{i}(j,6) =
(ruleCell{i}(j,4)*ruleCell{i}(j,4))/((ruleCell{i}(j,5)+ruleCell{i}(j,4))*
(instancesPerClass(i)));
    end
end

for i = 1:numClasses
    fprintf(fid,sprintf('\n\n-------- Class %d --------\n',i));
    if ~isempty(ruleCell{i})
        % Sort rules in ascending order of significance
        fprintf(fid,'\nSignificance:\n');
        ruleCell{i} = sortrows(ruleCell{i},[6 5 4]);
        for j = size(ruleCell{i},1):-1:1
            % Output rules
            fprintf(fid,sprintf('Var%d min=%d max=%d suc=%d fail=%d
sf=%d\n',ruleCell{i}(j,1),ruleCell{i}(j,2),ruleCell{i}(j,3),ruleCell{i}(j
,4),ruleCell{i}(j,5),ruleCell{i}(j,6)));
            fprintf(fid,sprintf('|| (DataIn(k,%d)>%d &&
DataIn(k,%d)<%d)\n',ruleCell{i}(j,1),ruleCell{i}(j,2),ruleCell{i}(j,1),ru
leCell{i}(j,3)));
        end
        % Sort rules for best coverage of the class
        fprintf(fid,'\nBest coverage:\n');
        ruleCell{i} = sortrows(ruleCell{i},[4 6 5]);
        for j = size(ruleCell{i},1):-1:1
            % Output rules
            fprintf(fid,sprintf('Var%d min=%d max=%d suc=%d fail=%d
sf=%d\n',ruleCell{i}(j,1),ruleCell{i}(j,2),ruleCell{i}(j,3),ruleCell{i}(j
,4),ruleCell{i}(j,5),ruleCell{i}(j,6)));
```

```
            fprintf(fid,sprintf('&& DataIn(k,%d)>%d &&
DataIn(k,%d)<%d\n',ruleCell{i}(j,1),ruleCell{i}(j,2),ruleCell{i}(j,1),rul
eCell{i}(j,3)));
        end
    else
        fprintf(fid,'No rules for this class\n\n');
    end
end
```

Βιβλιογραφία

[1] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου, Τεχνητή νοημοσύνη, 2η έκδοση, Εκδόσεις Γαργατάνη, 2005

[2] T. Kohonen, Self-Organizing Maps, 2nd edition, Springer, 1997.

[3] A. Ultsch, H.P. Siemon, Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis, in Proc. of Int. Neural Networks Conf., 1990.

[4] J. Vesanto, E. Alhoniemi, Clustering of the Self-Organizing Map, IEEE Trans. on Neural Networks, vol. 11 ,pp. 586-600, 2000.

[5] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On Clustering Validation Techniques, Journal of Intelligent Information Systems, 17:2/3, pp. 107–145, 2001.

[6] A. B. Tickle, R. Andrews, M. Golea, J. Diederich, The Truth Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded Within Trained Artificial Neural Networks, IEEE Transactions on Neural Networks, vol. 9, no. 6, pp. 1057-1068, November 1998.

[7] S. Mitra, S. K. Pal, P. Mitra, Data mining in soft computing framework: a survey, IEEE Trans. Neural Networks, vol. 13(1), pp. 3–14, January 2002.

[8] J. Vesanto, J. Hollmen, An Automated Report Generation Tool for The Data Understanding Phase, International Workshop on Hybrid Intelligent Systems (HIS'01).

[9] T. Fawcett, Using rule sets to maximize ROC performance, in Proc. IEEE Int. Conf. Data Mining, pp. 131–138, 2001.

[10] R. Andonie, B. Kovalerchuk, Neural Networks for Data Mining: Constrains and Open Problems, ESANN'2004 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), pp. 449-458, April 2004.

[11] W. Duch, R. Setiono, J. M. Zurada, Computational Intelligence Methods For Rule-Based Data Understanding, Proceedings of the IEEE, Vol. 92, No. 5, pp.771-805, May 2004.

[12] J. Malone, K. McGarry, S. Wermter, C. Bowerman, Data mining using rule extraction from Kohonen self-organising maps, Neural Computing & Applications, Vo 15, Issue 1, pp. 9-17, 2006.

[13] A. Ultsch, D. Korus, Automatic Acquisition of Symbolic Knowledge from Subsymbolic Neural Networks, Proc. 3rd European Congress on Intelligent Techniques and Soft Computing EUFIT'95, Aachen/Germany, Aug. 28-31, 1995, vol. I, pp. 326-331.

[14] A. Webb, Statistical Pattern Recognition, 2nd edition, John Wiley & Sons, 2002.

[15] UCI repository of machine learning databases, J. Mertz and P. M. Murphy, [Online], http://www.ics.uci.edu/pub/machine-learning-databases

[16] SOM Toolbox, E. Alhoniemi, J. Himberg, J. Parhankangas, J. Vesanto, [Online], http://www.cis.hut.fi/projects/somtoolbox/