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

ΤΑΞΙΝΟΜΗΣΗ ΑΚΟΛΟΥΘΙΩΝ DNA ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΙΩΡΓΟΣ Κ. ΜΠΑΡΟΥΤΑΣ

ΑΡΙΣΤΟΤΕΛΗΣ Κ. ΤΣΙΑΜΑΣ

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

Αθήνα, Ιανουάριος 2005

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

ΤΑΞΙΝΟΜΗΣΗ ΑΚΟΛΟΥΘΙΩΝ DNA ΜΕ ΤΗ ΧΡΗΣΗ ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΙΩΡΓΟΣ Κ. ΜΠΑΡΟΥΤΑΣ

ΑΡΙΣΤΟΤΕΛΗΣ Ι. ΤΣΙΑΜΑΣ

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Ιανουαρίου 2005.

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

Αθήνα, Ιανουάριος 2005

Περίληψη

Σκοπός της συγκεκριμένης διπλωματικής εργασίας είναι η δημιουργία ενός τεχνητού νευρωνικού δικτύου, το οποίο να έχει τη δυνατότητα ταξινόμησης ακολουθιών DNA. Σε πρώτη θεώρηση, επιδιώκεται να γίνει μια συστηματική παρουσίαση βιολογικών προβλημάτων, τα οποία μπορούν να αντιμετωπιστούν με τεχνικές μηχανικής μάθησης. Επιπλέον, παρουσιάζονται αναλυτικά οι διαφορετικές μέθοδοι της υπολογιστικής νοημοσύνης και οι δυνατότητες που αυτές παρέχουν στη βιοπληροφορική. Σε δεύτερη φάση, υλοποιείται ένα νευρωνικό δίκτυο με δυνατότητες ταξινόμησης ακολουθιών DNA του βακτηρίου Escherichia Coli σε υποκινητές και μη-υποκινητές. Το νευρωνικό δίκτυο λειτουργεί με τον αλγόριθμο ανάστροφης διάδοσης ενώ για την εκπαίδευσή του χρησιμοποιήθηκαν τρία διαφορετικά σύνολα δεδομένων.

Πιο συγκεκριμένα, το Κεφάλαιο 1 δίνει μια γενικότερη περιγραφή του υπό μελέτη προβλήματος που προσπαθήσαμε να αναλύσουμε. Το Κεφάλαιο 2 περιλαμβάνει εισαγωγικές βιολογικές έννοιες και λειτουργίες απαραίτητες για την κατανόηση των προβλημάτων που ανακύπτουν. Επίσης περιγράφει εργαλεία από τον χώρο της υπολογιστικής επιστήμης για την μετέπειτα επίλυσή τους. Στο Κεφάλαιο 3 παρουσιάζονται οι βασικότερες τάσεις που υπάρχουν σήμερα στο χώρο της βιοπληροφορικής. Αυτές σε συνδυασμό με την ανάπτυξη των διαδικτυακών εργαλείων προσφέρονται για την δημιουργία νέων μορφών προγραμματισμού με απώτερο σκοπό την εξυπηρέτηση των ερευνητών στον τομέα της βιοπληροφορικής. Το Κεφάλαιο 4 αφορά στις διάφορες μεθόδους μηχανικής μάθησης, επικεντρώνοντας σε αυτές της υπολογιστικής νοημοσύνης και ακόμα περισσότερο στον αλγόριθμο της ανάστροφης διάδοσης. Στο Κεφάλαιο 5 παρουσιάζεται το πρόβλημα της ταξινόμησης ακολουθιών DNA του βακτηρίου E.Coli και οι διάφορες μορφές επίλυσής του, που έχουν προταθεί μέχρι σήμερα. Τέλος, στο Κεφάλαιο 6 παρουσιάζουμε τα τεχνικά χαρακτηριστικά του νευρωνικού δικτύου με το οποίο προσπαθήσαμε να επιλύσουμε το πρόβλημα που περιγράφηκε σε βιολογικό επίπεδο στο προηγούμενο κεφάλαιο, επίσης παρατίθενται πειραματικά αποτελέσματα με τον απαραίτητο σχολιασμό.

Λέξεις Κλειδιά

μηχανική μάθηση, υπολογιστική νοημοσύνη, τεχνητά νευρωνικά δίκτυα, ταξινομητές, αλγόριθμος ανάστροφης διάδοσης, βιοπληροφορική, ακολουθίες DNA, θεωρία υποκινητών

Abstract

The purpose of the present diploma thesis is the materialization of an artificial neural network capable of classifying DNA sequences. During the primary stage, a systematic approach of biological problems which can be addressed by machine learning methods is presented. Moreover, the different methods of computational intelligence and the impact they have in bioinformatics is discussed. In addition, a neural network classifier is implemented in order to categorize DNA sequences of the bacterium Escherichia Coli in two main classes, promoters and non-promoters. The artificial neural network utilizes the backpropagation algorithm. For the needs of developing, training and testing the performance and accuracy of the system, we employed three different data sets.

More precisely, Chapter 1 contains a brief description/study of the problem we worked towards to analyze. Chapter 2 presents basic biological terms and functions which are essential for the comprehension of the arising problems. Also, tools from the field of computational science are being presented and moreover their contribution in addressing certain issues of bioinformatics is clarified. Chapter 3 focuses in the mainstreams of bioinformatics nowadays combined with the development of internet tools and the novel trends of programming aimed to assist researchers. The subject of Chapter 4 is to present the different methods of machine learning; focusing on these of computational intelligence and specifically on the backpropagation algorithm. Chapter 5 presents the problem of classifying the DNA sequences belonging to bacterium E.Coli and the different methods that are applied to address the problem efficiently. Finally, Chapter 6 contains the technical characteristics of the artificial neural network implemented in order to resolve the problem described in the previous chapter from a biological point of view. Finally, the results of simulation experiments are presented followed by essential comments/discussion.

Key Words

machine learning, computational intelligence, artificial neural networks, classifiers, backpropagation algorithm, bioinformatics, DNA sequences, promoters' theory

Ευχαριστίες

Η εκπόνηση αυτής της διπλωματικής εργασίας έγινε υπό την επίβλεψη του καθηγητή κ. Α. Γ. Σταφυλοπάτη, τον οποίο θα θέλαμε να ευχαριστήσουμε για την προτροπή του να ασχοληθούμε με το συγκεκριμένο θέμα και τις πάντοτε χρήσιμες συμβουλές του. Ευχαριστούμε επίσης θερμά τον υποψήφιο διδάκτορα Χρήστο Φερλέ, χωρίς την βοήθεια του οποίου η παρούσα διπλωματική εργασία θα είχε καθυστερήσει περισσότερο από όσο ήδη καθυστέρησε. Τέλος, δεν ξεχνάμε όλους όσους με τον ένα ή τον άλλο τρόπο στάθηκαν δίπλα μας σε όλη τη διάρκεια των σπουδών μας, τόσο στις ευχάριστες όσο και στις δυσάρεστες στιγμές τους.

Πίνακας Περιεχομένων

Περίληψη.....5

Abstract.....6

Κεφάλαιο 1
1 Εισαγωγή.....13

Κεφάλαιο 2
2 Βασικές Βιολογικές Έννοιες και Λειτουργίες.....17
2.1 Εισαγωγή.....19
2.2 Βασικές Βιολογικές Έννοιες.....19
2.2.1 Τα Νουκλεϊκά Οξέα.....20
2.2.2 Οι Πρωτεΐνες.....22
2.3 Βασικές Βιολογικές Λειτουργίες.....25
2.3.1 Η διαδικασία της Αντιγραφής (DNA replication).....28
2.3.2 Η διαδικασία της Μεταγραφής (DNA transcription).....29
2.3.3 Η διαδικασία της Μετάφρασης (DNA translation).....33

Κεφάλαιο 3
3 Βιοπληροφορική – Υπολογιστική Βιολογία.....37
3.1 Εισαγωγή.....39
3.2 Βιοπληροφορική, Υπολογιστική Βιολογία και αντίστοιχοι τομείς έρευνας.....39
3.3 Κυριότεροι Τομείς Έρευνας στη Βιοπληροφορική.....40
3.3.1 Ανάλυση, σύγκριση, κατηγοριοποίηση και ταξινόμηση ακολουθιών βιολογικών δεδομένων.....40
3.3.2 Ανάπτυξη μεθοδολογιών που επιτρέπουν την ερμηνεία αποτελεσμάτων βιολογικής σημασίας.....44
3.3.3 Αποδοτική οργάνωση βιολογικών δεδομένων.....45
3.4 Ανάπτυξη νέων εργαλείων στη Βιοπληροφορική.....52
3.4.1 Η γλώσσα προγραμματισμού Java.....52
3.4.2 Επισκόπηση της γλώσσας προγραμματισμού Biojava.....54
3.4.3 Το πακέτο εργαλείων Βιοπληροφορικής της MATLAB.....55

Κεφάλαιο 4
4 Τεχνικές Μηχανικής Μάθησης στη Βιοπληροφορική.....59
4.1 Εισαγωγή.....61
4.2 Εισαγωγικά στοιχεία για τα Τεχνητά Νευρωνικά Δίκτυα.....61
4.2.1 Ιστορικά στοιχεία για τα Τεχνητά Νευρωνικά Δίκτυα.....62
4.2.2 Ορισμός του Τεχνητού Νευρωνικού Δικτύου.....64
4.2.3 Βασικές Αρχές Νευρωνικών Δικτύων.....64
4.2.4 Εκπαίδευση Τεχνητών Νευρωνικών Δικτύων.....66

4.2.5 Γενίκευση.....67
4.2.6 Εφαρμογές.....68
4.2.7 Κατηγορίες ΤΝΔ.....69
4.2.7.1 Ανάστροφη Διάδοση Σφάλματος (Error Back Propagation).....70
4.2.7.2 Χρήσιμες παρατηρήσεις και ορισμός παραμέτρων για τον Αλγόριθμο Ανάστροφης Διάδοσης..... 74
4.2.8 Πλεονεκτήματα και μειονεκτήματα των ΤΝΔ.....76
4.3 Εισαγωγικά στοιχεία για την ασαφή λογική και τα ασαφή συστήματα.. 77
4.3.1 Ιστορικά στοιχεία για την ασαφή λογική.....77
4.3.2 Βασικές έννοιες και ορισμοί στα ασαφή σύνολα..... 78
4.3.3 Ασαφής Συλλογιστική..... 78
4.3.4 Ασαφή Συστήματα.....79
4.4 Εισαγωγικά στοιχεία για τους Γενετικούς Αλγόριθμους.....80
4.4.1 Ιστορικά στοιχεία για τους Γενετικούς Αλγόριθμους.....81
4.4.2 Βασικός Γενετικός Αλγόριθμος..... 82
4.4.3 Επιλογή «τροχού ρουλέτας».....83
4.4.4 Πλεονεκτήματα και μειονεκτήματα των ΓΑ..... 84
4.5 Μέθοδοι Μηχανικής Μάθησης εκτός του χώρου της υπολογιστικής νοημοσύνης.....85

Κεφάλαιο 5
5 Η θεωρία των υποκινητών και το βακτήριο E.Coli..... 87
5.1 Η θεωρία των υποκινητών..... 89
5.2 Το βακτήριο E.Coli.....89
5.3 Η θεωρία υποκινητών στην E.Coli..... 92
5.4 Μέθοδοι επίλυσης του προβλήματος υποκινητών στην E.Coli.....95
5.5 Εφαρμογές μεθόδων μηχανικής μάθησης για την πρόγνωση υποκινητών στην E.Coli..... 95

Κεφάλαιο 6
6 Εφαρμογή αναγνώρισης ακολουθιών υποκινητών στο DNA του βακτηρίου E.Coli.....107
6.1 Γενικός σχεδιασμός..... 109
6.1.1 Δημιουργία συνόλων δεδομένων (data sets).....109
6.1.2 Επιλογή παραμέτρων..... 110
6.2 Εκτέλεση της εφαρμογής και πειραματικά αποτελέσματα.....111
6.2.1 Τεχνικά χαρακτηριστικά και επιλογή παραμέτρων για το backpropagation νευρωνικό δίκτυο.....111
6.2.2 Πειραματικά αποτελέσματα.....111
6.3 Σχολιασμός και συμπεράσματα..... 139

Επίλογος.....143

Παράρτημα 1
Παραδείγματα μορφής FASTA και EMBL..... 145

Παράρτημα 2
Παράδειγμα χρήσης του Bioinformatics Toolbox της MATLAB 7
για στατιστική επεξεργασία ακολουθίας..... 149

Παράρτημα 3
Σύνολα Δεδομένων..... 153

Παράρτημα 4
Πηγαίοι κώδικες.....157

Κεφάλαιο 1

Εισαγωγή

1 Εισαγωγή

Η πρόσφατη ολοκλήρωση του εγχειρήματος της αποκωδικοποίησης του ανθρώπινου γονιδιώματος αποτέλεσε ένα από τα πιο σημαντικά σημεία στην ιστορία των σύγχρονων επιστημών. Το εγχείρημα αυτό δεν θα ήταν εφικτό να επιτευχθεί χωρίς την ουσιαστική βοήθεια που προσφέρθηκε από άλλες επιστήμες, κυρίως από αυτήν της πληροφορικής. Η με εκθετικούς ρυθμούς αυξανόμενη ποσότητα πληροφορίας που αποθηκεύεται σε βιολογικές βάσεις δεδομένων απαιτεί συνεχώς καλύτερες και αποδοτικότερες εφαρμογές για την ανάκτηση των πληροφοριών που επιθυμεί ο ερευνητής. Παράλληλα, ο επιτυχής εντοπισμός συγκεκριμένων βιολογικών ακολουθιών, οι οποίες έχουν καθοριστικό ρόλο σε ζωτικής σημασίας λειτουργίες για τον υπο εξέταση οργανισμό, αποτελεί, επίσης, ένα άλλο σημείο, στο οποίο ερευνητές εξοικειωμένοι με την πληροφορική και παρεμφερείς μεθοδολογίες, όπως η υπολογιστική νοημοσύνη, θα μπορούσαν να συμβάλουν θετικά[1].

Πιο συγκεκριμένα, οι μέθοδοι της μηχανικής μάθησης, που έχουν μέχρι σήμερα αναπτυχθεί, έχουν τη δυνατότητα να βοηθήσουν στην επίλυση περίπλοκων βιολογικών προβλημάτων, που γίνονται ακόμα πιο περίπλοκα εξαιτίας της τεράστιας ποσότητας πληροφορίας που ενυπάρχει στις βιολογικές ακολουθίες. Ένα τέτοιο βιολογικό πρόβλημα είναι και αυτό της εύρεσης γονιδίων (gene finding). Το πρόβλημα αυτό εμφανίζεται στο σύνολο των ζωντανών οργανισμών. Τα γονίδια είναι υποακολουθίες της συνολικής ακολουθίας DNA ενός οργανισμού που ενεργοποιούν την διαδικασία της πρωτεϊνοσύνθεσης. Η πρωτεϊνοσύνθεση γίνεται σε δυο στάδια: το πρώτο είναι αυτό της μεταγραφής και το δεύτερο αυτό της μετάφρασης. Σήμερα είναι γνωστό ότι πριν από κάθε κωδικοποιούμενη περιοχή ενός γονιδίου υπάρχει μια ακολουθία (promoter – υποκινητής ή προαγωγέας)[2] που σηματοδοτεί την έναρξη της μεταγραφής και, κατά συνέπεια, και την αρχή της κωδικοποιούμενης περιοχής[3]. Γίνεται, επομένως, αντιληπτό ότι το πρόβλημα της εύρεσης υποκινητών αποτελεί ένα από τα σημαντικότερα στοιχεία και για το πρόβλημα εύρεσης γονιδίων.

Στην παρούσα διπλωματική εργασία επιλέξαμε να ασχοληθούμε με έναν σχετικά απλό οργανισμό, αυτόν του βακτηρίου Escherichia Coli και την κατηγοριοποίηση ακολουθιών DNA σε υποκινητές και μη-υποκινητές, με τη χρήση νευρωνικών δικτύων που λειτουργούν σύμφωνα με τον αλγόριθμο ανάστροφης διάδοσης. Επιλέξαμε τον συγκεκριμένο οργανισμό για δύο λόγους: πρώτον, εξαιτίας του μεγάλου εύρους βιβλιογραφίας και αρθρογραφίας για αυτό, τόσο από βιολογικής άποψης, όσο και από πλευράς εφαρμογής τεχνικών μηχανικής μάθησης· δεύτερον, εξαιτίας της σχεδόν πλήρους αποκωδικοποίησης της DNA ακολουθίας του, το μέγεθος της οποίας επιτρέπει μια σχετικά γρήγορη εκπαίδευση για το νευρωνικό δίκτυο που δημιουργούμε, σε σύγκριση τουλάχιστον με το ανθρώπινο γονιδίωμα ή άλλους πιο «εξελιγμένους» οργανισμούς.

Το πρόβλημα της ταξινόμησης ακολουθιών DNA ανάμεσα σε υποκινητές και μη-υποκινητές ανήκει στο γενικότερο πρόβλημα της ταξινόμησης (classification) και ειδικότερα στο πρόβλημα της δυαδικής ταξινόμησης (binary classification), μιας και έχουμε δύο κατηγορίες. Στη δυαδική ταξινόμηση εισάγουμε τόσο θετικά πρότυπα εισόδου (υποκινητές – promoters) όσο και αρνητικά πρότυπα εισόδου (μη-υποκινητές – non-promoters). Στόχος μας είναι να υπάρξει μετά την εκπαίδευση η δυνατότητα σωστής ταξινόμησης άγνωστων ακολουθιών DNA και, επομένως, η σωστή πρόβλεψη ακολουθιών υποκινητών, άρα και γονιδίων.

Οι είσοδοι του νευρωνικού δικτύου προέρχονται από ακολουθίες DNA μήκους 101 χαρακτήρων. Οι βιολογικές ακολουθίες μετά από κατάλληλη

κωδικοποίηση εισάγονται στο νευρωνικό δίκτυο. Τα αρνητικά πρότυπα εισόδου δημιουργήθηκαν με τρεις τρόπους: είτε με την ανάγνωση όλου του γονιδιώματος, είτε με δημιουργία τυχαίων ακολουθιών, είτε με τις ακολουθίες DNA των γονιδίων αφού διαγράφηκαν οι περιοχές των υποκινητών. Το νευρωνικό δίκτυο εκπαιδεύτηκε επίσης και από το σύνολο δεδομένων που βρίσκεται στο UCI Repository of Machine Learning Databases[4]. Ο αλγόριθμος που χρησιμοποιείται είναι ο αλγόριθμος ανάστροφης διάδοσης (backpropagation algorithm).

Πέρα από το συγκεκριμένο πρόβλημα που περιγράψαμε συνοπτικά παραπάνω και το οποίο οριοθετείται στο χώρο της μηχανικής μάθησης, θεωρώντας ότι ένα από τα σημαντικότερα προβλήματα που εντοπίζονται σήμερα στο χώρο της βιοπληροφορικής είναι η έλλειψη γνώσεων πληροφορικής από τους επιστήμονες της βιολογίας και αντίστοιχα η έλλειψη γνώσεων βιολογίας από τους επιστήμονες της πληροφορικής, παρουσιάζουμε μια όσο το δυνατόν κατατοπιστικότερη και πιο συμπυκνωμένη εικόνα των βασικότερων εννοιών και λειτουργιών της σύγχρονης βιολογίας που αφορούν τη διαδικασία της πρωτεϊνοσύνθεσης. Επίσης παρουσιάζουμε τις σημαντικότερες βιολογικές βάσεις βιολογικών δεδομένων και κάποια από τα εργαλεία που διατίθενται μέσω του διαδικτύου από βάσεις δεδομένων και είναι ελεύθερα προς χρήση από οποιονδήποτε, έτσι ώστε να διευκολύνουμε επόμενες εργασίες που θα επιχειρήσουν μια εφαρμογή που θα προσδιορίζεται στον χώρο της βιοπληροφορικής[5].

[1] Wang, Shapiro and Shasha – Pattern Discovery in Biomolecular Data: Tools, Techinques and Applications – Oxford University Press, New York, 1999.

[2] χρησιμοποιούμε από εδώ και στο εξής τον όρο υποκινητής καθώς η ελληνική βιβλιογραφία και αρθρογραφία χρησιμοποιούν, μέχρι σήμερα τουλάχιστον, και τους δυο όρους εξίσου.

[3] B.Alberts, A.Johnson, J.Lewis, M.Raff, K.Roberts, P.Walter - Molecular Biology of the Cell - Garland Publishing Inc., New York, 2002.

[4] http://www.ics.uci.edu/~mlearn/mlrepository.htm

[5] Θεωρούμε ότι αρχικά μια γενικότερη εικόνα πάνω στο πρόβλημα εφαρμογής τεχνικών μηχανικής μάθησης στη βιοπληροφορική δίνεται με τον καλύτερο τρόπο από το βιβλίο: P. Baldi and S. Brunak – Bioinformatics: The machine learning approach – MIT press, 2002.

Κεφάλαιο 2

Βασικές Βιολογικές Έννοιες και Λειτουργίες

2 *Βασικές Βιολογικές Έννοιες και Λειτουργίες*

2.1 *Εισαγωγή*

Ο όρος κύτταρο (cell) προέρχεται από την λατινικής προέλευσης λέξη cellulae, που χρησιμοποιήθηκε για πρώτη φορά το 1665 όταν ο R.Hooke, με ένα αυτοσχέδιο μικροσκόπιο, εξέτασε λεπτά κομμάτια φελλού. Παρατήρησε ότι αυτά συνιστούσαν ένα δίκτυο μικροσκοπικών τμημάτων σε σχήμα κιβωτίου και για αυτόν τον λόγο τα ονόμασε με αυτόν τον τρόπο, χωρίς να γνωρίζει ότι έτσι όρισε την βασική μονάδα ζωής στην βιολογία: το κύτταρο. Με την ανάπτυξη τα επόμενα χρόνια της μεγεθυντικής ικανότητας των μικροσκοπίων, οι επιστήμονες κατάφεραν να παρατηρήσουν τον πυρήνα (nucleus) του κυττάρου, αλλά και τις λειτουργίες του, με αποτέλεσμα την διαμόρφωση της σύγχρονης κυτταρικής θεωρίας, η οποία βασίζεται σε τρεις βασικούς κανόνες:

Α. Όλοι οι οργανισμοί αποτελούνται από ένα ή περισσότερα κύτταρα.
Β. Το κύτταρο αποτελεί την βασική δομική μονάδα όλων των οργανισμών.
Γ. Όλα τα κύτταρα προκύπτουν μόνο από προϋπάρχοντα κύτταρα.

Η σύγχρονη βιολογία του κυττάρου περικλείει τρία διακριτά αλλά παράλληλα αλληλοσυμπληρούμενα επιστημονικά πεδία. Αυτά είναι: η κυτταρολογία (cytology), η βιοχημεία (biochemistry) και η γενετική (genetics). Η κυτταρολογία ασχολείται με την δομή του κυττάρου και βασίζεται κυρίως στην μικροσκοπία, ενώ η βιοχημεία καλύπτει την χημεία μιας βιολογικής δομής και λειτουργίας. Η γενετική επικεντρώνει στην διάχυση των γενετικών πληροφοριών, που ενυπάρχουν μέσα στο κύτταρο, σε συγκεκριμένες μορφές και την αποκωδικοποίησή τους με επίσης συγκεκριμένο τρόπο[1]. Βασικές έννοιες που καθορίζουν την ροή και την αποκωδικοποίηση των γενετικών πληροφοριών είναι αυτές του DNA, του γονιδίου, του χρωμοσώματος, του RNA και της πρωτεΐνης.

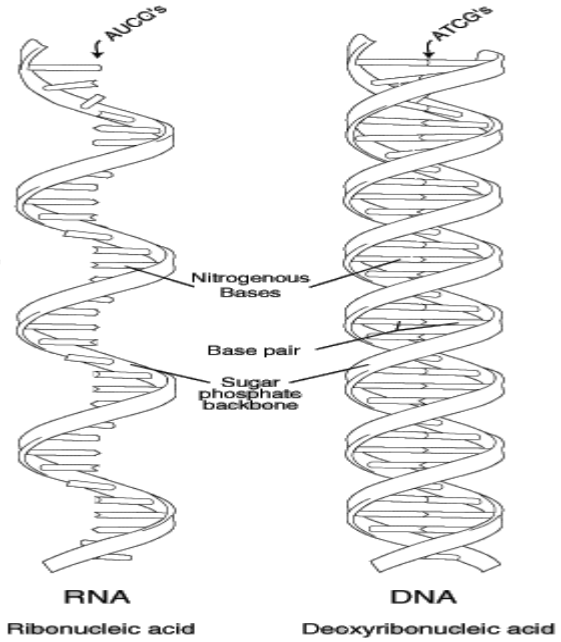
2.2 *Βασικές Βιολογικές Έννοιες*

Μία από τις κυριότερες ιδιομορφίες της ζώσας ύλης είναι ότι περιέχει μακρομόρια, μόρια πολύ μεγαλύτερα από αυτά που αποτελούν το αντικείμενο των ερευνών της χημείας. Τα μακρομόρια αυτά εμπλέκονται στην χημική δραστηριότητα στο εσωτερικό του κυττάρου, δραστηριότητα η οποία όχι μόνο καταφέρνει να δημιουργεί πολυποίκιλα μοριακά συγκροτήματα, αλλά επιπλέον καταφέρνει και να τα αναπαράγει[2]. Τα μακρομόρια που εμπλέκονται στο «πρόγραμμα» του κυττάρου – «πρόγραμμα» το οποίο ποικίλει ανάλογα με το κάθε κύτταρο και τον οργανισμό στον οποίο ανήκει – διαχωρίζονται σε δυο μεγάλες κατηγορίες:

Α. στα **νουκλεϊκά οξέα**
Β. στις **πρωτεΐνες**

2.2.1 Τα Νουκλεϊκά Οξέα

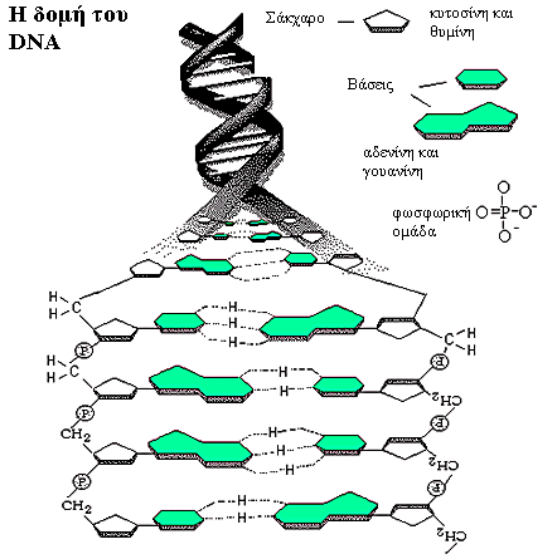
Τα νουκλεϊκά οξέα (nucleic acids) είναι εκείνα τα μακρομόρια τα οποία περιέχουν τη γενετική πληροφορία και εξασφαλίζουν τη μεταβίβασή της. Όπως και οι πρωτεΐνες, τα νουκλεϊκά οξέα είναι πολυμερή μόρια τα οποία συνίστανται από μια γραμμική αλληλουχία δομικών μονάδων που ανήκουν σε έναν περιορισμένο αριθμό μικρών μορίων απλής σχετικά δομής, παρόμοιας με αυτήν των συνηθισμένων μορίων της οργανικής χημείας. Πιο συγκεκριμένα για τα νουκλεϊκά οξέα, η συγκρότηση αυτής της γραμμικής αλληλουχίας πραγματοποιείται με βάση τέσσερις δομικές μονάδες, τέσσερα διαφορετικά **νουκλεοτίδια**. Το κάθε νουκλεοτίδιο αποτελείται από τρία απλούστερα μόρια, που συνδέονται μεταξύ τους με χημικούς δεσμούς: μια αζωτούχα βάση, ένα σάκχαρο και μια φωσφορική ομάδα. Το μόνο από τα τρία επιμέρους μόρια ενός νουκλεοτιδίου που μεταβάλλεται είναι η αζωτούχα βάση, ενώ το σάκχαρο και η φωσφορική ομάδα παραμένουν τα ίδια και για τα τέσσερα είδη δομικών νουκλεοτιδίων. Μπορούμε, δηλαδή, να πούμε ότι τα δυο τελευταία αποτελούν τη «σπονδυλική στήλη» της πολυνουκλεοτιδικής αλυσίδας, πάνω στην οποία έχουν προσαρτηθεί οι τέσσερις διαφορετικές αζωτούχες βάσεις. Κατά συνέπεια μπορούμε να θεωρήσουμε ότι η διαδοχή των νουκλεοτιδίων αντιστοιχεί πρακτικά στην διαδοχή των αζωτούχων βάσεων που ανήκουν στα διαφορετικά νουκλεοτίδια.**Οι δυο τύποι νουκλεϊκών οξέων που υπάρχουν είναι το DNA και το RNA** (Σχήμα 2.1).



Σχήμα 2.1: Η διπλή έλικα του DNA και η αλυσίδα RNA

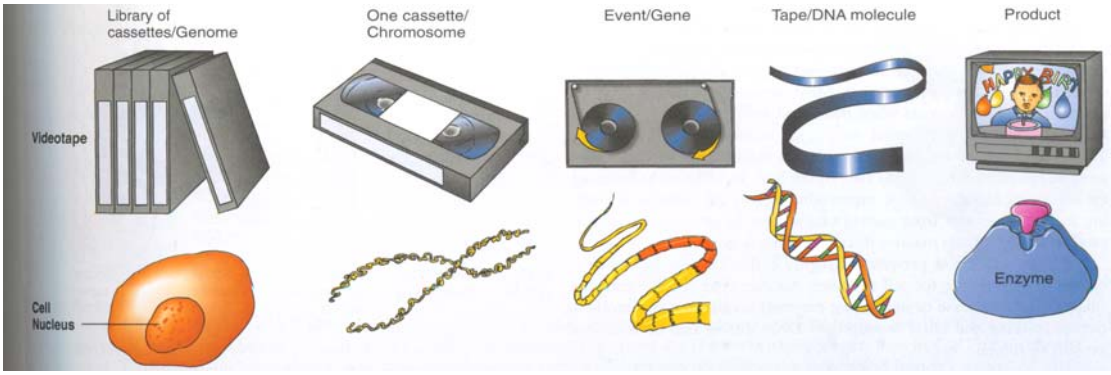
Το DNA (δεσοξυριβονουκλεϊκό οξύ) είναι το υπόβαθρο και ο φορέας των γενετικών πληροφοριών. Οι τέσσερις αζωτούχες βάσεις που μπορούν να περιγράψουν την γραμμική απεριοδική ακολουθία του DNA είναι η αδενίνη, η κυτοσίνη, η γουανίνη και η θυμίνη, οι οποίες συμβολίζονται με τα αρχικά των αντίστοιχων ονομάτων στο λατινικό αλφάβητο: A, C, G και T. Το DNA αποτελείται από δύο πολυνουκλεοτιδικές αλυσίδες όπου η μία περιελίσσεται στην άλλη και οι οποίες συνδέονται μεταξύ τους με δεσμούς υδρογόνου ανάμεσα στις συμπληρωματικές βάσεις. Οι συμπληρωματικές βάσεις στο DNA είναι η αδενίνη (A) με την θυμίνη (T) και η γουανίνη (G) με την κυτοσίνη (C). Με αυτόν τον τρόπο δημιουργείται ένα σπειροειδές μόριο, η γνωστή **διπλή έλικα** του DNA. Η κάθε έλικα του DNA έχει μια εγγενή κατευθυντικότητα που συμβολίζεται με βάση την αρίθμηση των ατόμων άνθρακα στο σάκχαρο, έχοντας ως σημείο εκκίνησης το άτομο του άνθρακα που συνδέεται με την αζωτούχα βάση. Με αυτόν τον τρόπο η βασική πολυνουκλεοτιδική αλυσίδα συνδέεται με το 3° άτομο άνθρακα στο επόμενο νουκλεοτίδιο και με το 5° άτομο άνθρακα με την φωσφορική ομάδα. Η κατεύθυνση αυτή συμβολίζεται σαν 5'→ 3'. Η συμπληρωματική πολυνουκλεοτιδική αλυσίδα έχει αντιπαράλληλη κατευθυντικότητα, και προφανώς συμβολίζεται σαν 3'→ 5'. Με την λειτουργία της μεταγραφής και της μετάφρασης που θα περιγραφεί μετέπειτα, το

DNA καθορίζει ποια πρωτεΐνη θα οικοδομηθεί και ποιον ρόλο θα έχει αυτή μέσα σε έναν συγκεκριμένο οργανισμό(Σχήμα 2.2).



Σχήμα 2.2: Η δομή του DNA

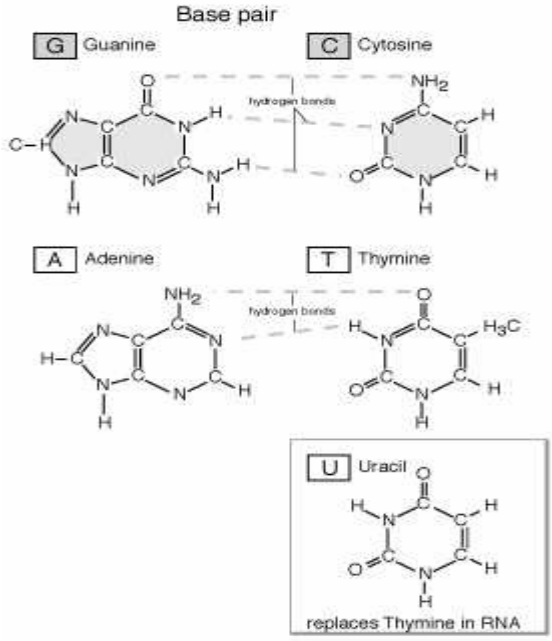
Κάθε γονίδιο (gene) αντιπροσωπεύει και μια διαφορετική πληροφορία, η οποία υποδεικνύει τον τρόπο οικοδόμησης μιας συγκεκριμένης πρωτεΐνης, ή σύμφωνα με τις τελευταίες έρευνες, μιας ή περισσότερων στενά συνδεδεμένων πολυπεπτιδικών αλυσίδων. Το γονίδιο αντιστοιχεί σε ένα τμήμα του DNA. Ας υποθέσουμε στο σημείο αυτό ότι κάθε γενετική πληροφορία, δηλαδή κάθε γονίδιο, τυπώνεται σε ένα βιβλίο, σε μια γλώσσα με τέσσερις μόνο χαρακτήρες, τους A, C, G, T. Κάθε τέτοια πληροφορία έχει μέσο μήκος μερικές χιλιάδες γράμματα και καλύπτει στο βιβλίο έκταση μιας σελίδας. Αν το σύνολο των γενετικών πληροφοριών που δεχόμαστε είναι 4.000, τότε το σύνολο αυτό θα περιέχεται σε έναν μοναδικό τόμο με 4.000 περίπου σελίδες. Αν όμως το σύνολο των γενετικών πληροφοριών είναι μεγαλύτερης τάξης μεγέθους από αυτό, θα ήταν βοηθητικό να μην έχουμε μόνο έναν τόμο αλλά περισσότερους. **Τον ρόλο αυτό στους εξελιγμένους οργανισμούς έχουν τα χρωμοσώματα (chromosomes), στα οποία κατανέμονται οι γενετικές πληροφορίες.** Οι εξελιγμένοι οργανισμοί θα μπορούσαμε να πούμε ότι είναι οι οργανισμοί οι οποίοι αποτελούνται από περισσότερα του ενός κύτταρα με πυρήνα, οργανισμοί που ονομάζονται και **ευκαρυωτικοί οργανισμοί**, ενώ οι οργανισμοί οι οποίοι είναι μονοκύτταροι και δεν έχουν πυρήνα ονομάζονται **προκαρυωτικοί** (π.χ. βακτήρια).



Σχήμα 2.3: Η σχέση μεταξύ του γονιδιώματος, των χρωμοσωμάτων, των γονιδίων και του DNA, μπορεί να γίνει ευκολότερα κατανοητή αν κάνουμε την αντιστοιχία της οργάνωσης αυτής στο εσωτερικό ενός κυττάρου με την οργάνωση μιας συλλογής βιντεοκασετών. Α) Σε κάθε επίπεδο (συλλογή βιντεοκασετών, κασέτα, στη διαδικασία που επιλέγουμε να χρησιμοποιήσουμε την κασέτα (εγγραφή, αναπαραγωγή κοκ), η βασική πληροφορία υπάρχει πάντα στην κασέτα. Β) Αντίστοιχα με μια βιντεοκασέτα, το μόριο του DNA μπορεί να αναπαραχθεί, να επεξεργαστεί κοκ. Γ) Όπως το περιεχόμενο της βιντεοκασέτας που βρίσκεται στην ταινία είναι προσεκτικά περιτυλιγμένο και αν διαταραχθεί η περιτύλιξη αυτή θα υπάρξει πρόβλημα στην αναπαραγωγή της κασέτας, έτσι και το μόριο του DNA είναι προσεκτικά διατεταγμένο ούτως ώστε η πληροφορία που περιέχει να μπορεί να αποκωδικοποιηθεί ομαλά. Δ) Όπως είναι απαραίτητη η ύπαρξη συσκευής βίντεο για να μεταφραστεί η ταινία της βιντεοκασέτας σε εικόνες, έτσι και ο κώδικας του DNA χρειάζεται την ύπαρξη ενός ειδικού μηχανισμού στο κύτταρο για να εμφανίσει τις πληροφορίες που διαθέτει. Ε) Παρόμοια με την ταινία

της βιντεοκασέτας η πληροφορία του DNA μορίου γίνεται κατανοητή μόνο όταν κινείται προς μια συγκεκριμένη κατεύθυνση. Ζ) Σε καμία περίπτωση δεν παίζονται συγχρόνως όλες οι βιντεοκασέτες. Παρόμοια, τα γονίδια δεν αποκωδικοποιούν την πληροφορία που διαθέτουν συγχρόνως[3].

Το σύνολο των γενετικών πληροφοριών που κληρονομεί ένας έμβιος οργανισμός στους απογόνους του ονομάζεται γονιδίωμα (genome). Για παράδειγμα, το γονιδίωμα του ανθρώπου είναι το σύνολο των γενετικών πληροφοριών που λαμβάνουμε από τα 23 χρωμοσώματα που έχει (Σχήμα 2.3). Το μέγεθος του γονιδιώματος μετριέται συνήθως σε **ζεύγη βάσεων (base pairs – bp)** (Σχήμα 2.4). Το μέγεθος του γονιδιώματος ποικίλλει, για αυτόν τον λόγο χρησιμοποιούνται πολλές φορές τα πολλαπλάσια του: Kb (10^3 bp), Mb (10^6 bp) και Gb (10^9 bp).



Σχήμα 2.4: οι αντιστοιχίες μεταξύ των βάσεων – base pairs (bp)

Το RNA (ριβονουκλεϊκό οξύ) είναι υπεύθυνο για την ορθή μεταφορά της γενετικής πληροφορίας και την αποκωδικοποίησή της από το DNA. Οι κύριες διαφορές του RNA από το DNA, όσον αφορά τη δομή του, είναι η ύπαρξη του σακχάρου της ριβόζης αντί της δεσοξυριβόζης στο DNA και η αντικατάσταση της βάσης της θυμίνης με αυτήν της ουρακίλης (T→U). Παράλληλα, το RNA στις περισσότερες περιπτώσεις δεν βρίσκεται σε μορφή διπλής έλικας, αλλά έχει μόνο μια πολυνουκλεοτιδική αλυσίδα. Οι λειτουργίες που επιτελεί το RNA εξαρτώνται από την ειδική μορφή που αυτό παίρνει: το **αγγελιοφόρο RNA (mRNA)** είναι υπεύθυνο για τη μεταφορά του γενετικού κώδικα που έχει αντιγραφεί από το DNA, το **ριβοσωμικό RNA (rRNA)** εμπλέκεται στην διαδικασία κατασκευής των ριβοσωμάτων, ενώ το **μεταφορικό RNA (tRNA)** εμπλέκεται στη σύνθεση των αμινοξέων σε μια πρωτεϊνική αλυσίδα.

2.2.2 Οι Πρωτεΐνες

Οι πρωτεΐνες είναι μακρομόρια με σαφώς καθορισμένη χημική δομή, παρουσιάζουν μεγάλη ποικιλία ιδιοτήτων και σχήματος και είναι απαραίτητες σε όλες σχεδόν τις βιολογικές λειτουργίες[4]. Η χημική σύσταση των πρωτεϊνών είναι μάλλον απλή. Οι πρωτεΐνες είναι πολυμερή, στα οποία μικρά μόρια 20 διαφορετικών ειδών, τα L-α-αμινοξέα, συνδέονται και σχηματίζουν μακριές αλυσίδες χωρίς διακλαδώσεις. Τα αμινοξέα παρουσιάζουν όλα την ίδια γενική δομή. Ένα κεντρικό άτομο άνθρακα (που ονομάζεται κατά συνθήκη α-άτομο άνθρακα, C*α)* συνδέεται με 4 διαφορετικά άτομα ή ομάδες ατόμων: ένα άτομο υδρογόνου, μία καρβοξυλομάδα, μία αμινομάδα και μία από 20 δυνατές διαφορετικές πλευρικές αλυσίδες διαφόρων χημικών τύπων (-R)

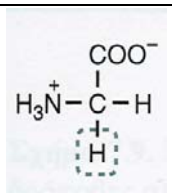
COOH
|
CHNH2
|
R

Σχήμα 2.5: Γενικός χημικός τύπος αμινοξέων

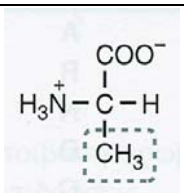
Τα 20 διαφορετικά αμινοξέα διακρίνονται ανάλογα με τις R-ομάδες ή πλευρικές αλυσίδες τους. Η σύνδεση των μορίων δύο αμινοξέων, με τον χαρακτηριστικό τρόπο που απαντάται στις πρωτεΐνες, περιλαμβάνει μία απλή αντίδραση συμπύκνωσης, στην οποία αποβάλλεται ένα μόριο νερού και σχηματίζεται ένας πεπτιδικός δεσμός. Τα πρωτεϊνικά μόρια αποτελούνται από μία ή περισσότερες πολυπεπτιδικές αλυσίδες, κάθε μία από τις οποίες μπορεί να αποτελείται από μερικές εκατοντάδες αμινοξικά κατάλοιπα με μία συγκεκριμένη σειρά, η οποία είναι γνωστή ως **αμινοξική ακολουθία** ή **αλληλουχία** (**amino-acid sequence**) ή **πρωτοταγής δομή.** Μία πολυπεπτιδική αλυσίδα έχει κατεύθυνση, αφού έχει διαφορετικά άκρα, δηλαδή, το **άμινο-άκρο** και το **καρβόξυλο-άκρο.** Κατά συνθήκη, το άμινο-άκρο θεωρείται η αρχή μίας πολυπεπτιδικής αλυσίδας. Η ακολουθία των αμινοξέων σε μια πολυπεπτιδική αλυσίδα γράφεται αρχίζοντας με το αμινοτελικό κατάλοιπο. Μία πολυπεπτιδική αλυσίδα αποτελείται από ένα κανονικά επαναλαμβανόμενο μέρος, την **κύρια αλυσίδα** (main chain) και ένα μεταβλητό μέρος, τις χαρακτηριστικές πλευρικές αλυσίδες ή ομάδες (side chains) των αμινοξικών καταλοίπων. Η κύρια αλυσίδα ονομάζεται πολλές φορές σπονδυλική στήλη (backbone). Μπορούμε πλέον να κατανοήσουμε πώς προκύπτει η καταπληκτική ποικιλία των πρωτεϊνών. Δεδομένων δομικών στοιχείων 20 διαφορετικών ειδών, τα οποία μπορούν να ληφθούν σε οποιουσδήποτε αριθμούς και να διευθετηθούν με οποιαδήποτε σειρά σε μια γραμμική ακολουθία, είναι φανερό ότι ο αριθμός των πρωτεϊνικών μορίων που μπορούν να κατασκευαστούν είναι τεράστιος. Εάν υποθέσουμε ότι περιοριζόμαστε σε πολυπεπτιδικές αλυσίδες που αποτελούνται μόνο από 100 αμινοξικά κατάλοιπα, έναν αριθμό που είναι σχετικά μικρός για μία πραγματική πρωτεΐνη, κάθε κατάλοιπο μπορεί να εκλεγεί κατά 20 διαφορετικούς τρόπους. Έτσι, 20^{100} διαφορετικά είδη πρωτεϊνικών μορίων του μεγέθους αυτού μπορούν να δημιουργηθούν, αριθμός που είναι πολύ μεγαλύτερος από τον υποτιθέμενο αριθμό των ατόμων του σύμπαντος.
Τα αμινοξέα διαφέρουν μεταξύ τους στη φύση της ομάδας R (βλέπε Σχήμα 2.5) και παρουσιάζονται στον Πίνακα 2.1, όπου τα πλαίσια των διακεκομμένων γραμμών υποδεικνύουν τις πλευρικές αλυσίδες. Η κατάταξη των αμινοξέων μπορεί να γίνει με διαφορετικούς τρόπους. Στον Πίνακα 2.1[5] τα αμινοξέα έχουν καταταγεί με βάση το ρόλο της πλευρικής αλυσίδας στον σχηματισμό της χωροδιάταξης των πρωτεϊνών.

Πίνακας 2.1: Τα βασικά αμινοξέα και η κατάταξή τους σε τέσσερις κατηγορίες

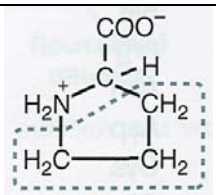
1)



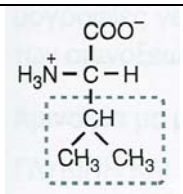
Γλυκίνη



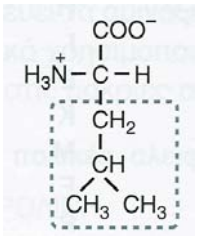
Αλανίνη

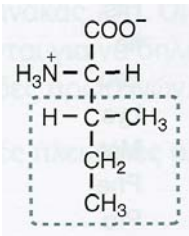


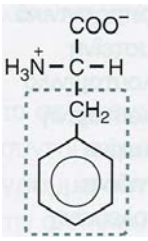
Προλίνη



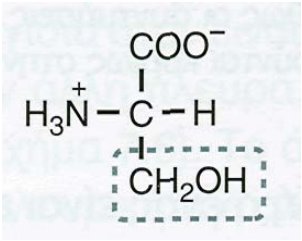
Βαλίνη

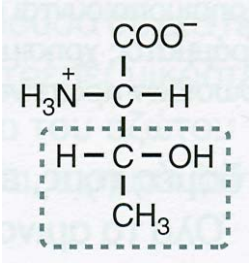

Λευκίνη

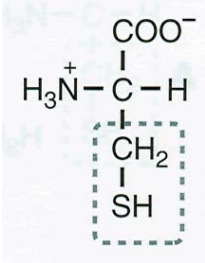

Ισολευκίνη

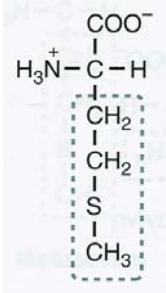

Φαινυλαλανίνη

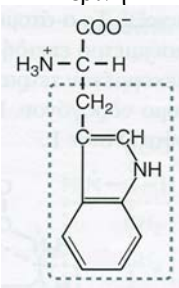
2)

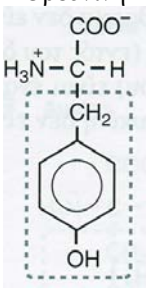

Σερίνη

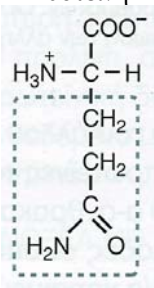

Θρεονίνη

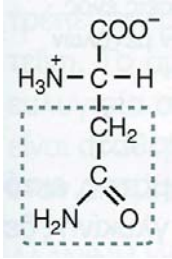

Κυστεΐνη


Μεθειονίνη

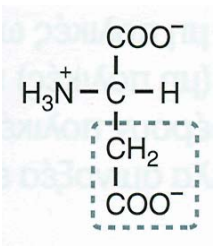

Τρυπτοφάνη

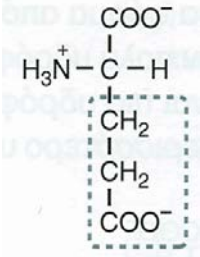

Τυροσίνη


Ασπαραγίνη

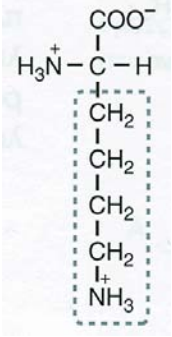

Γλουταμίνη

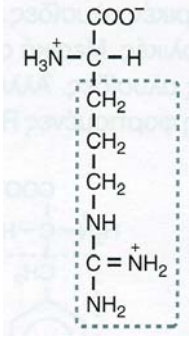
3)

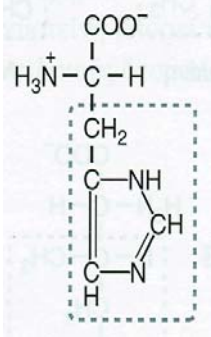

Ασπαραγινικό οξύ


Γλουταμινικό οξύ

4)


Λυσίνη


Αργινίνη


Ιστιδίνη

Με τον τρόπο αυτό έχουν διαχωριστεί 4 ομάδες αμινοξέων:

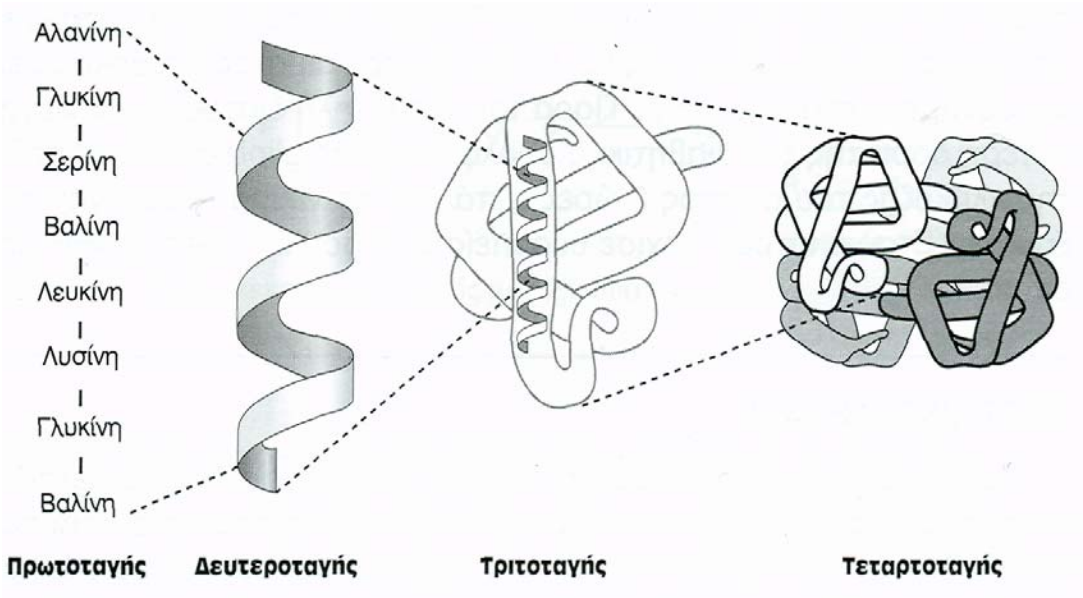
α. Αμινοξέα με μη πολική ομάδα R, δηλαδή με πλευρική αλυσίδα αποκλειστικά από υδρογονάνθρακα.

β. Αμινοξέα με μη ιονιζόμενες, αλλά πολικές ομάδες (όπως -OH, -SH, -$CONH_2$) στην πλευρική αλυσίδα.

γ. Όξινα αμινοξέα, «μονοαμινο-δικαρβοξυλικά οξέα».

δ. Βασικά αμινοξέα ή «διαμινο-μονοκαρβοξυλικά» οξέα.

Το 1959, οι ***Linderstrom-Lang και Schelman*** υποστήριξαν ότι στις πρωτεΐνες μπορούν να αναγνωριστούν τέσσερα επίπεδα οργάνωσης[6]: η πρωτοταγής, η δευτεροταγής, η τριτοταγής και η τεταρτοταγής δομή (Σχήμα 2.6).



Σχήμα 2.6: Δομικά επίπεδα σε μία πρωτεΐνη

Ας δώσουμε έναν σύντομο χαρακτηρισμό αυτών των κατηγοριών:

α. Η πρωτοταγής δομή αντιστοιχεί στην αμινοξική ακολουθία, στη διάταξη δηλαδή των αμινοξέων σε μία ή περισσότερες πολυπεπτιδικές αλυσίδες.

β. Η δευτεροταγής δομή αναφέρεται στην κανονική στερεοδιάταξη τμημάτων της πολυπεπτιδικής αλυσίδας.

γ. Η τριτοταγής δομή ενός πρωτεϊνικού μορίου ή μίας υπομονάδας του αναφέρεται στην τριδιάστατη δομή ή τη στερεοδιάταξη όλων των ατόμων του.

δ. Η τεταρτοταγής δομή μίας πρωτεΐνης είναι η διάταξη στο χώρο των υπομονάδων της, δηλαδή των πολυπεπτιδικών αλυσίδων από τις οποίες αποτελείται. Ο όρος γενικότερα αναφέρεται στη διάταξη στο χώρο ενός συνόλου πρωτεϊνικών μορίων.

2.3 Βασικές Βιολογικές Λειτουργίες

Το 1958 διαπιστώθηκε ότι το μόριο του DNA χρησιμεύει σαν καλούπι για τον διπλασιασμό του, καθώς επίσης και για τη δημιουργία του mRNA, το οποίο μεταφέρει τις γενετικές πληροφορίες για να κατευθύνει τη σύνθεση των πρωτεϊνών.

Η διαπίστωση αυτή είναι γνωστή σαν κεντρικό δόγμα της Βιολογίας (Σχήμα 2.7).

Σχήμα 2.7: Το κεντρικό δόγμα της βιολογίας

Παρατηρούμε ότι το βέλος στο σχήμα 2.7 (δηλαδή η γενετική πληροφορία) πηγαίνει μόνο προς μία κατεύθυνση. Η κύρια εξαίρεση στο κεντρικό δόγμα της Βιολογίας είναι μια διαδικασία γνωστή σαν αντίστροφη μεταγραφή, κατά την οποία κωδικοποιημένη πληροφορία που υπάρχει στο RNA ορισμένων ιών μπορεί να μεταγραφεί σε DNA.

Σύμφωνα με το κεντρικό δόγμα της Βιολογίας η σύνθεση των πρωτεϊνών γίνεται σε δύο στάδια:

1) **Δημιουργία του αγγελιοφόρου (mRNA) που είναι συμπληρωματικό ενός τμήματος του DNA με μια διαδικασία που ονομάζεται μεταγραφή (transcription).**

2) **Το mRNA κινείται προς το κυτταρόπλασμα (για τους ευκαρυωτικούς οργανισμούς) όπου μεταφράζεται σε μια συγκεκριμένη αλληλουχία αμινοξέων. Η διαδικασία αυτή ονομάζεται μετάφραση (translation).**

Σε αντίθεση με την αντιγραφή του DNA, η οποία συμβαίνει μια μόνο φορά κατά τον κύκλο ζωής ενός κυττάρου, η μεταγραφή και η μετάφραση είναι φαινόμενα που επαναλαμβάνονται αδιάκοπα.

Το κεντρικό δόγμα καθορίζει ότι η αλληλουχία των νουκλεοτιδίων στο DNA καθώς και στο συμπληρωματικό αντίγραφο mRNA πρέπει με κάποιο τρόπο να κατευθύνει τη σωστή σειρά τοποθέτησης των αμινοξέων στην κατασκευή της πρωτεΐνης. Το μεγάλο ερώτημα ήταν με ποιο τρόπο αυτή η αλληλουχία των νουκλεοτιδίων κωδικοποιούσε για την αλληλουχία των αμινοξέων. Είναι γνωστό ότι υπάρχουν τέσσερις διαφορετικές βάσεις αλλά είκοσι διαφορετικά αμινοξέα (Πίνακας 2.2).

Πίνακας 2.2: Τα 20 διαφορετικά αμινοξέα

Αμινοξύ	**Συμβολισμός Τριών Γραμμάτων**	**Συμβολισμός Ενός Γράμματος**
Αλανίνη	Ala	A
Κυστεΐνη	Cys	C
Ασπαραγινικό οξύ	Asp	D
Γλουταμινικό οξύ	Glu	E
Φαινυλαλανίνη	Phe	F
Γλυκίνη	Gly	G
Ιστιδίνη	His	H
Ισολευκίνη	Ile	I
Λυσίνη	Lys	K
Λευκίνη	Leu	L
Μεθειονίνη	Met	M
Ασπαραγίνη	Asn	N
Προλίνη	Pro	P
Γλουταμίνη	Gln	Q
Αργινίνη	Arg	R
Σερίνη	Ser	S
Θρεονίνη	Thr	T
Βαλίνη	Val	V
Τρυπτοφάνη	Trp	W
Τυροσίνη	Tyr	Y

Γίνεται λοιπόν φανερό ότι η κωδική λέξη για την κωδικοποίηση κάθε αμινοξέος δεν θα μπορούσε να είναι ούτε μία μόνο ούτε δύο αζωτούχες βάσεις. Εάν ήταν **μία** βάση π.χ. A ή T, μόνο τέσσερα διαφορετικά αμινοξέα θα μπορούσαν να κωδικοποιηθούν. Εάν ήταν **δύο** βάσεις π.χ. AT ή AC μόνο δεκαέξι διαφορετικά αμινοξέα θα μπορούσαν να κωδικοποιηθούν *(4^2 διαφορετικοί συνδυασμοί)*. Εάν όμως η κωδικοποίηση γινόταν με συνδυασμό **τριών** αζωτούχων ενώσεων όπως π.χ. ACT, τότε θα υπήρχαν 64 πιθανά αμινοξέα που θα μπορούσαν να κωδικοποιηθούν από το συνδυασμό των 4 αζωτούχων βάσεων ανά τρεις *(4^3 διαφορετικοί συνδυασμοί)*.

Τα είκοσι, επομένως, αμινοξέα που χρησιμοποιούνται στην κατασκευή των πρωτεϊνών θα μπορούσαν εύκολα να κωδικοποιηθούν με τη χρήση ενός τέτοιου κώδικα βασισμένου σε τριπλέτες βάσεων (triplets).

Μια προσεκτική εξέταση του πίνακα των τριπλετών (κωδικωνίων - codons) επιτρέπει να εξαγάγουμε τα ακόλουθα σημαντικά συμπεράσματα που σχετίζονται με το γενετικό κώδικα:

1. **Ο γενετικός κώδικας είναι εκφυλισμένος** (πλεοναστικός), δηλαδή, πολλές τριπλέτες μπορούν να κωδικοποιήσουν για το ίδιο αμινοξύ. Η λευκίνη, λόγου χάρη, κωδικοποιείται από έξι τριπλέτες: (UUA, UUG, CUU, CUC, CUA, CUG).
2. **Ο γενετικός κώδικας έχει μία τριπλέτα έναρξης AUG και τρεις τριπλέτες λήξης** (UUA, UAG, UGA) που είναι γνωστές και σαν τριπλέτες χωρίς νόημα (nonsense codons).
3. **Η τρίτη βάση είναι λιγότερο σημαντική απ' ό,τι οι δύο πρώτες στον καθορισμό του αμινοξέος που θα κωδικοποιηθεί.** Αυτή η δυνατότητα στη θέση της τρίτης βάσης είναι γνωστή σαν "υπόθεση ταλαντεύσεως".
4. **Ο γενετικός κώδικας που περιγράφηκε έχει σχεδόν παγκόσμια εφαρμογή στους ζωντανούς οργανισμούς.** Κάτι τέτοιο μπορεί να καταδειχτεί στο εργαστήριο παράγοντας τεχνητά πρωτεϊνικά προϊόντα σύμφωνα με γενετικές πληροφορίες από διάφορα μόρια mRNA εντελώς διαφορετικών ειδών φυτών και ζώων. Αυτό αποδεικνύει ότι ο γενετικός κώδικας έχει τις ρίζες του στους πρώτους οργανισμούς πάνω στον πλανήτη και παρέμεινε αναλλοίωτος στην πορεία των αιώνων (Πίνακας 2.3).

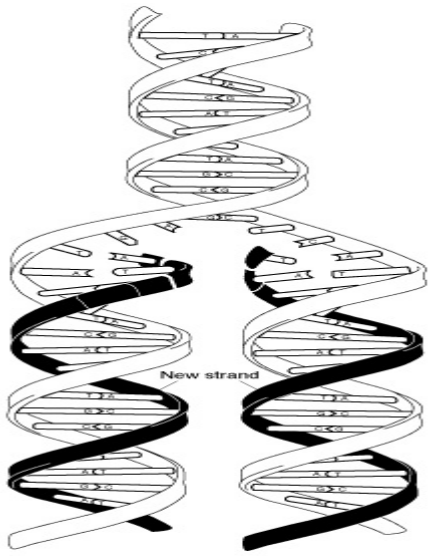
Πίνακας 2.3: Ο Γενετικός κώδικας

Πρώτη βάση	Δεύτερη βάση				Τρίτη βάση
	U	C	A	G	
U	UUU Φαινυλαλανίνη	UCU Σερίνη	UAU Τυροσίνη	UGU Κυστεΐνη	U
	UUG Φαινυλαλανίνη	UCC Σερίνη	UAC Τυροσίνη	UGC Κυστεΐνη	C
	UUA Λευκίνη	UCA Σερίνη	UAA Λήξη	UGA Λήξη	A
	UUG Λευκίνη	UCG Σερίνη	UAG Λήξη	UGG Τρυπτοφάνη	G
C	CUU Λευκίνη	CCU Προλίνη	CAU Ιστιδίνη	CGU Αργινίνη	U
	CUC Λευκίνη	CCC Προλίνη	CAC Ιστιδίνη	CGC Αργινίνη	C
	CUA Λευκίνη	CCA Προλίνη	CAA Γλουταμίνη	CGA Αργινίνη	A
	CUG Λευκίνη	CCG Προλίνη	CAG Γλουταμίνη	CGG Αργινίνη	G
A	AUU Ισολευκίνη	ACU Θρεονίνη	AAU Ασπαραγγίνη	AGU Σερίνη	U
	AUC Ισολευκίνη	ACC Θρεονίνη	AAC Ασπαραγγίνη	AGC Σερίνη	C
	AUA Ισολευκίνη	ACA Θρεονίνη	AAA Λυσίνη	AGA Αργινίνη	A
	AUG (Έναρξη) Μεθειονίνη	ACG Θρεονίνη	AAG Λυσίνη	AGG Αργινίνη	G
G	GUU Βαλίνη	GCU Αλανίνη	GAU Ασπαρτικό οξύ	GGU Γλυκίνη	U
	GUC Βαλίνη	GCC Αλανίνη	GAC Ασπαρτικό οξύ	GGC Γλυκίνη	C
	GUA Βαλίνη	GCA Αλανίνη	GAA Γλουταμινικό οξύ	GGA Γλυκίνη	A
	GUG Βαλίνη	GCG Αλανίνη	GAG Γλουταμινικό οξύ	GGC Γλυκίνη	G

2.3.1 Η Διαδικασία της Αντιγραφής του DNA (DNA replication)

Μία από τις πιο βασικές και πλέον χαρακτηριστικές ιδιότητες του DNA είναι η δυνατότητα αντιγραφής του. Χωρίς αυτήν την δυνατότητα ο πολλαπλασιασμός των κυττάρων στους οργανισμούς θα πραγματοποιούνταν χωρίς το νέο κύτταρο να περιέχει γενετικό υλικό, χωρίς δηλαδή να υπάρχει η δυνατότητα για αυτό να καθορίσει τις λειτουργίες του και πολύ περισσότερο να πολλαπλασιαστεί και αυτό με τη σειρά του.

Η διπλή έλικα του DNA κατά την λειτουργία αυτή «ανοίγει» χάρη στο σπάσιμο των ασθενών υδρογονικών δεσμών μεταξύ των δυο συμπληρωματικών αλυσίδων. Ένα ένζυμο, η **ελικάση**, χρησιμοποιείται για να ξετυλιχτεί η διπλή έλικα του μορίου. Με τον τρόπο αυτό σχηματίζονται δυο ξεχωριστές πολυνουκλεοτιδικές αλυσίδες που χρησιμεύουν η καθεμία ως καλούπι (εκμαγείο, όπως χαρακτηριστικά αποκαλείται από πολλούς βιολόγους) για το σχηματισμό μιας συμπληρωματικής της. Καταλήγουμε, δηλαδή, στο σχηματισμό δυο δίκλωνων μορίων DNA, των οποίων μόνο μια αλυσίδα είναι καινούρια, ενώ η άλλη προέρχεται από το πατρικό μόριο DNA (καλούπι). Για το λόγο αυτό ο συγκεκριμένος τρόπος αντιγραφής του DNA ονομάζεται ημισυντηρητικός τρόπος διπλασιασμού (Σχήμα 2.8).

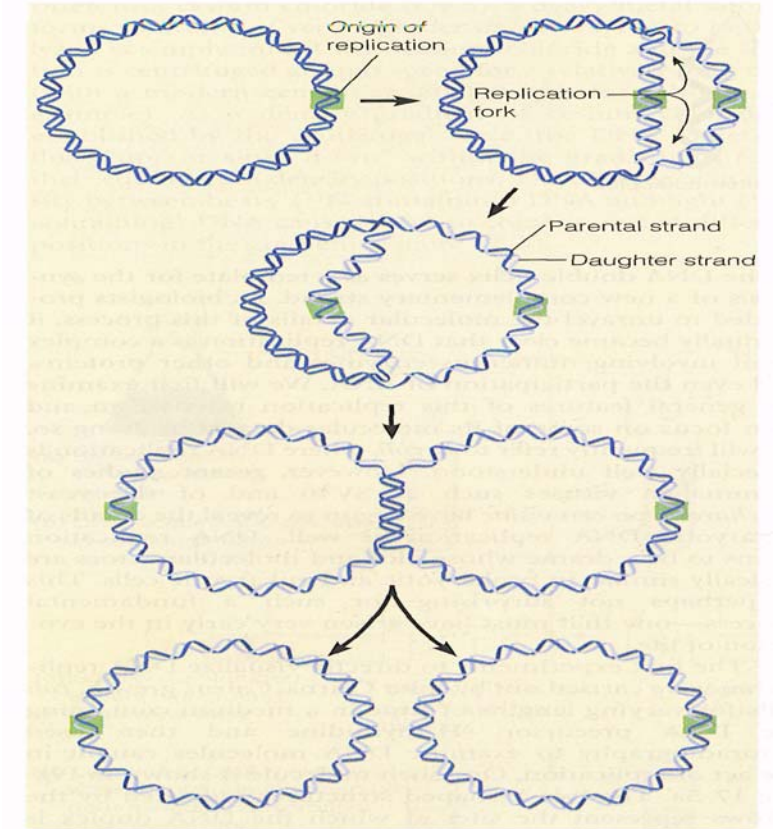


Σχήμα 2. 8: Η αντιγραφή του DNA

Ειδικά για τα προκαρυωτικά κύτταρα, δεδομένου ότι το μόριο του DNA είναι κυκλικό, η αντιγραφή του DNA έχει ένα συγκεκριμένο σημείο εκκίνησης ή, θα λέγαμε, μια συγκεκριμένη ακολουθία DNA (**origin of replication**). Με βάση τα πρώτα πειράματα που πραγματοποιήθηκαν για να μελετηθεί ο τρόπος αντιγραφής του DNA, παρατηρήθηκε ότι από συγκεκριμένα σημεία των μορίων DNA δημιουργούνταν **διχάλες αντιγραφής (replication forks)**, οι οποίες κινούνταν κατά μήκος της ακολουθίας του DNA, «ξετυλίγοντας» και αντιγράφοντας την ακολουθία της αλυσίδας του DNA, σε αντίθετη κατεύθυνση η μία από την άλλη. Ειδικά για το βακτήριο Escherichia Coli (E.Coli), αλλά και για άλλους οργανισμούς με κυκλικό DNA, η διαδικασία της αντιγραφής του DNA ονομάζεται **«αντιγραφή θήτα» (theta replication)**, καθώς, όπως φαίνεται και στο σχήμα, κατά την διάρκεια της αντιγραφής δημιουργούνται μορφές παρόμοιες με το σχήμα του ελληνικού γράμματος θ (Σχήμα 2.9).

Η αντίστοιχη διαδικασία αντιγραφής του DNA στα ευκαρυωτικά κύτταρα είναι πιο περίπλοκη σε σχέση με αυτή των προκαρυωτικών κυττάρων. Σε αυτήν την περίπτωση το σημείο εκκίνησης της αντιγραφής του DNA δεν είναι συγκεκριμένο και δημιουργεί πολλές μονάδες αντιγραφής, τα **ρεπλικόνια (replicons)**. Τα ρεπλικόνια καθορίζουν τα σημεία στα οποία δημιουργούνται και σταδιακά αναπτύσσονται οι **φυσαλίδες αντιγραφής (replication bubbles),** οι οποίες συγχωνεύονται, με αποτέλεσμα να δημιουργείται η νέα ακολουθία DNA. Προφανώς η διαδικασία αυτή είναι εξαιρετικά περίπλοκη. Είναι χαρακτηριστικό ότι στην διαδικασία της αντιγραφής του DNA στα ευκαρυωτικά κύτταρα εμπλέκονται 30

ένζυμα, αλλά η παραπάνω περιγραφή συνοψίζει την συγκεκριμένη διαδικασία.



Σχήμα 2.9: Η διαδικασία αντιγραφής κυκλικής μορφής μορίων DNA, όπως στην E.Coli, εξαιτίας του σχήματος που διαμορφώνεται, ονομάζεται και αντιγραφή θήτα.

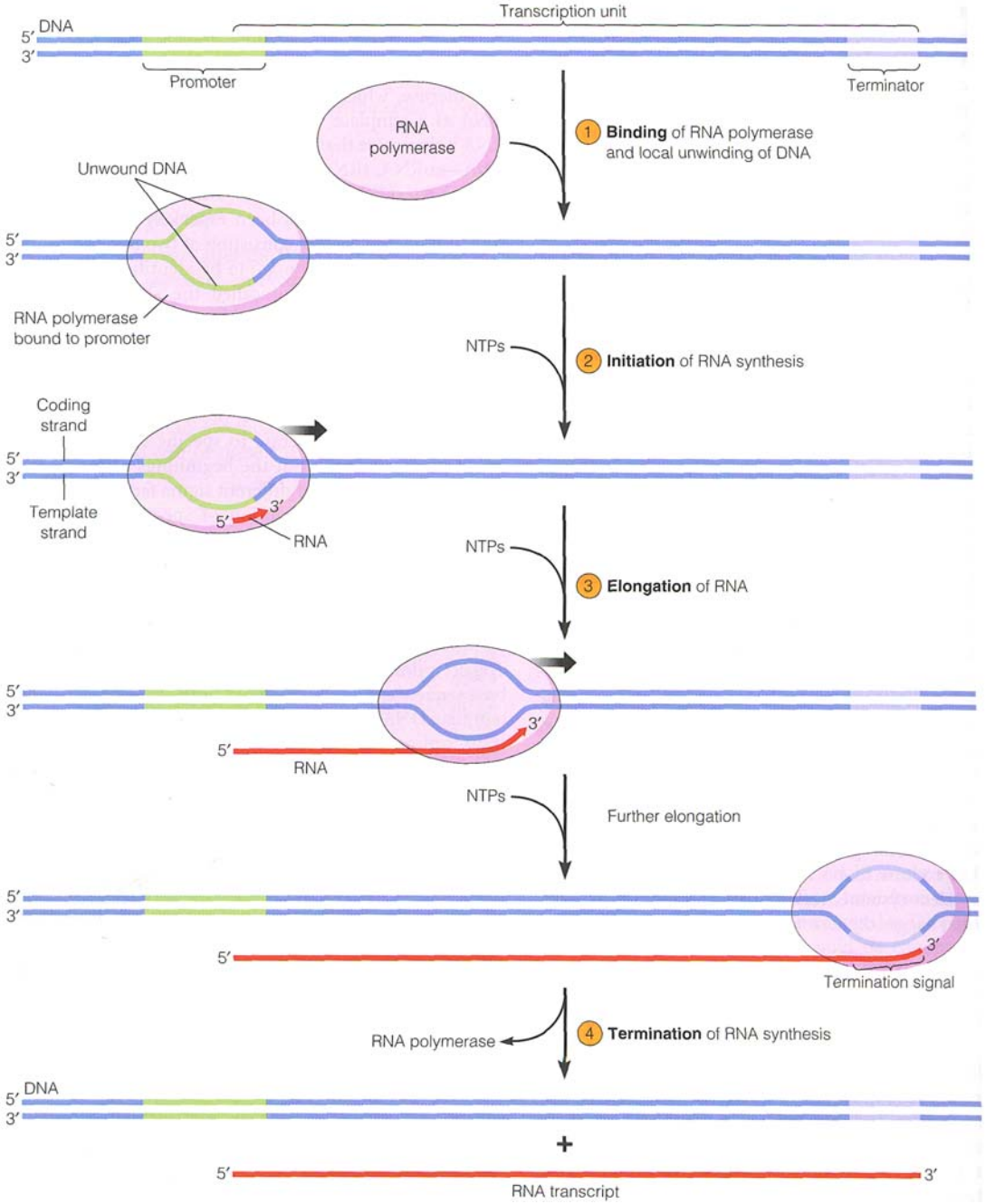
2.3.2 Η Διαδικασία της Μεταγραφής του DNA (DNA transcription)

Απαραίτητο ενδιάμεσο στάδιο για την πρωτεϊνοσύνθεση είναι η δημιουργία των διαφόρων ειδών RNA. **Μεταγραφή είναι η σύνθεση ενός RNA μορίου, η ακολουθία βάσεων του οποίου είναι συμπληρωματική στην ακολουθία βάσεων του DNA**. Κατά τη μεταγραφή το ένζυμο RNA-πολυμεράση έχει καταλυτικό ρόλο αφού χρησιμοποιεί το DNA σαν βάση για τη δημιουργία του RNA. Η μεταγραφή τόσο στα προκαρυωτικά, όσο και στα ευκαρυωτικά κύτταρα περιλαμβάνει τέσσερα στάδια (Σχήμα 2.10):

1. την **δέσμευση (binding)** της RNA-πολυμεράσης σε συγκεκριμένες υποακολουθίες της ακολουθίας του DNA. **Οι υποακολουθίες αυτές ονομάζονται υποκινητές (promoters)**.
2. την **έναρξη (initiation)** της δημιουργίας της RNA αλυσίδας.
3. την **επιμήκυνση (elongation)** της RNA αλυσίδας.
4. τον **τερματισμό (termination)** της επιμήκυνσης της RNA αλυσίδας, με τη λήψη του μηνύματος λήξης από συγκεκριμένες ακολουθίες βάσεων (terminators).

Για τα προκαρυωτικά κύτταρα, κατά το πρώτο στάδιο της μεταγραφής η RNA-πολυμεράση δεσμεύεται στην περιοχή του υποκινητή. Οι περιοχές των υποκινητών είναι συγκεκριμένες ακολουθίες νουκλεοτιδίων, οι οποίες καθορίζουν από ποιο σημείο ξεκινάει η δημιουργία του RNA και ποια από τις δυο αλυσίδες του DNA θα χρησιμεύσει σαν «καλούπι» για τη σύνθεση του RNA. Κάθε μονάδα μεταγραφής (transcription unit) έχει έναν υποκινητή στην αρχή της ακολουθίας που

πρόκειται να μεταγραφεί. Στο σχήμα 2.10 παρατηρούμε την περιγραφείσα διαδικασία.

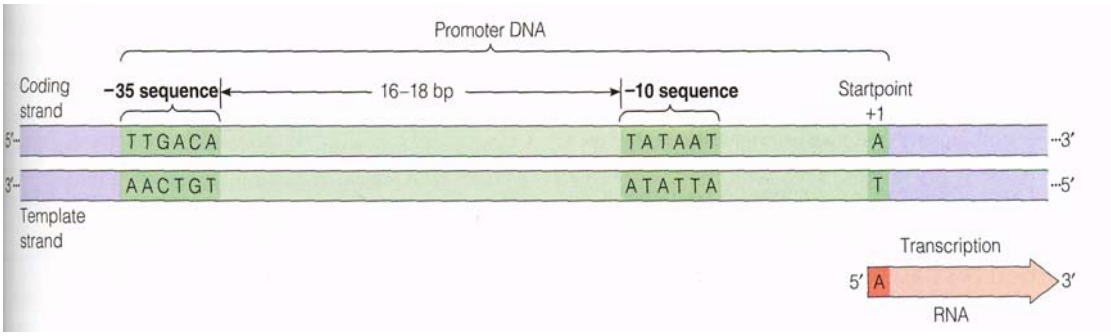


Σχήμα 2.10: Η διαδικασία της μεταγραφής του DNA για έναν προκαρυωτικό οργανισμό

Συμβατικά, οι ακολουθίες υποκινητών περιγράφονται στην κατεύθυνση **3'→5' (upstream)** για την αλυσίδα που θα μεταγραφεί (αλυσίδα με νόημα ή coding strand). Αντίστοιχα, η αλυσίδα του DNA που δεν θα μεταγραφεί (αλυσίδα χωρίς νόημα ή template strand) έχει, λόγω συμπληρωματικότητας των δυο αλυσίδων του DNA, 5'→3' κατεύθυνση (downstream). Η δέσμευση της RNA-πολυμεράσης στον υποκινητή προκαλεί το «ξετύλιγμα» της διπλής έλικας του DNA στην περιοχή αυτή (με μέσο μήκος 15-18 bp).

Μετά από μια σειρά ερευνών και καθορισμού των ακολουθιών των υποκινητών με κύρια μέθοδο αυτή του DNA footprinting, παρατηρήθηκε ότι οι υποκινητές διαφέρουν χαρακτηριστικά μεταξύ τους. Εύλογα, το ερώτημα που τέθηκε ήταν με ποιόν τρόπο η RNA-πολυμεράση καταφέρνει να αναγνωρίσει το σύνολο

αυτών των διαφορετικών μεταξύ τους ακολουθιών. Η πιο πρόσφατη απάντηση στο ερώτημα αυτό είναι ότι η αναγνώριση των ακολουθιών αυτών γίνεται όχι σε όλο το μήκος του υποκινητή, αλλά σε πολύ μικρές στο μήκος υποακολουθίες του. Οι υποακολουθίες αυτές βρίσκονται σε συγκεκριμένες θέσεις σε κάθε περιοχή υποκινητή. Ένας τυπικός προκαρυωτικός υποκινητής παρουσιάζεται στο Σχήμα 2.11



Σχήμα 2.11: Η δομή ενός τυπικού προκαρυωτικού υποκινητή. Εμφανίζονται οι περιοχές -35, -10 καθώς και το σημείο εκκίνησης της μεταγραφής (Startpoint)

Όπως παρατηρούμε, υπάρχουν δυο μικρές υποακολουθίες, οι οποίες συμβολίζονται σαν περιοχή -35 και περιοχή -10. Σαν σημείο εκκίνησης (startpoint) ορίζεται το σημείο +1 (στις περισσότερες περιπτώσεις στην βιβλιογραφία δεν ορίζεται σημείο 0). Περίπου 10 θέσεις από το σημείο εκκίνησης αντίθετα στην βασική κατεύθυνση (δηλ. με κατεύθυνση 3'→5', upstream) υπάρχει ακολουθία 6 νουκλεοτιδίων (TATAAT), η οποία ονομάζεται ακολουθία -10 ή κουτί Pribnow (Pribnow box). Αντίστοιχα, περίπου στην θέση -35 υπάρχει άλλη μια υποακολουθία 6 νουκλεοτιδίων (TTGACA), η οποία ονομάζεται ακολουθία -35.Οι υποακολουθίες -35 και -10 δεν είναι σε όλους τους προκαρυωτικούς οργανισμούς οι ίδιες, αλλά ακόμα και στο γονιδίωμα ενός οργανισμού υπάρχουν διαφοροποιήσεις. Για παράδειγμα, στην Escherichia Coli η ακολουθία -10 είναι για την ενεργοποίηση κάποιων γονιδίων TATGAT ενώ για την ενεργοποίηση μιας άλλης ομάδας γονιδίων είναι TATGTT. Παρατηρούμε ότι στο σχήμα η υποακολουθία στην περιοχή -10 είναι διαφορετική και από τις δυο υποακολουθίες που αναφέραμε παραπάνω. **Η συγκεκριμένη υποακολουθία που εμφανίζεται στο σχήμα (TATAAT) αποτελεί την «πλειοψηφική ακολουθία» (consensus sequence) για την περιοχή -10, ενώ αντίστοιχα η «πλειοψηφική ακολουθία» για την περιοχή -35 είναι η TTGACA**.Στο ενδιάμεσο των περιοχών -35, -10 και της περιοχής που αρχίζει η μεταγραφή υπάρχει διάστημα που κυμαίνεται, όπως βλέπουμε, μεταξύ 16 και 17 bp για το διάστημα μεταξύ των περιοχών -35 και -10 και μεταξύ 6 και 7 bp για το διάστημα μεταξύ της περιοχής -10 και της περιοχής εκκίνησης της μεταγραφής. Το πλήθος των νουκλεοτιδίων που παρεμβάλλονται μεταξύ του σημείου εκκίνησης, της περιοχής -10 και της περιοχής -35 καθορίζει τον ρόλο του υποκινητή. Από την άλλη πλευρά δεν μας ενδιαφέρει η ταυτότητα των νουκλεοτιδίων που βρίσκονται στις θέσεις αυτές.

Πρέπει να σημειωθεί ότι η έννοια του consensus box είναι μια στατιστική δημιουργία. Επομένως, μας παρέχεται απλώς μια ενδεικτική εικόνα του ποια χαρακτηριστικά μπορεί να έχει μια ακολουθία που μας δίνεται. Η πραγματική βαρύτητα και σημασία κάθε βάσης σε μια ακολουθία μπορεί να δοθεί με γενετική, φυσική και βιοχημική ανάλυση. Μια γενική προσέγγιση του προβλήματος μπορεί να παρομοιαστεί με το πρόβλημα αναγνώρισης των φοιτητών της σχολής ΗΜΜΥ, οι οποίοι έχουν δηλώσει ένα συγκεκριμένο μάθημα με βάση τον αριθμό μητρώου τους. Έστω ότι 154 φοιτητές έχουν δηλώσει ένα μάθημα επιλογής του $7^{ου}$ εξαμήνου κατά την διάρκεια του ακαδημαϊκού έτους 2004. Δεδομένου ότι οι αριθμοί μητρώων των

φοιτητών του ΕΜΠ είναι οκταψήφιοι και τα τρία πρώτα ψηφία δηλώνουν την σχολή στην οποία φοιτούν, ενώ τα επόμενα δυο ψηφία δηλώνουν το έτος εισαγωγής τους και τα επόμενα την σειρά επιτυχίας τους στη σχολή, έχουμε τα παρακάτω αποτελέσματα που εμφανίζονται στον πίνακα 2.4.

Πίνακας 2.4

Αριθμός εμφανίσεων ψηφίων στις θέσεις 4 ως 8					
9	34	22		12	12
8		7		12	17
7			17	**24**	16
6			12	20	**23**
5				12	19
4				**24**	**23**
3				13	11
2		5	**47**	14	9
1		**81**	36	11	13
0	**120**	34	42	12	11

Στον πίνακα οι αριθμοί με έντονα γράμματα δείχνουν τα ψηφία που εμφανίζονται συχνότερα στις συγκεκριμένες θέσεις. Για παράδειγμα στην θέση 4 εμφανίζεται 120 φορές το 0. Για τις θέσεις 7 και 8 θα πρέπει να δημιουργήσουμε έναν κανόνα. Καθώς κάθε νούμερο θα πρέπει να εμφανίζεται κατά μέσον όρο 15.4 φορές (154/10), θεωρούμε ότι όσα ψηφία εμφανίζονται 1.5 φορά επί τον μέσο όρο (23) είναι χαρακτηριστικά για να υπάρχουν στο consensus id. Το sequence id θα είναι επομένως συνολικά το εξής (για τις θέσεις 4 ως 8):

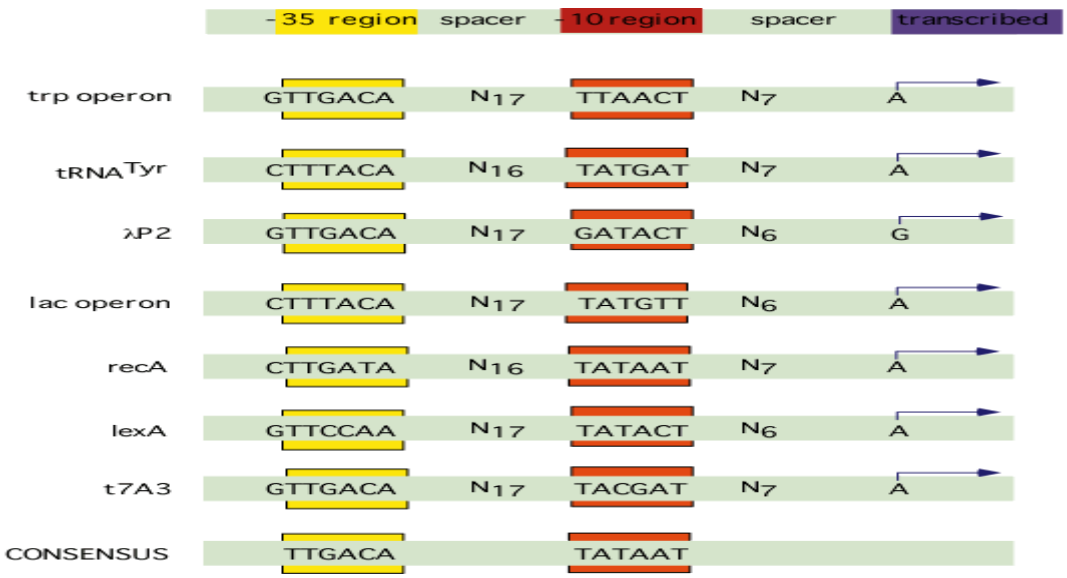
$$\mathbf{0_{120}1_{81}2_{47}(7/4)_{24}(6/4)_{23}}$$

ή απλούστερα

$$\mathbf{012(7/4)(6/4)}$$

Προφανώς κανένας από τους φοιτητές δεν έχει τέτοιον αριθμό μητρώου. Παρ' όλα αυτά μας δίνονται χρήσιμες πληροφορίες, ειδικά για τις θέσεις 4 και 5 του αριθμού μητρώου. Από την άλλη πλευρά αποκλείονται μεγάλα κομμάτια φοιτητών που έχουν, για παράδειγμα, δηλώσει το μάθημα και ο αριθμός μητρώου τους δεν ξεκινάει από μηδέν.

Αντίστοιχα φαινόμενα έχουμε και κατά την δημιουργία consensus box στις ακολουθίες DNA. Το consensus box για τους υποκινητές της E.Coli, συγκρινόμενο με πραγματικές ακολουθίες υποκινητών παρουσιάζονται στο σχήμα 2.12.



Σχήμα 2.12: Το consensus box για το βακτήριο της E.Coli και ακολουθίες πραγματικών υποκινητών. Εμφανίζονται επίσης και τα κενά μεταξύ των περιοχών -35, -10 και -10, TSS (Transcription Start Site)

Αμέσως μετά τον εντοπισμό της ακολουθίας του υποκινητή, το «ξετύλιγμα» της διπλής έλικας του DNA και το δέσιμο της RNA-πολυμεράσης γύρω από τον υποκινητή λαμβάνει χώρα η έναρξη (initiation) της σύνθεσης του RNA. Η RNA-πολυμεράση προωθείται στην ακολουθία του υποκινητή μέχρι η υπο σχηματισμό νέα αλυσίδα RNA αποκτήσει μήκος περίπου 9 νουκλεοτιδίων.

Η επιμήκυνση (elongation) της αλυσίδας συνεχίζεται να γίνεται κατά την κατεύθυνση 5'→3'. Καθώς η αλυσίδα του mRNA επιμηκύνεται, αποσυνδέεται από την αλυσίδα του DNA και το μόριο του DNA ξανατυλίγεται. Το μόριο RNA που φτιάχνεται με τον τρόπο αυτό είναι συμπληρωματικό με την αλυσίδα από την οποία μεταγράφηκε, ενώ είναι ταυτόχρονα πανομοιότυπο (με εξαίρεση ότι η θυμίνη έχει αντικατασταθεί από την ουρακίλη) με την αλυσίδα του DNA που δεν μεταγράφηκε (αλυσίδα χωρίς νόημα ή template strand). Σημειώνεται ότι, όπως και στη σύνθεση του DNA, χρησιμοποιούνται τριφωσφορικά νουκλεοτίδια, των οποίων οι δύο φωσφορικές ομάδες απομακρύνονται, ώστε η ενέργεια που εκλύεται να χρησιμεύσει για τη δράση της RNA-πολυμεράσης.

Στη συνέχεια, η RNA πολυμεράση αντιλαμβάνεται ένα μήνυμα τερματισμού από μια αλληλουχία βάσεων που ονομάζεται **απολητκική (terminator)** και με τον τρόπο αυτό το ολοκληρωμένο μόριο mRNA ελευθερώνεται πλήρως από το «καλούπι» του DNA.

Στο σημείο αυτό θα πρέπει να σημειώσουμε ότι παρά την ομοιότητα που παρουσιάζουν στα βήματα κατά την διάρκεια της μεταγραφής τα προκαρυωτικά και ευκαρυωτικά κύτταρα, δεν υπάρχει πλήρης ταύτιση της διαδικασίας μεταξύ τους. Πιο συγκεκριμένα, στα ευκαρυωτικά κύτταρα σε σύγκριση με τα προκαρυωτικά παρατηρούνται:

1. 3 διαφορετικά είδη RNA-πολυμεράσης
2. 3 διαφορετικές τάξεις υποκινητών, μια για κάθε διαφορετικό είδος RNA-πολυμεράσης
3. δέσμευση της RNA-πολυμεράσης στο DNA απαιτεί την συμμετοχή πρόσθετων πρωτεϊνών-ενζύμων
4. αλληλεπιδράσεις μεταξύ των ήδη υπαρχόντων πρωτεϊνών, οι οποίες είναι πολύ σημαντικές κατά το πρώτο στάδιο της ευκαρυωτικής μεταγραφής
5. διαφορετικός τρόπο καθορισμού του σημείου τερματισμού της μεταγραφής
6. χημική διαφοροποίηση των νεοσχηματισμένων μορίων RNA τόσο κατά την διάρκεια, όσο και μετά την μεταγραφή.

2.3.3 Η Διαδικασία της Μετάφρασης του DNA (DNA translation)

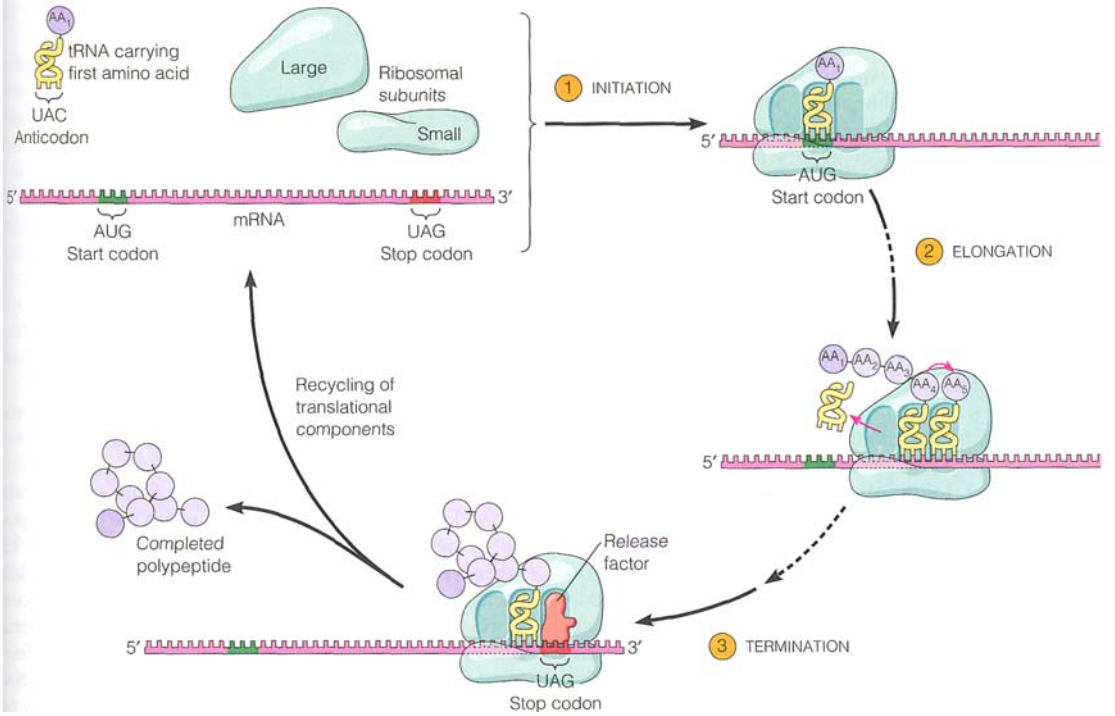
Η διαδικασία της μετάφρασης πρέπει να γίνει με έναν πολύ δομημένο και τακτικό τρόπο ώστε να διασφαλιστεί η αποφυγή λαθών στην τελική αλληλουχία αμινοξέων που θα παραχθεί, δηλαδή στην πρωτεΐνη. Τα σημαντικότερα γεγονότα αυτής της διαδικασίας συνοψίζονται με τον ακόλουθο τρόπο[7] (Σχήμα 2.13):

1. Έναρξη της μετάφρασης γίνεται, όταν η μικρή υπομονάδα του ριβοσώματος ενώνεται με το mRNA στην περιοχή του κωδικονίου έναρξης (start codon) που είναι το AUG. Το πρώτο tRNA, που ονομάζεται **εναρκτής (initiation factor),** ζευγαρώνει με το κωδικόνιο χάρη στη συμπληρωματικότητα των βάσεων μεταξύ του αντικωδικονίου του και του αντίστοιχου κωδικονίου του mRNA. Αυτό το αρχικό tRNA μεταφέρει μια τροποποιημένη μορφή του αμινοξέος μεθειονίνη, τη N φορμυλμεθειονίνη ή fmet που αργότερα μπορεί να αποχωριστεί. Ο συνδυασμός της μικρής ριβοσωμικής υπομονάδας του mRNA και του εναρκτή tRNA ονομάζεται

σύμπλεγμα έναρξης. Η μεγάλη ριβοσωμική υπομονάδα συνδέεται με το tRNA στην P θέση (πεπτιδική θέση).

2. Αρχίζει η επιμήκυνση της αλυσίδας. Ένα δεύτερο σύμπλεγμα αμινοξέος - tRNA συνδέεται στη δεύτερη θέση A (αμινοακυλ- θέση) της μεγάλης υπομονάδας του ριβοσώματος με τον ίδιο τρόπο (συμπληρωματικότητα κωδικονίου - αντικωδικονίου). Όταν συμπληρωθούν και οι δύο θέσεις του ριβοσώματος (A και P), το ένζυμο πεπτιδική τρανσφεράση που αποτελεί τμήμα της μεγάλης ριβοσωμικής υπομονάδας, δημιουργεί έναν πεπτιδικό δεσμό μεταξύ των δύο αμινοξέων. Ταυτόχρονα ο δεσμός μεταξύ του πρώτου αμινοξέος και του tRNA του σπάζει και το tRNA ελευθερώνεται προσφέροντας ενέργεια για τη δημιουργία του πεπτιδικού δεσμού. Στη συνέχεια, το ριβόσωμα μετακινείται κατά μήκος του μορίου mRNA έτσι ώστε το δεύτερο tRNA που τώρα έχει το διπεπτίδιο προσκολλημένο επάνω του να καταλάβει την P θέση ελευθερώνοντας με τον τρόπο αυτό τη θέση A. Το τρίτο σύμπλεγμα αμινοξύ tRNA κινείται προς την ελεύθερη θέση A και η διαδικασία της δημιουργίας πεπτιδικού δεσμού επαναλαμβάνεται. Καθώς το ριβόσωμα κινείται διαδοχικά σε όλο το μήκος της αλυσίδας του mRNA, κωδικόνιο προς κωδικόνιο, προσθέτει με τον ίδιο τρόπο ένα καινούριο σύμπλεγμα αμινοξέος-tRNA στη θέση A. Καθώς το άκρο του μορίου mRNA, από το οποίο ξεκίνησε η μετάφραση, ελευθερώνεται, ένα άλλο ριβόσωμα μπορεί να προσδεθεί πάνω του και να αρχίσει μετάφραση πριν ακόμα το πρώτο ριβόσωμα ολοκληρώσει τη σύνθεση του πολυπεπτιδίου. Το σύνολο των ριβοσωμάτων που διαβάζουν το ίδιο μόριο mRNA ονομάζεται **πολυριβόσωμα.**

3. Ο τερματισμός της αλυσίδας επέρχεται όταν το ριβόσωμα αντιληφθεί ένα σήμα λήξης με τη μορφή ενός κωδικονίου λήξης (UAG, UAA και UGA). Κανένα tRNA δεν θα τοποθετηθεί στην περιοχή A, μια που κανένα tRNA δεν φέρνει αντικωδικόνιο συμπληρωματικό με αυτές τις αλληλουχίες. Το πολυπεπτίδιο βγαίνει από το ριβόσωμα και οι δύο ριβοσωμικές υπομονάδες διαχωρίζονται.



Σχήμα 2.13: Τα τρία βήματα με τα οποία ολοκληρώνεται η μετάφραση σε ένα κύτταρο

Με τον τρόπο αυτό δημιουργούνται οι πρωτεΐνες, οι οποίες θα παίξουν σημαντικό ρόλο σε διάφορες λειτουργίες του οργανισμού:

- **Ενζυμική κατάλυση.** Σχεδόν όλες οι χημικές αντιδράσεις στα βιολογικά συστήματα καταλύονται από ειδικά μακρομόρια που ονομάζονται ένζυμα. Μερικές από τις αντιδράσεις αυτές είναι αρκετά απλές, όπως π.χ. η ενυδάτωση του CO_2. Άλλες, όπως η αντιγραφή ενός ολοκλήρου χρωμοσώματος, είναι αρκετά πολύπλοκες. Σχεδόν όλα τα ένζυμα επιδεικνύουν τεράστια καταλυτική ικανότητα. Συνήθως, αυξάνουν την ταχύτητα των αντιδράσεων τουλάχιστον 10^6 φορές. Περισσότερα από 100 ένζυμα έχουν ταυτιστεί και πολύ περισσότερα έχουν κρυσταλλωθεί. Το αξιοσημείωτο είναι ότι όλα τα γνωστά ένζυμα είναι πρωτεΐνες. Έτσι, οι πρωτεΐνες διαδραματίζουν τον μοναδικό ρόλο του καθορισμού της πορείας των χημικών αντιδράσεων που πραγματοποιούνται στα βιολογικά συστήματα.
- **Μεταφορά και αποθήκευση.** Πολλά μικρά μόρια και ιόντα μεταφέρονται από ειδικές πρωτεΐνες. Για παράδειγμα, η αιμοσφαιρίνη μεταφέρει οξυγόνο στα ερυθροκύτταρα, ενώ η μυοσφαιρίνη στους μυς. Σίδηρος μεταφέρεται στο πλάσμα του αίματος από την τρανσφερρίνη και αποθηκεύεται στο συκώτι ως σύμπλοκο μαζί με τη φερριτίνη, μία διαφορετική πρωτεΐνη.
- **Κίνηση.** Πρωτεΐνες είναι τα κύρια συστατικά των μυών. Η συστολή των μυών επιτυγχάνεται με την ολισθητική κίνηση δύο ειδών πρωτεϊνικών νηματίων. Επίσης, η κίνηση των χρωμοσωμάτων στη μίτωση και η προώθηση των σπερμάτων με τη βοήθεια των μαστιγίων τους επιτυγχάνεται με συσταλτά συγκροτήματα που αποτελούνται από πρωτεΐνες.
- **Μηχανική στήριξη.** Η μεγάλη αντοχή του δέρματος και των οστών οφείλεται στην παρουσία του κολλαγόνου, μιας πρωτεΐνης που σχηματίζει ίνες.
- **Ανοσοπροστασία.** Τα αντισώματα είναι πολύ ειδικές πρωτεΐνες που αναγνωρίζουν και συνενώνονται με ξένες ουσίες όπως οι ιοί, τα βακτήρια και κύτταρα από άλλους οργανισμούς. Ο ζωτικός ρόλος, επομένως, των πρωτεϊνών είναι προφανής.
- **Δημιουργία και μετάδοση νευρικών παλμών.** Η απόκριση των νευρικών κυττάρων σε ειδικά ερεθίσματα υποβοηθείται από πρωτεΐνες-υποδοχείς. Για παράδειγμα, η ροδοψίνη είναι ο φωτοϋποδοχέας-πρωτεΐνη στα ραβδία του αμφιβληστροειδή. Μόρια υποδοχείς που μπορούν να «διεγερθούν» με ειδικά μόρια, όπως η ακετυλοχολίνη, είναι υπεύθυνα για τη μετάδοση των νευρικών παλμών στις συνάψεις, δηλαδή τις επαφές μεταξύ νευρικών κυττάρων.
- **Έλεγχος της ανάπτυξης και διαφοροποίησης.** Η ελεγχόμενη έκφραση των γενετικών πληροφοριών είναι ουσιώδης για την κανονική ανάπτυξη και διαφοροποίηση των κυττάρων. Μόνο ένα μικρό κλάσμα του γονιδιώματος ενός κυττάρου εκφράζεται κάθε φορά.

---

[1] W. Becker, L. Kleinsmith, J. Hardin – The world of the cell – Benjamin Cummings, 2002

[2] Φίλιπ Κουρίλσκι – DNA, Το νήμα της ζωής και οι γενετικές επεμβάσεις – Εκδόσεις Ράππα, 1993

[3] K. P. Talaro, Arthur Talaro – Foundations in microbiology – WCB/McGraw-Hill, 1999

[4] Schulz G.E. and Schirmer R.H. - Principles of Protein Structure – 1979

[5] Α.Θ. Καλοφούτης, Κ.Ε. Σέκερης - Ιατρική Βιοχημεία, Κλινική Προσέγγιση – 2000

[6] Dickerson, R.E. and Geis, I. - The Structure and Action of Proteins – 1969

[7] K.Arms and P.S. Camp – Biology, a journey into life – Sounders College Publishing, 1991

Κεφάλαιο 3

Βιοπληροφορική – Υπολογιστική Βιολογία

3 Βιοπληροφορική-Υπολογιστική Βιολογία

3.1 Εισαγωγή

Η ανακάλυψη του μοντέλου της διπλής έλικας για το DNA το 1953 από τους Watson και Crick αποτέλεσε την βάση για την εντυπωσιακή ανάπτυξη της σύγχρονης μοριακής βιολογίας τα επόμενα 50 χρόνια. Οι ολοένα και περισσότερες ανακαλύψεις στον τρόπο λειτουργίας του DNA και άλλων μακρομοριακών ακολουθιών (RNA, πρωτεΐνες) τα επόμενα χρόνια, έφτασαν στο αποκορύφωμά τους τον Απρίλη του 2003 με την ανακοίνωση της επιτυχούς ολοκλήρωσης αποκωδικοποίησης του ανθρώπινου γονιδιώματος από μια σειρά συνεργαζόμενων ερευνητικών κέντρων σε όλο τον κόσμο, τα οποία είχαν ξεκινήσει αυτό το εγχείρημα (γνωστό σαν Human Genome Project) από το 1990[1]. Τα νέα δεδομένα στον χώρο της σύγχρονης μοριακής βιολογίας δημιούργησαν νέες απαιτήσεις, που ήταν δύσκολο να ικανοποιηθούν, αν η επιστήμη της βιολογίας δεν ερχόταν σε ώσμωση με μια σειρά άλλων επιστημών και γνωστικών αντικειμένων.

Η περίπου παράλληλη χρονικά ανάπτυξη της πληροφορικής τα τελευταία 50 περίπου χρόνια με αυτήν της βιολογίας οδήγησε, σε πολλές περιπτώσεις, σε πρώιμη συνεργασία μεταξύ των δυο αυτών εξαιρετικά διαχωρισμένων μέχρι πρότινος αντικειμένων. Η σταδιακή συστηματοποίηση της συνεργασίας επιστημόνων από τους δυο αυτούς χώρους αποκρυστάλλωσε την τάση για συγχώνευση γνώσεων με σκοπό την εφαρμογή και εξειδίκευση λύσεων από τον κλάδο της πληροφορικής πάνω σε βιολογικά προβλήματα[2]. Η βιοπληροφορική (bioinformatics) και η υπολογιστική βιολογία (computational biology) είναι οι δυο νέοι όροι που συμπυκνώνουν τη νέα πραγματικότητα στον χώρο της βιολογίας και της πληροφορικής.

3.2 Βιοπληροφορική, Υπολογιστική Βιολογία και αντίστοιχοι Τομείς Έρευνας

Η βιοπληροφορική θα μπορούσε να οριστεί ως «…η έρευνα, ανάπτυξη ή εφαρμογή υπολογιστικών εργαλείων και προσεγγίσεων για την επέκταση της χρήσης των βιολογικών δεδομένων καθώς και για τη συλλογή, αποθήκευση, οργάνωση, αρχειοθέτηση, ανάλυση και οπτικοποίηση αυτών των δεδομένων».

Αντίστοιχα, **η υπολογιστική βιολογία θα μπορούσε να οριστεί ως «…η ανάπτυξη και εφαρμογή αναλυτικών και θεωρητικών μεθόδων, μαθηματικών μοντέλων και τεχνικών υπολογιστικών προσομοιώσεων για τη μελέτη βιολογικών συστημάτων»**[3] (Σχήμα3.1).

Bioinformatics- A Definition

Bio-informatics: Bioinformatics is conceptualizing biology in terms of molecules (in the sense of Physical Chemistry) and applying **"informatics techniques"** (derived from disciplines such as applied maths, computer science and statistics) to **understand** and **organise** the **information** associated with these molecules, on a **large scale**. In short bioinformatics is a management information system for molecular biology and has many **practical applications.**
Oxford English Dictionary.

Σχήμα 3.1: Ορισμός της βιοπληροφορικής

Με βάση τους παραπάνω ορισμούς, είναι φανερή η εξαιρετική ομοιότητα των δυο αντικειμένων – να σημειώσουμε ότι δεν υπάρχουν ακόμα κοινά συμφωνημένοι ορισμοί για τις δυο έννοιες. Παρ' όλα αυτά, η βιοπληροφορική έχει καθιερωθεί σαν μια «εφαρμοσμένη επιστήμη», ενώ η υπολογιστική βιολογία έχει περισσότερο θεωρητικές απολήξεις (βελτίωση αλγορίθμων σε βιολογικά δεδομένα κ.ο.κ.).

3.3 Κυριότεροι Τομείς Έρευνας στη Βιοπληροφορική

Οι κυριότεροι τομείς έρευνας στη βιοπληροφορική και την υπολογιστική βιολογία είναι[4]:

i. Η ανάπτυξη εργαλείων που να επιτρέπουν την ανάλυση, σύγκριση και κατηγοριοποίηση ακολουθιών βιολογικών δεδομένων.
ii. Η ανάπτυξη εργαλείων που να επιτρέπουν την ερμηνεία αποτελεσμάτων βιολογικής σημασίας.
iii. Η αποδοτική οργάνωση των δεδομένων, ώστε να είναι δυνατή η αποθήκευση, ανάκτηση και ενημέρωσή τους.

3.3.1 Ανάλυση, Σύγκριση, Κατηγοριοποίηση και Ταξινόμηση ακολουθιών βιολογικών δεδομένων

Η βασική υπόθεση για την ανάπτυξη και εφαρμογή τεχνικών διαχείρισης συμβολοσειρών βιολογικών δεδομένων είναι ότι **κάθε βιολογικό μόριο μπορεί να περιγραφεί ως μια ακολουθία συμβόλων από ένα ορισμένο αλφάβητο Σ**. Συγκεκριμένα, κάθε μόριο του DNA μπορεί να θεωρηθεί ως μια ακολουθία συμβόλων (συμβολοσειρά), από ένα αλφάβητο τεσσάρων χαρακτήρων / γραμμάτων: A,C,G,T, ενώ κάθε μόριο πρωτεΐνης μπορεί να θεωρηθεί ως μια ακολουθία συμβόλων (συμβολοσειρά) από ένα αλφάβητο είκοσι χαρακτήρων / γραμμάτων, των 20 αμινοξέων.

Κατά την ανάλυση ακολουθιών βιολογικών δεδομένων μας ενδιαφέρει είτε η ακριβής εύρεση προτύπου, είτε η προσεγγιστική εύρεση προτύπου. Με αυτόν τον τρόπο στις ακολουθίες DNA μπορούμε να προσδιορίσουμε τις περιοχές όπου βρίσκονται γονίδια, περιοχές όπου τερματίζει ή ξεκινάει η αντιγραφή του DNA κ.ο.κ. Σε πρωτεϊνικές ακολουθίες μπορούμε να καθορίσουμε εξελικτικές σχέσεις και να προβλέψουμε την δευτεροταγή ή τριτοταγή δομή τους.

Ένα από τα σημαντικότερα πεδία έρευνας αποτελεί η πολλαπλή στοίχιση ακολουθιών (multiple sequence alignment). Η μέθοδος είναι αναπόσπαστα δεμένη με την εξέλιξη (κληρονομούμενες αλλαγές πληροφορίας) ανεξάρτητα από το εάν η εξέλιξη αποτελεί τμήμα του προβλήματος. Η στοίχιση ακολουθιών μπορεί να κατηγοριοποιηθεί σε:

a) τοπική στοίχιση (local alignment)
b) ολική στοίχιση (global alignment)

Στην τοπική ευθυγράμμιση αναζητούμε περιοχές τοπικής ομοιότητας. Ο πρώτος αλγόριθμος τοπικής ευθυγράμμισης δημιουργήθηκε από τους Smith-Waterman (SW) και αρκετές σύγχρονες τεχνικές βασίζονται σε αυτόν. Αντίστοιχα ο

πρώτος αλγόριθμος ολικής ευθυγράμμισης δημιουργήθηκε από τους Needleman & Wunsch (NW). Και στις δυο περιπτώσεις υπάρχουν παραπάνω από μια δυνατές ευθυγραμμίσεις. Η βέλτιστη λύση πρέπει να ελαχιστοποιεί τις διαφορές ανάμεσα στις δυο ακολουθίες ή διαφορετικά να μεγιστοποιεί τη συνάρτηση ομοιότητας.

Το πρόβλημα εύρεσης τοπικών ευθυγραμμίσεων χρησιμοποιείται ευρέως στη σύγκριση μιας δοσμένης ακολουθίας μικρού μήκους (input query sequence) ως προς το σύνολο γνωστών ακολουθιών που αποθηκεύονται σε μια βάση δεδομένων. Παρά την μεγάλη αξία των αλγορίθμων SW και NW, εξαιτίας του ότι ανήκουν στην κατηγορία των λεγόμενων «σχολαστικών» (rigorous) αλγορίθμων, δεν ικανοποιούν από πλευράς ταχύτητας, ειδικά σε περίπτωση έρευνας βάσεων δεδομένων με εκατοντάδες χιλιάδες ή και εκατομμύρια ακολουθιών. Για αυτόν το λόγο αναπτύχθηκαν οι λεγόμενοι ευρεστικοί (heuristic) αλγόριθμοι.

Οι ευρεστικοί αλγόριθμοι χρησιμοποιούν προσεγγίσεις, οι οποίες επιτρέπουν η αναζήτηση ομόλογων ακολουθιών να γίνεται πολύ πιο γρήγορα (ομόλογες ονομάζονται οι ακολουθίες που έχουν αποκλίνει από μια κοινή προγονική ακολουθία). Τα προγράμματα που χρησιμοποιούνται πιο συχνά και βασίζονται στους ευρεστικούς αλγόριθμους, είναι τα BLAST και FASTA που αναζητούν περιοχές τοπικής ομοιότητας. Ουσιαστικά και τα δυο αυτά προγράμματα αποτελούν μια συλλογή εργαλείων ευθυγράμμισης ακολουθιών.

Το πρόγραμμα BLAST (Basic Local Alignment Search Tool) δημιουργήθηκε από τον Altschul το 1990 και βασίζεται στην κεντρική ιδέα της εύρεσης κοινών υποακολουθιών ίδιου μήκους (segment pairs) που εμφανίζονται και στη δοσμένη ακολουθία μικρού μήκους (input query sequence) και στο σύνολο των ακολουθιών μιας βάσης δεδομένων με βάση μια συγκεκριμένη συνάρτηση ομοιότητας (scoring threshold).

Ο αλγόριθμος FASTA δημιουργήθηκε από τους Lipman & Pearson το 1985 και βασίζεται στην κεντρική ιδέα της αναζήτησης μικρών λέξεων (words ή k-tuples) που εμφανίζονται και στις δυο ακολουθίες. Στην περίπτωση πρωτεϊνικών ακολουθιών το μήκος των λέξεων είναι 1-2 κατάλοιπα, ενώ για ακολουθίες DNA το μήκος μιας λέξης μπορεί να φθάνει τις 6 βάσεις. Ο αλγόριθμος χρησιμοποιεί ευρεστικές μεθόδους για να δημιουργήσει περιοχές που περιέχουν κοινές λέξεις. Η στοίχιση που προκύπτει περιλαμβάνει διαφορές ανάμεσα σε κοινές λέξεις.

Εκτός από τα δυο προγράμματα που αναφέραμε παραπάνω και τα οποία χρησιμοποιούνται ευρύτατα, μια σειρά από άλλες μεθόδους και αλγόριθμους βρίσκουν εφαρμογή στην ανάλυση και σύγκριση βιολογικών ακολουθιών. Πολλοί σχετικά πρόσφατοι αλγόριθμοι βρίσκουν εφαρμογή σε προβλήματα βιοπληροφορικής (ενδεικτικά αναφέρουμε τους αλγόριθμους Boyer-Moore, Knuth-Morris-Prat για προβλήματα ακριβούς εύρεσης προτύπου). Μεγάλη εφαρμογή το τελευταίο διάστημα έχουν και τα λεγόμενα δέντρα επιθεμάτων[5] (suffix tree και generalized suffix tree).

Πέρα από την ανάλυση και σύγκριση ακολουθιών, η κατηγοριοποίηση αποτελεί ένα ευρύ πεδίο έρευνας στην βιοπληροφορική και την υπολογιστική βιολογία. Οι τεχνικές ανάλυσης συστάδων (clustering) αποτελούν μια στατιστική διαδικασία πολλών μεταβλητών, η οποία, ξεκινώντας από ένα σύνολο δεδομένων, επιχειρεί να το οργανώσει σε ομάδες ομοειδών στοιχείων που ονομάζουμε συστάδες (clusters). Οι ομάδες αυτές δεν είναι εκ των προτέρων γνωστές, αλλά προκύπτουν δυναμικά. Μια σειρά από μεθόδους κατηγοριοποίησης χρησιμοποιούνται, οι οποίες θα μπορούσαν να καταταχθούν στις/στα :

1. **ιεραρχικές μεθόδους (hierarchical methods)**
2. **διαιρετικές μεθόδους (partitioning methods)**

3. **γραφοθεωρητικές μεθόδους (graph-based methods)**
4. **τεχνητά νευρωνικά δίκτυα (artificial neural networks)**
5. **εξελικτικές μεθόδους (genetic algorithms)**
6. **μηχανές υποστήριξης διανύσματος (support vector machines)**

Χωρίς να εισερχόμαστε σε μια ανάλυση της κάθε μεθόδου, θα πρέπει να σημειώσουμε ότι κάθε μέθοδος έχει σε κάποιες εφαρμογές καλή απόδοση, ενώ σε κάποιες άλλες δεν εμφανίζει σημαντικά ποσοστά επιτυχίας. Οι μεγαλύτερες διαφορές των μεθόδων κατηγοριοποίησης εντοπίζονται: α) στη δομή των συστάδων, β) στην ύπαρξη επικαλύψεων και στο αντίστοιχο ποσοστό και γ) στη μετρική ομοιότητας που χρησιμοποιείται. Η δομή των συστάδων αναφέρεται τόσο στο πλήθος των συστάδων, και στο σχήμα τους, όσο και στη δυναμικότητά τους, δηλαδή το πλήθος των στοιχείων που περιλαμβάνουν σε απόλυτο ή σχετικό μέγεθος. Επίσης, η ύπαρξη επικάλυψης ανάμεσα σε συστάδες και το ποσοστό επικάλυψης που επιτρέπουμε επηρεάζουν τις παραμέτρους των μεθόδων κατηγοριοποίησης. Ανάλογα με το είδος των δεδομένων που επεξεργαζόμαστε επιλέγουμε τόσο τη μέθοδο ομαδοποίησης, όσο και τις αντίστοιχες παραμέτρους. Επομένως, η γνώση των χαρακτηριστικών και του είδους των δεδομένων λειτουργεί ως είσοδος στο πρόβλημα κατηγοριοποίησης (knowledge-based clustering).

Το ζήτημα της ταξινόμησης μπορεί να καταταχθεί στο ευρύτερο αντικείμενο της αναγνώρισης προτύπων (pattern recognition). Δυο από τα κυριότερα προβλήματα αναγνώρισης προτύπων στη βιοπληροφορική είναι η αναγνώριση γονιδίων και ο καθορισμός από τα συστατικά μιας ακολουθίας αμινοξέων της δευτεροταγούς δομής της πρωτεΐνης που θα προκύψει. Υπάρχουν αρκετοί τρόποι διεκπεραίωσης του προβλήματος αναγνώρισης προτύπων σε βιολογικά μακρομόρια. Πολλά από αυτά βασίζονται στην μηχανική μάθηση (machine learning) και στα πιθανοθεωρητικά μοντέλα όπως επίσης και στα νευρωνικά δίκτυα.

Όσον αφορά τα πιθανοθεωρητικά μοντέλα η μορφή που χρησιμοποιείται ως επί το πλείστον είναι οι αλυσίδες Markov[6]. Οι Αλυσίδες Markov (Markov Chains), είναι στοχαστικά μοντέλα, με τα οποία περιγράφουμε και αναλύουμε τις ακολουθίες βιολογικών πολυμερών όπως το DNA και οι πρωτεΐνες. Πρέπει εδώ να τονιστεί ότι το μοντέλο Markov θεωρείται από πολλούς ερευνητές ως το πιο φυσικό για να περιγράψει αλληλουχίες μακρομορίων όπως του DNA, αλλά και των πρωτεϊνών, και αυτό φαίνεται διαισθητικά φυσικό καθώς αυτή η εξάρτηση φαίνεται να προσεγγίζει την έννοια της πληροφορίας που εμπεριέχεται σε μια αλληλουχία. Ήδη από την δεκαετία του ’70 τα μοντέλα αυτά χρησιμοποιούνταν και χρησιμοποιούνται με σκοπό την αναγνώριση και επεξεργασία εικόνας, ήχου κ.α. και υπάρχει πλούσια βιβλιογραφία πάνω στα θέματα αυτά. Η πιο απλή εξήγηση για τα παραπάνω είναι το γεγονός ότι σε οποιοδήποτε κωδικοποιημένο σύστημα επικοινωνίας, όπως στις φυσικές γλώσσες, υπάρχει μια εσωτερική δομή που καθορίζει κάποιο είδος εξάρτησης των συμβόλων. Για παράδειγμα, στην αγγλική γλώσσα το γράμμα Q ακολουθείται σχεδόν πάντοτε από το U, άρα η πιθανότητα να εμφανιστεί το U σε μια θέση δεν είναι πάντα ίδια, αλλά εξαρτάται από το αν προηγήθηκε το Q. Για την ακρίβεια ο ίδιος ο Ρώσος Μαθηματικός Andrey Markov (1856-1922) οδηγήθηκε στην σύλληψη της έννοιας των ομώνυμων αλυσίδων μελετώντας τις εναλλαγές φωνηέντων και συμφώνων σε κάποιο ποίημα του Pushkin.

Έστω ότι έχουμε μια αλυσίδα π.χ. DNA

ATTGTAATCTCACGGTGTACGCGCATGCACAGTCAGT

ή μια αμινοξική αλληλουχία

AEDGPRGSDADKLIAVCLIGFVLIFVSLVCVTYTRED

Αν θεωρήσουμε ότι τα σύμβολα (νουκλεοτίδια ή αμινοξέα) δεν είναι ανεξάρτητα μεταξύ τους, αλλά το ποιο θα ακολουθήσει εξαρτάται μόνο από το αμέσως προηγούμενό του, τότε η πιθανότητα να εμφανιστεί π.χ. κάποιο b, δεδομένου ότι το αμέσως προηγούμενο του είναι α, (πιθανότητα μεταβάσεως - transition probability) θα είναι:

$$p_{ab} = P(x_i = b | x_{i-1} = a)$$

και η συνολική πιθανότητα να παρατηρηθεί η δεδομένη αλληλουχία θα είναι:

$$p(x) = P(x_n | x_{n-1}) P(x_{n-1} | x_{n-2}) \ldots P(x_1) =$$
$$= P(x_1) \prod_{i=2}^{n} P(x_i | x_{i-1})$$

Το μοντέλο αυτό με επεκτάσεις του σε εξάρτηση, πέραν της πρώτης τάξης χρησιμοποιείται ευρέως στην ανίχνευση γονιδίων σε νεοανακαλυφθείσες ακολουθίες DNA.[7]

Μια επέκταση των αλυσίδων Markov, είναι και το Hidden Markov Model (HMM). Στο μοντέλο αυτό, έχουμε μια ακολουθία συμβόλων και μια αλληλουχία καταστάσεων π.χ.

```
..AEDGPRGSDADKLIAVILIGFVLIFVALVCVTYTRED..
..------------+++++++++++++++++-------..
```

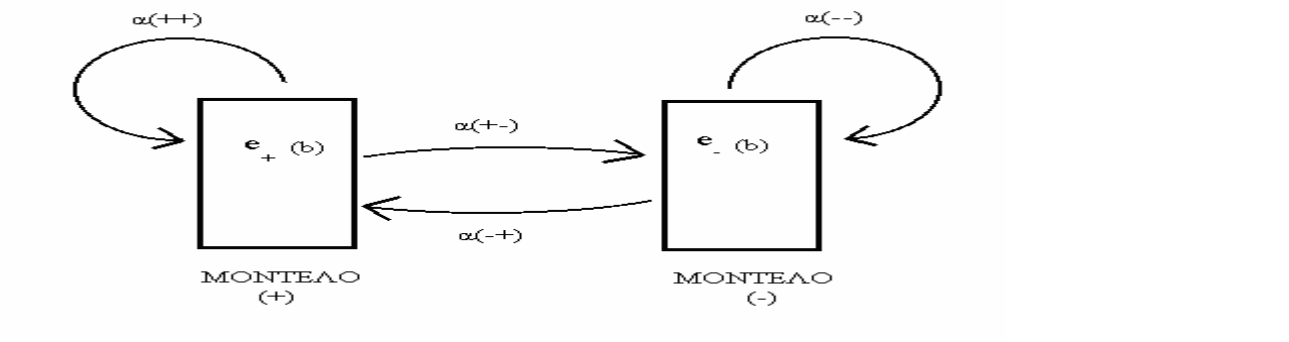
Στη συγκεκριμένη περίπτωση με (+) συμβολίζουμε τα διαμεμβρανικά τμήματα της ακολουθίας, ενώ με (-) τα μη διαμεμβρανικά. Γενικότερα τα (+) θα μπορούσαν να συμβολίζουν την υποακολουθία που μας ενδιαφέρει, και τα (-) αυτή που δεν μας αφορά. Η βασική διαφορά από το προηγούμενο μοντέλο είναι ότι τώρα αλυσίδα Markov συνιστούν όχι τα ίδια τα σύμβολα (αμινοξέα) αλλά η αλληλουχία των καταστάσεων (+,-). Έτσι έχουμε τώρα τις πιθανότητες μεταβάσεως

$$a_{kl} = P(\pi_i = l | \pi_{i-1} = k)$$

και σε κάθε κατάσταση (state) ισχύουν διαφορετικές πιθανότητες εμφάνισης των συμβόλων (αμινοξέα). Οι πιθανότητες αυτές ονομάζονται πιθανότητες γενέσεως (emission probabilities) και, όπως είναι φυσικό, είναι δεσμευμένες στην κατάσταση που βρισκόμαστε (άλλα αμινοξέα έχουν προτίμηση για τα διαμεμβρανικά τμήματα και άλλα για τα μη διαμεμβρανικά):

$$e_k(b) = P(x_i = b | \pi_i = k)$$

Στο παρακάτω σχήμα φαίνεται το HMM που περιγράψαμε παραπάνω:



Σχήμα 3.2: Η μορφή ενός βασικού Hidden Markov Model

Ένα HMM, αφού υπολογιστούν οι παράμετροι του (πιθανότητες μεταβάσεως κλπ) από ένα γνωστό σύνολο δεδομένων (training set), χρησιμοποιείται για την πρόγνωση-αποκωδικοποίηση, σε ένα σύνολο δεδομένων με απροσδιόριστα χαρακτηριστικά (test set). Οι μέθοδοι αποκωδικοποίησης, δηλαδή εύρεσης της αλληλουχίας των καταστάσεων εάν είναι γνωστή αλληλουχία των συμβόλων, είναι βασικά 2, η αποκωδικοποίηση Viterbi, και η εκ των υστέρων αποκωδικοποίηση (posterior decoding). Συνήθως σε περιπτώσεις πολύπλοκων μοντέλων, είναι πιο χρήσιμη η εκ των υστέρων αποκωδικοποίηση.

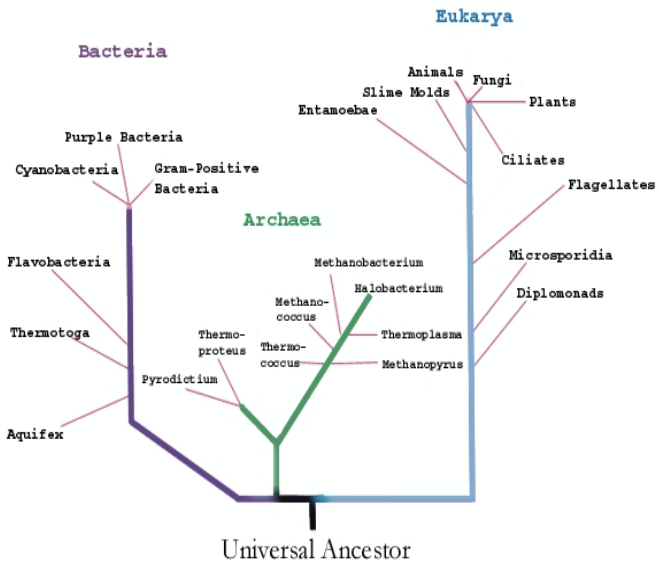
Εκτός από την ευρεία εφαρμογή των HMM και των νευρωνικών δικτύων σε προβλήματα βιοπληροφορικής, μια σειρά από άλλες μεθόδους συναντιούνται στην αρθρογραφία όπως αυτή της υπολογιστικής γλωσσολογίας, η οποία βασίζεται στην γλωσσολογική θεωρία του Chomsky, θεωρία που βασίζεται στους γενικευμένους γραμματικούς κανόνες που υπάρχουν για να δημιουργηθεί μια πρόταση ή αλλιώς μια ακολουθία χαρακτήρων[8]. Μια σειρά από εφαρμογές που σχετίζονται με τα νευρωνικά δίκτυα και το βακτήριο Escherichia Coli παρουσιάζονται στα κεφάλαια 4 και 5 αντίστοιχα.

3.3.2 Ανάπτυξη Μεθοδολογιών που επιτρέπουν την ερμηνεία αποτελεσμάτων βιολογικής σημασίας

Η αύξηση των βιολογικών δεδομένων με εκθετικό ρυθμό τα τελευταία χρόνια δεν θα μπορούσε να προσφέρει στην επιστημονική κοινότητα αξιόλογες πληροφορίες χωρίς την ανάπτυξη μεθοδολογιών που θα εξασφάλιζαν την όσο το δυνατόν καλύτερη ερμηνεία των νέων δεδομένων. Για αυτόν το λόγο τομείς όπως η φυλογενετική πρόβλεψη, ο καθορισμός τρισδιάστατων μορφών πρωτεϊνών από τις ακολουθίες μακρομορίων, η εξαγωγή συμπερασμάτων για την ρύθμιση της λειτουργίας ενός κυττάρου ή μιας πρωτεΐνης, επικεντρώνουν σε πολλές περιπτώσεις το ενδιαφέρον των ερευνητών. Παρακάτω παρουσιάζουμε συνοπτικά τις σημαντικότερες από αυτές τις μεθοδολογίες.

Η φυλογενετική πρόβλεψη υλοποιείται στις περισσότερες περιπτώσεις μέσα από τη δημιουργία εξελικτικών δέντρων[9] (Σχήμα 3.3). Τα δέντρα αυτά σχεδιάζονται μετά από σύγκριση βιολογικών ακολουθιών που ανήκουν σε διαφορετικούς οργανισμούς. Η ομαδοποίηση των βιολογικών ακολουθιών στο δέντρο γίνεται ανάλογα με τον βαθμό ομοιότητάς τους. Με αυτόν τον τρόπο έχουμε μια σαφή μορφή

απεικόνισης πάνω στο πώς οι βιολογικές ακολουθίες, οι οποίες αντιπροσωπεύουν και διαφορετικούς βιολογικούς οργανισμούς, μετασχηματίστηκαν κατά τη διάρκεια της εξελικτικής διαδικασίας. Οι μέθοδοι που χρησιμοποιούνται κυρίως για να δημιουργηθούν φυλογενετικά δέντρα, αφού οι ακολουθίες ευθυγραμμιστούν, είναι α) η μέθοδος μέγιστης συντήρησης (maximum parsimony), β) η μέθοδος ιεραρχικής ομαδοποίησης (hierarchical clustering) και γ) η μέθοδος μέγιστης πιθανοφάνειας (maximum likelihood). Ανάλογα με τον τρόπο δημιουργίας του φυλογενετικού δέντρου – είτε βάσει της απόστασης οπότε και επιλέγεται η ιεραρχική ομαδοποίηση, είτε βάσει εξελικτικού μοντέλου οπότε και επιλέγεται η μέθοδος μέγιστης συντήρησης ή η μέθοδος μέγιστης πιθανοφάνειας - επιλέγεται η μια από τις παραπάνω μεθόδους. Μία από τις πιο γνωστές μορφές ιεραρχικής ομαδοποίησης είναι και ο αλγόριθμος UPGMA (Unweighted Pair Group Method using arithmetic Averages), όπως και το εργαλείο PHYLIP (Phylogenetic Inference Package) που μπορεί να βρεθεί και στο διαδίκτυο[10].

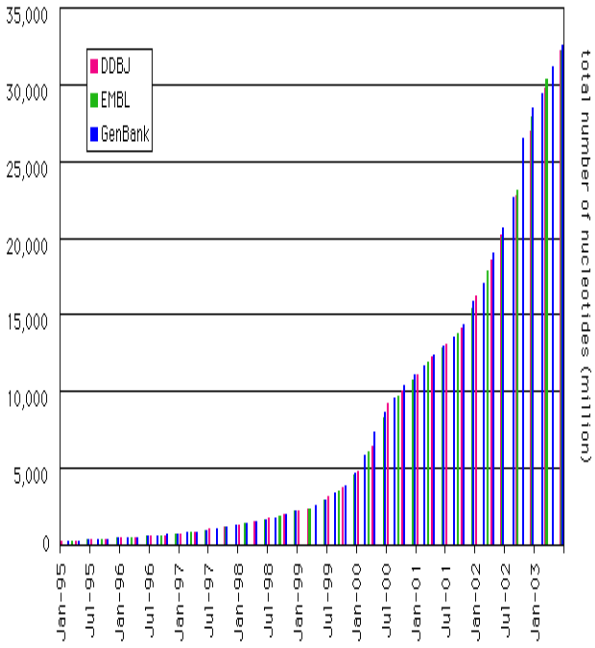


Σχήμα 3.3: Παράδειγμα φυλογενετικού δέντρου

Ο καθορισμός των τρισδιάστατων μορφών πρωτεϊνών από τις βιολογικές ακολουθίες έχει αποτελέσει ένα από τα σημαντικότερα σημεία έρευνας, για αυτόν άλλωστε το λόγο μια σειρά από τέτοιου είδους εφαρμογές μπορούν να βρεθούν στο διαδίκτυο. Υπολογιστικά, το πρόβλημα καθορισμού από μια ακολουθία RNA της τρισδιάστατης μορφής μιας πρωτεΐνης είναι δύσκολο, αφού απαιτούνται αλγόριθμοι πολυπλοκότητας τρίτου βαθμού. Παράλληλα, η εξαγωγή συμπεράσματος για το σχήμα που παίρνει η τρισδιάστατη μορφή μιας πρωτεΐνης από μια αμινοξική ακολουθία παραμένει ένα άλυτο πρόβλημα. Μεγάλη εφαρμογή και σε αυτήν την περίπτωση βρίσκουν τα HMM (Hidden Markov Models), ενώ εφαρμογή βρίσκει και η μέθοδος CFG (Context-Free Grammar), εφαρμογή που βασίζεται και πάλι στην υπολογιστική γλωσσολογία.

3.3.3 Αποδοτική Οργάνωση βιολογικών δεδομένων

Η αποδοτική οργάνωση βιολογικών δεδομένων είναι κάτι παραπάνω από απαραίτητη εξαιτίας του τεράστια και ολοένα αυξανόμενου πλήθους βιολογικών ακολουθιών που πρέπει να αποθηκεύονται με έναν τρόπο που να διευκολύνει τόσο την ανάκτησή τους, όσο και την επεξεργασία τους. Όταν πρωτοξεκίνησε η δημιουργία των βιολογικών βάσεων δεδομένων, ο όγκος της πληροφορίας ήταν τόσο μικρός που ένας μικρός αριθμός ερευνητών αρκούσε για την συντήρηση και για την ανανέωση των βάσεων αυτών. Αν κάποιος ερευνητής ενδιαφερόταν να έχει πρόσβαση στις εγγραφές της βάσης, επικοινωνούσε με τους επιστημονικούς υπευθύνους και εκείνοι του έστελναν με συμβατικό ταχυδρομείο όλη τη βάση η οποία αρκούσε να αποθηκευτεί ακόμη και σε μερικές δισκέτες ή μια μαγνητοταινία.



Σχήμα 3.4: Η αύξηση των νουκλεοτιδίων τα τελευταία 10 περίπου χρόνια στις κυριότερες νουκλεοτιδικές βάσεις.

Την τελευταία δεκαετία όμως η τεχνολογική εξέλιξη βοήθησε στη διεκπεραίωση μεγάλου όγκου πειραματικής εργασίας, η οποία σε συνάρτηση με τον διαρκή προσδιορισμό γονιδιωμάτων διαφόρων οργανισμών αύξησε τον όγκο της πληροφορίας στο επίπεδο της ακολουθίας, και όχι μόνο, σε δυσθεώρητα μεγέθη. Οι βάσεις πλέον δεν περιέχουν απλώς πολλά δεδομένα, αλλά και η διαδικασία ανανέωσης τους είναι απαραίτητη καθημερινή υπόθεση. Πλέον η συντήρηση μιας βάσης απαιτεί ένα πολυάριθμο επιτελείο επιστημόνων, οι οποίοι ασχολούνται αποκλειστικά με το σχολιασμό (annotation) των νεοεισερχόμενων δεδομένων καθώς και με τη διόρθωση λαθών των ήδη υπαρχόντων. Χαρακτηριστικά παραδείγματα αποτελούν η βάση πρωτεϊνικών ακολουθιών **SWISS-PROT** που περιέχει 167089 ακολουθίες (version 45.5 - Ιανουάριος 2005), ενώ η **EMBL Nucleotide Sequence Database** που περιέχει νουκλεοτιδικές αλληλουχίες έχει 46105397 εγγραφές (release 81 - Δεκέμβριος 2004). Η πρόσβαση στις βάσεις αυτές είναι πλέον εύκολη μέσω της χρήσης του διαδικτύου. Ο χρήστης μπορεί να επισκεφτεί την ιστοσελίδα που διατηρείται από τους υπευθύνους της βάσης και να κάνει αναζητήσεις αποθηκεύοντας στον υπολογιστή του δεδομένα του άμεσου ενδιαφέροντός του. Παράλληλα έχουν δημιουργηθεί και μια σειρά από βάσεις που αποσκοπούν στην ταξινόμηση της πληροφορίας στο επίπεδο της ακολουθίας και της δομής, προκειμένου να οργανωθεί η πληροφορία και να εξαχθούν συμπεράσματα για την βιολογική τους σημασία.

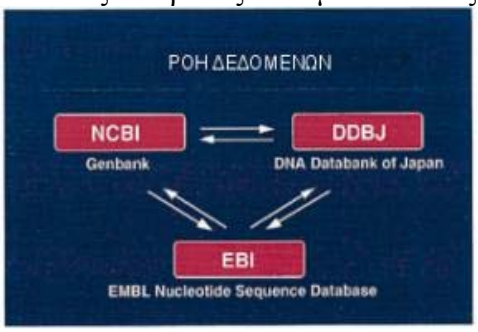
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΝΟΥΚΛΕΟΤΙΔΙΚΩΝ ΑΚΟΛΟΥΘΙΩΝ

Οι βάσεις δεδομένων νουκλεοτιδικών αλληλουχιών αποτελούν τις μεγαλύτερες βάσεις στο ευρύτερο πεδίο της Βιολογίας τόσο από άποψη του όγκου της πληροφορίας που περιέχουν, όσο και από την άποψη του εκθετικού ρυθμού συσσώρευσης δεδομένων που εμφανίζουν. Τα τελευταία χρόνια λόγω της εξέλιξης της τεχνολογίας στην εύρεση της αλληλουχίας (sequencing) πολυνουκλεοτιδίων έγινε εφικτός, σε μικρό χρονικό διάστημα, ο προσδιορισμός της αλληλουχίας ολόκληρων γονιδιωμάτων αρκετών οργανισμών, όπως ο άνθρωπος. Σε αρκετές περιπτώσεις μάλιστα υπάρχουν εξειδικευμένες βάσεις δεδομένων που περιέχουν τις αλληλουχίες για ένα και μόνο οργανισμό.

Εδώ πρέπει να σημειώσουμε τις τρεις μεγαλύτερες βάσεις δεδομένων νουκλεοτιδικών αλληλουχιών που είναι ελεύθερα διαθέσιμες στην ακαδημαϊκή κοινότητα. Πρόκειται για τις GENBANK (NCBI), DNA Data Bank of Japan (DDBJ) και EMBL Nucleotide Sequence Database (EBI), οι οποίες σε συνεργασία έχουν δημιουργήσει την International Nucleotide Sequence Database Collaboration. Η συνεργασία μεταξύ των βάσεων περιλαμβάνει την ανταλλαγή σε καθημερινή βάση

εγγραφών που κατατίθενται ανεξάρτητα σε κάθε βάση δεδομένων έχοντας θέσει παράλληλα και κοινούς κανόνες για την ταξινόμηση και το σχολιασμό των δεδομένων. Στο δίπλα σχήμα παρουσιάζεται η ροή της πληροφορίας ανάμεσα στις βάσεις.

GENBANK: (http://www.ncbi.nlm.nih.gov/Genbank/index.html) Η GENBANK αποτελεί μια βάση δεδομένων νουκλεοτιδικών αλληλουχιών που είναι ελεύθερα διαθέσιμη στην επιστημονική κοινότητα. Βρίσκεται υπό την αιγίδα του Εθνικού Ινστιτούτου Υγείας των ΗΠΑ. Η κύρια πηγή της πληροφορίας που περιέχεται στην GENBANK προέρχεται από απευθείας υποβολές δεδομένων όπως προκύπτουν από πειραματικές διεργασίες διαφόρων ερευνητικών ομάδων. Τα νεοεισερχόμενα δεδομένα υφίστανται επεξεργασία και προστίθενται σχόλια (annotation) για την διευκόλυνση των ερευνητών. Ανά τακτά χρονικά διαστήματα τα ήδη κατατεθειμένα δεδομένα επανεξετάζονται και γίνονται διορθώσεις, αν προκύπτουν νέα δεδομένα σχετικά με τις εγγραφές. Η διαδικασία κατάθεσης των δεδομένων μπορεί να πραγματοποιηθεί πολύ γρήγορα μέσω του Διαδικτύου με την συμπλήρωση κατάλληλης φόρμας και στη συνέχεια οι υπεύθυνοι της βάσης αναλαμβάνουν το σχολιασμό της εγγραφής και τη δημοσιοποίησή της στην βάση. Έχει διαπιστωθεί ότι κάθε 14 μήνες ο αριθμός των νουκλεοτιδικών βάσεων που περιέχονται στην GENBANK διπλασιάζεται.



Σχήμα 3.5: Η επικοινωνία μεταξύ των τριών μεγαλύτερων βάσεων βιολογικών δεδομένων είναι συνεχής και αμφίδρομη.

EMBL-Bank: (http://www.ebi.ac.uk/embl/) Η EMBL Nucleotide Sequence Database αποτελεί τη μεγαλύτερη βάση νουκλεοτιδικών αλληλουχιών στην Ευρώπη και βρίσκεται υπό την αιγίδα του Ευρωπαϊκού Εργαστηρίου Μοριακής Βιολογίας (EMBL). Εδράζεται και συντηρείται στο Ευρωπαϊκό Ινστιτούτο Βιοπληροφορικής (EBI) στο Cambridge, UK. Τα δεδομένα προέρχονται από ανεξάρτητα ερευνητικά εργαστήρια καθώς και από ομάδες που ασχολούνται με τον προσδιορισμό των γονιδιωμάτων διαφόρων οργανισμών. Η κατάθεση ακολουθιών στην EMBL-Bank είναι μια διαδικασία απλή και πραγματοποιείται μέσω του Διαδικτύου κατ' αντίστοιχο τρόπο με αυτό της GENBANK. Στη συνέχεια οι νεοεισερχόμενες ακολουθίες υφίστανται επεξεργασία και σχολιασμό από τους υπευθύνους της βάσης προτού γίνουν διαθέσιμες στην επιστημονική κοινότητα. Επιπλέον μέσω του Διαδικτύου παρέχονται μια σειρά από εργαλεία ανάλυσης ακολουθιών (π.χ. Fasta, BLAST).

DDJB: (http://www.ddbj.nig.ac.jp/) Η DNA Databank of Japan ιδρύθηκε το 1986 στο Εθνικό Ινστιτούτο Γενετικής (NIG) το οποίο βρίσκεται υπό την αιγίδα του Υπουργείου Παιδείας, Επιστημών και Αθλητισμού της Ιαπωνίας. Αποτελεί τη μοναδική διεθνώς αναγνωρισμένη βάση νουκλεοτιδικών αλληλουχιών στην Ιαπωνία, ενώ η κύρια πηγή δεδομένων της είναι οι εργασίες Ιαπώνων ερευνητών. Επιπλέον παρέχονται μια σειρά από εργαλεία για την ανάλυση των νουκλεοτιδικών αλληλουχιών.

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΠΡΩΤΕΪΝΙΚΩΝ ΑΚΟΛΟΥΘΙΩΝ ΚΑΙ ΒΑΣΕΙΣ ΓΙΑ ΤΗΝ ΑΝΑΛΥΣΗ ΑΚΟΛΟΥΘΙΩΝ

Η SWISS-PROT (http://www.expasy.ch/sprot/), είναι μια βάση δεδομένων πρωτεϊνικών ακολουθιών που ιδρύθηκε το 1986 και στις μέρες μας συντηρείται από

το Ελβετικό Ινστιτούτο Βιοπληροφορικής (Swiss Institute of Bioinformatics) σε συνεργασία με το Ευρωπαϊκό Ινστιτούτο Βιοπληροφορικής (European Bioinformatics Institute). Η παρούσα έκδοση της SWISS-PROT περιέχει 167089 καταχωρίσεις (Ιανουάριος 2005) στις οποίες εκτός από την ακολουθία υπάρχουν και συμπληρωματικά σχόλια όπως, βιβλιογραφικές αναφορές, γενικά στοιχεία δευτεροταγούς δομής, σύνδεσμοι σε άλλες βάσεις δεδομένων σχετικές με κάθε εγγραφή καθώς και σημειώσεις για τη βιολογική λειτουργία (αν είναι γνωστές) και άλλες χρήσιμες πληροφορίες.

Η **Protein Information Resource (PIR)** (http://pir.georgetown.edu/) εδράζεται στο Πανεπιστήμιο του Georgetown και αποτελεί τμήμα του Εθνικού Ιδρύματος Βιοϊατρικής Έρευνας (NBRF) των Η.Π.Α. Η PIR περιλαμβάνει μια σειρά από βάσεις δεδομένων που σχετίζονται με τη μελέτη των πρωτεϊνών με κυριότερη από αυτές την PIR-International Protein Sequence Database (PSD). Η PSD αποτελεί όπως και η SWISS-PROT μια βάση δεδομένων πρωτεϊνικών ακολουθιών συνοδευόμενη από συμπληρωματικά σχόλια. Τα δεδομένα της PSD προκύπτουν από την συνεργασία της PIR με το Munich Information Center for Protein Sequences (MIPS) και την Japanese International Protein Information Database (JIPID). Η τελευταία έκδοση της PSD (release 80, December 2004) περιλαμβάνει 283416 εγγραφές. Πρέπει να σημειωθεί ότι η PIR-PSD σε 4 υποενότητες τις PIR1, PIR2, PIR3 και PIR4. Μεταξύ των PIR1 και PIR2 δεν υπάρχει ουσιαστική διαφορά. Ο διαχωρισμός διατηρείται κυρίως για ιστορικούς λόγους. Οι ενότητες αυτές περιέχουν το 99% των εγγραφών της βάσης και τα κριτήρια ταξινόμησης και σχολιασμού είναι ακριβώς τα ίδια. Αντίθετα οι εγγραφές της PIR3 δεν έχουν ακόμη υποστεί έλεγχο και σχολιασμό. Τα περιεχόμενα της PIR4 είναι ακολουθίες, οι οποίες είτε δεν συναντώνται στη φύση είτε δεν εκφράζονται υπό φυσιολογικές συνθήκες. Επίσης μπορεί να περιέχει ακολουθίες που έχουν συντεθεί de novo σε εργαστήριο. Σε κάθε περίπτωση πάντως έχουν υποστεί έλεγχο και σχολιασμό από τους υπεύθυνους της βάσης.

Η **PROSITE** (http://www.expasy.ch/prosite/) είναι μια βάση ταξινόμησης σε οικογένειες πρωτεϊνικών ακολουθιών και αυτοτελών περιοχών ακολουθιών (sequence domains). Βασίζεται στη γενικότερη παρατήρηση ότι ενώ υπάρχει ένας τεράστιος αριθμός διαφορετικών πρωτεϊνών στη φύση, αυτές μπορούν να ομαδοποιηθούν με βάση την ομοιότητα στην ακολουθία τους σε ένα μικρό αριθμό οικογενειών. Οι πρωτεΐνες ή οι αυτοτελείς δομικές περιοχές που ανήκουν στην ίδια οικογένεια έχουν την ίδια λειτουργία και προέρχονται από κοινό πρόγονο. Είναι φανερό ότι πρωτεΐνες που ανήκουν στην ίδια οικογένεια, έχουν τμήματα της ακολουθίας τους που είναι περισσότερο συντηρημένα στην πορεία της εξέλιξης τους. Αυτές οι περιοχές σχετίζονται άμεσα με τη λειτουργία τους και με τη δομή των πρωτεϊνών στο χώρο. Αναλύοντας τις ακολουθίες πρωτεϊνών που ανήκουν στην ίδια οικογένεια είναι δυνατό να προκύψει ένα «αποτύπωμα» χαρακτηριστικό για κάθε ομάδα, ικανό ώστε να τη διαχωρίζει από τις άλλες πρωτεϊνικές αλληλουχίες που δεν ανήκουν στην οικογένεια αυτή. Μια ανάλογη περίπτωση αποτελεί η λήψη αποτυπωμάτων από την αστυνομία. Ένα αποτύπωμα είναι ικανό για να ταυτίσει ένα άτομο. Παρόμοια και στις πρωτεΐνες η χρήση ενός τέτοιου αποτυπώματος μπορεί να χρησιμεύσει για να ταξινομηθεί μια άγνωστη πρωτεϊνική αλληλουχία σε μια γνωστή οικογένεια πρωτεϊνών δίνοντάς μας ενδείξεις για την πιθανή λειτουργία τους. Αυτή τη στιγμή η PROSITE περιέχει «αποτυπώματα» για 1000 περίπου οικογένειες. Για κάθε οικογένεια υπάρχει λεπτομερής ανάλυση για τη δομή και τη λειτουργία των πρωτεϊνών αυτών.

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΔΟΜΙΚΗΣ ΒΙΟΛΟΓΙΑΣ

Protein Data Bank (PDB): (www.rcsb.org) Η Protein Data Bank αποτελεί τη μοναδική βάση παγκοσμίως όπου είναι κατατεθειμένες οι τρισδιάστατες δομές βιολογικών μακρομορίων. Ιδρύθηκε το 1971 στα Brookhaven National Laboratories (BNL) των ΗΠΑ και περιελάμβανε 7 δομές μακρομορίων όπως αυτές προέκυψαν από κρυσταλλογραφικές μελέτες. Ο ρυθμός αύξησης των εγγραφών στη δεκαετία του '70 ήταν πολύ μικρός. Από το 1980 και μετά λόγω της τεχνολογικής εξέλιξης σε κάθε στάδιο του προσδιορισμού δομών ο ρυθμός προσθήκης δεδομένων στην PDB αυξήθηκε δραματικά. Πλέον στην βάση περιλαμβάνονται και δομές όπως προκύπτουν με φασματοσκοπία Πυρηνικού Μαγνητικού Συντονισμού (NMR). Στην παρούσα φάση (Ιανουάριος 2005) η PDB περιλαμβάνει 29040 δομές βιομορίων. Οι εγγραφές στην PDB εκτός από τις συντεταγμένες των ατόμων που απαρτίζουν τη δομή περιλαμβάνουν και επιπρόσθετα βοηθητικά στοιχεία όπως βιβλιογραφικές αναφορές, λεπτομέρειες για τον προσδιορισμό της δομής καθώς και άλλα στοιχεία που προκύπτουν από τη συγκεκριμένη δομή. Κάθε δομή, προτού διατεθεί στο κοινό, υφίσταται έλεγχο για την ορθότητα της με τη χρήση ειδικού λογισμικού. Στη συνέχεια, εφόσον περάσει τις δοκιμές με επιτυχία, αποκτά ένα χαρακτηριστικό κωδικό και προστίθεται στη βάση.

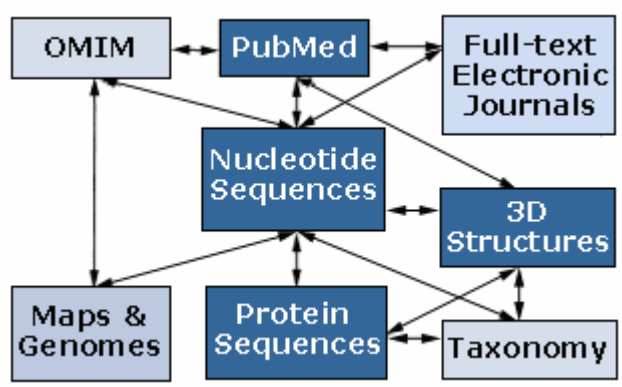
CATH: Η CATH (http://www.biochem.ucl.ac.uk/bsm/cath_new/index.html) είναι μια βάση ιεραρχικής ταξινόμησης των πρωτεϊνικών δομών που είναι κατατεθειμένες στην PDB με βάση τις αυτοτελείς δομικές περιοχές (domains) που τις απαρτίζουν. Για τον καταρτισμό της CATH δεν λαμβάνονται υπόψιν μη πρωτεϊνικές δομές, ενώ οι πρωτεϊνικές δομές που περιέχονται πρέπει να είναι προσδιορισμένες σε διακριτικότητα υψηλότερη των 3 Angstroms. Η CATH χρησιμοποιεί κυρίως αυτοματοποιημένες μεθόδους για την ταξινόμηση. Σε ειδικές περιπτώσεις όμως τα ανθρώπινα κριτήρια είναι δυνατόν να δώσουν καλύτερα αποτελέσματα από τις αυτοματοποιημένες μεθόδους όποτε και προτιμούνται.

SCOP: Η βάση SCOP (http://scop.mrc-lmb.cam.ac.uk/scop/index.html) έχει σαν βασικό στόχο την ανάλυση των δομικών και εξελικτικών σχέσεων μεταξύ όλων των πρωτεϊνών γνωστής δομής κατατεθειμένων στην Protein Data Bank (PDB). Για την αναγνώριση των παραπάνω σχέσεων και την ταξινόμηση με βάση τις σχέσεις αυτές, των πρωτεϊνών η διαδικασία δεν είναι αυτοματοποιημένη αλλά πραγματοποιείται αποκλειστικά με βάση τον ανθρώπινο παράγοντα μετά από λεπτομερή μελέτη και σύγκριση των πρωτεϊνικών δομών. Αυτοματοποιημένες μέθοδοι χρησιμοποιούνται μόνο για την ομοιογένεια των δεδομένων που περιέχονται στη βάση.

ΟΛΟΚΛΗΡΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΑΝΑΚΛΗΣΗΣ ΠΛΗΡΟΦΟΡΙΩΝ ΑΠΟ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Το **SRS** είναι ένα ισχυρό, εύχρηστο σύστημα διαχείρισης δεδομένων, το οποίο διατίθεται από την εταιρία LION Bioscience. Το SRS μέσω ενός φιλικού προς το χρήστη γραφικού περιβάλλοντος δίνει την δυνατότητα αναζήτησης και ανάκτησης δεδομένων από περισσότερες από 400 βάσεις δεδομένων, οι οποίες μπορεί να είναι αποθηκευμένες στον ίδιο κεντρικό υπολογιστή. Το μεγάλο του πλεονέκτημα είναι ότι μπορείς να κάνεις ταυτόχρονη αναζήτηση για ένα ζήτημα άμεσου ενδιαφέροντος σε παραπάνω από μια βάσεις δεδομένων που δεν περιέχουν ανάλογου είδους πληροφορία και η μορφοποίηση των δεδομένων σε καθεμιά να είναι διαφορετική. Ένα άλλο μεγάλης σημασίας πλεονέκτημα είναι η ταχύτητα με την οποία εκτελούνται

οι αναζητήσεις παρά το γεγονός ότι διαχειρίζεται πραγματικά τεράστιο όγκο πληροφορίας λόγω του μεγάλου αριθμού βάσεων που μπορεί να διαχειρίζεται ταυτόχρονα. Τέλος, δίνεται η δυνατότητα στον κάτοχο του συστήματος να ενσωματώνει σε αυτό και βάσεις που έχει δημιουργήσει ο ίδιος ή προγράμματα για κάθε είδος υπολογιστική ανάλυση χωρίς να επηρεάζεται η απόδοση του συστήματος.



Σχήμα 3.6: Οι δυνατότητες ενός ολοκληρωμένου εργαλείου όπως το Entrez σε μορφή διαγράμματος

Το **Entrez** αποτελεί ένα σύστημα διαχείρισης για την αναζήτηση και ανάκτηση πληροφοριών ανάλογο του SRS όλων των βάσεων δεδομένων που περιέχονται στο NCBI (National Center for Biotechnology Information) των ΗΠΑ. Το Entrez δίνει τη δυνατότητα αναζήτησης σε βάσεις δεδομένων νουκλεοτιδικών και πρωτεϊνικών ακολουθιών, δομές βιομορίων, γονιδιωμάτων και στη βάση για την αναζήτηση βιβλιογραφίας MEDLINE μέσω του ίδιου γραφικού περιβάλλοντος επιτρέποντας και πιο πολύπλοκες αναζητήσεις ανάμεσα στα στοιχεία τους. Βέβαια το γεγονός ότι περιορίζεται μόνο στις βάσεις δεδομένων του NCBI και ότι δεν επιτρέπει πολύπλοκες αναζητήσεις το καθιστούν υποδεέστερο έναντι του SRS.

Πέρα από τον διαχωρισμό που μπορεί να γίνει με βάση τους μηχανισμούς αποθήκευσης και διαχείρισης των δεδομένων σε απλές συλλογές αρχείων (flat-file), σχεσιακές (relational) και αντικειμενοστραφείς (object-oriented), ένας εναλλακτικός διαχωρισμός που βασίζεται στον τύπο των βιολογικών δεδομένων (ακολουθίες, μοτίβα, δομές) που περιέχουν, κατηγοριοποιεί τις βιολογικές βάσεις δεδομένων σε πρωτοταγείς και δευτεροταγείς. Οι πρωτοταγείς βάσεις βιολογικών δεδομένων περιέχουν την πειραματικά προσδιορισμένη πληροφορία, για παράδειγμα ακολουθίες νουκλεϊκών οξέων και πρωτεϊνών. Μερικές από τις γνωστές πρωτοταγείς βάσεις δεδομένων είναι οι EMBL, GenBank, DDBJ. Οι δευτεροταγείς βάσεις δεδομένων περιέχουν πληροφορίες οι οποίες προήλθαν από την ανάλυση των πρωτοταγών βάσεων. Με αυτόν τον τρόπο οι δευτεροταγείς βάσεις ανακαλύπτουν για παράδειγμα κοινά μοτίβα. Στις δευτεροταγείς βάσεις κατατάσσονται οι PROSITE, SWISS-PROT, BLOCKS. Παράλληλα έχουν αναπτυχθεί σύνθετες πρωτοταγείς και δευτεροταγείς βάσεις, οι οποίες αποτελούν έναν έμμεσο τρόπο επίλυσης του προβλήματος της ετερογένειας της αποθήκευσης των πληροφοριών στις απλές πρωτοταγείς και δευτεροταγείς βάσεις. Παραδείγματα σύνθετων πρωτοταγών βάσεων είναι οι OWL, NRDB και σύνθετων δευτεροταγών βάσεων οι ProWeb. Στις βάσεις δομικής βιολογίας το πρόβλημα είναι σαφώς μικρότερο αφού υπάρχει μόνο μια πρωτοταγής βάση (η PDB) και τα αποτελέσματα ταξινόμησης και ανάλυσης αυτών εμπεριέχονται σε μια πληθώρα δευτεροταγών βάσεων.

Ο τρόπος παρουσίασης μιας οποιασδήποτε βιολογικής πληροφορίας μέσα από αυτές τις βάσεις διαφέρει σε κάποιο βαθμό. Ο απλούστερος και αρκετά διαδεδομένος είναι αυτός της λεγόμενης μορφής FASTA (FASTA-format). Μια συνοπτική επεξήγηση τόσο για την μορφή FASTA, όσο και για την μορφή EMBL (EMPL-format) παραθέτουμε στο παράρτημα 1.

```
>gi|21326584|ref|NC_003977.1| Hepatitis B virus, complete genome
CTCCACAACATTCCACCAAGCTCTGCTAGATCCCAGAGTGAGGGGCCTATATTTTCCTGCTGGTGGCTCC
AGTTCCGGAACAGTAAACCCTGTTCCGACTACTGCCTCACCCATATCGTCAATCTTCTCGAGGACTGGGG
ACCCTGCACCGAACATGGAGAGCACAACATCAGGATTCCTAGGACCCCTGCTCGTGTTACAGGCGGGGTT
TTTCTTGTTGACAAGAATCCTCACAATACCACAGAGTCTAGACTCGTGGTGGACTTCTCTCAATTTTCTA
GGGGGAGCACCCACGTGTCCTGGCCAAAATTCGCAGTCCCCAACCTCCAATCACTCACCAACCTCTTGTC
CTCCAACTTGTCCTGGCTATCGCTGGATGTGTCTGCGGCGTTTTATCATATTCCTCTTCATCCTGCTGCT
```

Σχήμα 3.7: Η FASTA-μορφή, η πιο διαδεδομένη μορφή σε βιολογικές ακολουθίες.

ΕΡΓΑΛΕΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΙΟΛΟΓΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

Η ανάπτυξη του διαδικτύου άλλαξε τα δεδομένα και στη βιοπληροφορική και τις εφαρμογές της, καθώς δεν χρειάζεται πλέον η εύρεση και εγκατάσταση ενός προγράμματος σε κάθε υπολογιστή, αλλά μοναδική προϋπόθεση είναι η ύπαρξη σύνδεσης στο διαδίκτυο. Με άλλα λόγια η ανάπτυξη του μοντέλου χρήστη-εξυπηρετητή (client-server model), δίνει τη δυνατότητα να προσπελασθούν από το χρήστη βάσεις με τεράστιο αριθμό καταχωρίσεων βιολογικών δεδομένων, ενώ δίνεται και η δυνατότητα εκτέλεσης ενός προγράμματος στο υπολογιστικό σύστημα αυτού που παρέχει την υπηρεσία. Ο χρήστης ουσιαστικά βλέπει μόνο ένα user interface, και τα αποτελέσματα από την εκτέλεση του προγράμματος που επέλεξε. Το γεγονός αυτό έχει ιδιαίτερη σημασία για εφαρμογές στην βιολογία, καθώς τα δεδομένα έχουν μεγάλο μέγεθος και μπορούν να γίνουν με αυτόν τον τρόπο εκμεταλλεύσιμα με την απλή ύπαρξη λίγων αλλά ισχυρών εξυπηρετητών.

Άμεσο αποτέλεσμα των δυνατοτήτων που δόθηκαν από την ανάπτυξη του διαδικτύου ήταν η δημιουργία μιας σειράς εφαρμογών που διατίθενται μέσω αυτού. Η ανάπτυξη συλλογών εφαρμογών από τις ήδη γνωστές τράπεζες βιολογικών δεδομένων ήταν το επόμενο βήμα. Χαρακτηριστικό παράδειγμα είναι η συλλογή εργαλείων της τράπεζας EMBL, η οποία διαχωρίζει σε 4 κύριες κατηγορίες τις εφαρμογές αυτές: α) στις εφαρμογές που αφορούν την αναλογία και ομοιότητα ακολουθιών, β) σε αυτές που αναλύουν τον ρόλο των πρωτεϊνών γ) σε αυτές που ασχολούνται με τη ανάλυση της δομής και την πρόβλεψη της δευτεροταγούς δομής τους και δ) σε εφαρμογές που έχουν στόχο να αναλύσουν τις βιολογικές ακολουθίες με στόχο την εύρεση επαναλαμβανόμενων μοτίβων, DNA που κωδικοποιείται κοκ. Αντίστοιχα οργανωμένη είναι τόσο η τράπεζα NCBI όσο και η DDBJ με παρόμοιες εφαρμογές προς χρήση. Παράλληλα, μια σειρά από ερευνητικούς και εκπαιδευτικούς φορείς παρουσιάζουν μέσω διαδικτύου νέες εφαρμογές.

Εκτός από τα εργαλεία που προσφέρονται από τις συγκεκριμένες βάσεις δεδομένων μια σειρά άλλων υπάρχουν σε δικτυακούς τόπους ερευνητικών κέντρων και πανεπιστημιακών τμημάτων. Ειδικότερα όσον αφορά τα εργαλεία αναζήτησης γονιδίων, τα κυριότερα που προσφέρονται μέσα από δικτυακούς τόπους είναι τα εξής:

- ✓ GENSCAN http://genes.mit.edu/GENSCAN.html : Αποτελεί ένα από τα εργαλεία που χρησιμοποιείται κατά κόρον από τους ερευνητές. Βασίζεται σε πιθανοθεωρητικά μοντέλα για τη δομή γονιδίων.[11]
- ✓ NetGene http://www.cbs.dtu.dk/services/NetGene2/ : Χρησιμοποιεί τεχνητά νευρωνικά δίκτυα για την ανάλυση splice περιοχών (splice sites) κυρίως στο ανθρώπινο γονιδίωμα.[12]
- ✓ GeneID-3 http://www1.imim.es/geneid.html : Χρησιμοποιεί μοντέλα που βασίζονται σε κανόνες (rule-based models).[13]
- ✓ GeneMark http://opal.biology.gatech.edu/GeneMark/ : Βασίζεται στη μέθοδο των Hidden Markov Models.[14]

- ✓ GenLang http://www.cbil.upenn.edu/genlang/genlang.html : Χρησιμοποιεί γλωσσολογικές μεθόδους (linguistic methods).[15]

Όσον αφορά εργαλεία πρόβλεψης υποκινητών τα πιο αξιόπιστα και γνωστά είναι τα εξής:

- ✓ Neural Network Promoter Prediction (NNPP) http://www.fruitfly.org/seq_tools/promoter.html : Όπως φαίνεται και από την ονομασία του η συγκεκριμένη εφαρμογή βασίζεται στα νευρωνικά δίκτυα. Αναλυτική παρουσίαση υπάρχει στο κείμενο [19] της βιβλιογραφίας του παρόντος κεφαλαίου[16].
- ✓ PromoterScan http://bimas.dcrt.nih.gov/molbio/proscan/ : Η συγκεκριμένη εφαρμογή δεν δέχεται παράλληλη επεξεργασία δεδομένων από περισσότερους από έναν χρήστες.[17]
- ✓ PromoterInspector http://www.gsf.de/biodv/ : βασίζεται και αυτό στη τεχνική των νευρωνικών δικτύων.[18]

Να σημειώσουμε ότι τα εργαλεία αυτά απευθύνονται ως επί το πλείστον τόσο σε ευκαρυωτικούς, όσο και σε προκαρυωτικούς οργανισμούς[19].

3.4 Ανάπτυξη νέων εργαλείων στη Βιοπληροφορική

Η αλματώδης ανάπτυξη του κλάδου της Βιοπληροφορικής ανάγκασε μια σειρά από προγραμματιστές να εξετάσουν και να αναπτύξουν σε ήδη υπάρχουσες γλώσσες προγραμματισμού τη δυνατότητα διαχείρισης βιολογικών ακολουθιών από αυτές, όπως επίσης και τη δυνατότητα αποθήκευσής τους σε μορφή που να είναι πιο εύκολα διαχειρίσιμη και προσπελάσιμη. Τα αποτελέσματα αυτά οδήγησαν στην ανάπτυξη της Biojava, η οποία ουσιαστικά δίνει δυνατότητες στην προγραμματιστική γλώσσα Java να επεκταθεί στο πεδίο της Βιοπληροφορικής. Αντίστοιχες υλοποιήσεις, αλλά όχι με τόσο μεγάλο βαθμό ανάπτυξης, έγιναν τόσο στην γλώσσα Perl, όσο και στην γλώσσα Python.

3.4.1 Η Γλώσσα Προγραμματισμού Java

Η γλώσσα Biojava αποτελεί μια εκδοχή της γλώσσας Java, που δίνει τη δυνατότητα ευκολότερης διαχείρισης ακολουθιών βιολογικών δεδομένων. Μια σειρά από διαδικασίες που εφαρμόζονται συχνά στις βιολογικές ακολουθίες, κωδικοποιούνται σε κλάσεις και δίνουν τη δυνατότητα για πιο συστηματική και απλή διαχείριση των ακολουθιών.

Μια συνοπτική παρουσίαση της γλώσσας προγραμματισμού Java και της πλέον πρόσφατης έκδοσής της, της Java 2, θα μπορούσε να κωδικοποιηθεί στα παρακάτω σημεία:

1) Ιστορικά, η γλώσσα προγραμματισμού Java, αναπτύχθηκε με γοργούς ρυθμούς από το 1991, με αρχική επιδίωξη της ερευνητικής ομάδας που ασχολήθηκε στην αρχή με αυτή, την ανάπτυξη λογισμικού για έλεγχο ηλεκτρονικών συσκευών ευρείας κατανάλωσης. Ενώ αρχικά το όλο εγχείρημα θα υλοποιούνταν στη γλώσσα προγραμματισμού C++, η όχι καλή λειτουργικότητα της γλώσσας, οδήγησε στην ανάπτυξη μιας νέας γλώσσας προγραμματισμού, που αρχικά ονομάστηκε Oak

και στη συνέχεια Java. Εξαιτίας της μεταφερσιμότητας, της ευελιξίας και της αποδοτικότητας που θα έπρεπε να τη διακρίνει ούτως ώστε να είναι χρήσιμη σε μια σειρά διαφορετικών μεταξύ τους οικιακών συσκευών, αλλά και λόγω της μεγάλης αξιοπιστίας που θα έπρεπε να έχει, η Java είχε χαρακτηριστικά τα οποία ανταποκρίνονταν στις απαιτήσεις για νέους τρόπους ανάπτυξης και διανομής του λογισμικού με την ραγδαία εξάπλωση του Διαδικτύου (Internet) και του Παγκόσμιου Ιστού (World-Wide-Web). Η Java σχεδιάστηκε με σκοπό την ανάπτυξη εφαρμογών που τρέχουν σε ετερογενή δικτυακά περιβάλλοντα. Η γλώσσα προγραμματισμού Java και ειδικότερα η πλατφόρμα Java 2 υποστηρίζει κομβικά τα ακόλουθα χαρακτηριστικά:

2) Η γλώσσα Java είναι αντικειμενοστραφής (object-oriented language). Η διαχείριση όμως των αντικειμένων γίνεται με διαφορετικό σε σχέση με τη γλώσσα προγραμματισμού C++ τρόπο. Πιο συγκεκριμένα, η γλώσσα προγραμματισμού Java δεσμεύει και αποδεσμεύει μνήμη αυτόματα καθώς το πρόγραμμα δημιουργεί και καταστρέφει αντικείμενα. Όπου οι προγραμματιστές σε Java δημιουργούν και χρησιμοποιούν, προβλέψιμες και ασφαλείς, άμεσες αναφορές (references) σε αντικείμενα, οι προγραμματιστές σε C++ χρησιμοποιούν δείκτες (pointers), μια πρακτική, η οποία εγκυμονεί κινδύνους από διάφορες απόψεις.
3) Μία ακόμα θεμελιώδης διαφορά ανάμεσα στις δύο γλώσσες βρίσκεται στους μηχανισμούς με τους οποίους δίνουν σε ένα αντικείμενο πολλαπλούς τύπους. Η C++ υποστηρίζει πολλαπλή κληρονομικότητα (multiple inheritance), το οποίο σημαίνει ότι οι κλάσεις μπορούν να έχουν πολλαπλές βασικές κλάσεις. Αυτός ο μηχανισμός οδηγεί σε ασάφεια και σύγχυση. Η γλώσσα Java αντικαθιστά αυτόν το μηχανισμό με ένα πιο ξεκάθαρο είδος αφαιρετικής δομής, τις διεπαφές (interfaces). Μία κλάση Java μπορεί να έχει μία μόνο βασική κλάση (parent class) αλλά μπορεί να υλοποιεί πολλαπλές διεπαφές.
4) Η πλατφόρμα Java χρησιμοποιεί αρχιτεκτονική εικονικής μηχανής (virtual machine), ή (όπως αλλιώς είναι γνωστή) *διερμηνέα* Java (**Java Interpreter ή Java runtime**). Με αυτόν τον τρόπο, η εικονική μηχανή μπορεί να υλοποιηθεί, ώστε να λειτουργεί σε διάφορα λειτουργικά συστήματα και πλατφόρμες με σταθερότητα, ενώ παράλληλα παρέχει αυστηρό έλεγχο σε εκτελούμενα προγράμματα, δίνοντας έτσι την δυνατότητα για ασφαλή εκτέλεση αναξιόπιστου κώδικα.
5) Η πλατφόρμα Java περιλαμβάνει μια εκτεταμένη συλλογή από συγκεκριμένες βιβλιοθήκες κλάσεων που ονομάζονται Διεπαφές Προγραμματισμού Εφαρμογών (**Application Programming Interfaces** σύντομα **APIs**). Αυτά υποστηρίζουν οτιδήποτε μπορεί να χρειάζεται μια εφαρμογή.
6) Υποστηρίζεται δημιουργία ανεξάρτητων εφαρμογών και applets. Applets ονομάζονται προγράμματα που περιλαμβάνονται σε σελίδες γλώσσας σήμανσης υπερκειμένου

(HyperText Markup Language / HTML) και εκτελούνται από τον εξεταστή (Web browser).

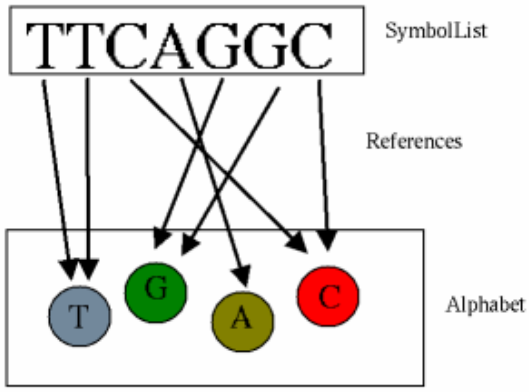
7) Η Java είναι κατανεμημένη (distributed), δηλαδή ένα πρόγραμμα Java είναι δυνατό να εγκατασταθεί και να λειτουργήσει διαμέσου του Διαδικτύου ή ακόμα και από διαφορετικές συσκευές στο Διαδίκτυο.
8) Η πλατφόρμα Java είναι ασφαλής (secure), καθώς στο Διαδίκτυο ελλοχεύουν πολλοί κίνδυνοι για τον χρήστη – παραλήπτη μιας δικτυακής εφαρμογής, ενώ η Java έχει σχεδιαστεί έτσι ώστε να ελαχιστοποιείται η πιθανότητα προσβολής του συστήματος του χρήστη από κάποιο πρόγραμμα γραμμένο για αυτό το σκοπό.
9) Υποστηρίζει ταυτοχρονισμό (multithreading). Η Java υποστηρίζει εγγενώς την χρήση νημάτων (Threads). Προκειμένου να το πετύχει αυτό σε συστήματα με έναν επεξεργαστή, το **Java runtime system (interpreter)** υλοποιεί ένα δικό του χρονοδρομολογητή (scheduler), ενώ σε συστήματα που υποστηρίζουν πολυεπεξεργασία η δημιουργία νημάτων ανατίθεται στο λειτουργικό σύστημα. Η όλη διαδικασία είναι αόρατη τόσο στον προγραμματιστή όσο και στον χρήστη.
10) Τέλος η Java υποστηρίζει εφαρμογές πολυμέσων τόσο με την ευελιξία της ως γλώσσα προγραμματισμού, όσο και με τις πλούσιες και συνεχώς εμπλουτιζόμενες βιβλιοθήκες.

3.4.2 Επισκόπηση της Γλώσσας Προγραμματισμού Biojava

Η αλματώδης ανάπτυξη, τόσο σε όγκο όσο και σε μορφές, της βιολογικής πληροφορίας έκανε απαραίτητη την αποθήκευση και διαχείρισή της με έναν βολικό τρόπο. Μια λογική προσέγγιση ήταν η αναπαράσταση κάθε μονομερούς σε ένα μακρομόριο με τη χρήση συγκεκριμένου αλφαβήτου. Η εισαγωγή των δεδομένων αυτών και στους υπολογιστές, έγινε με τον ίδιο τρόπο. Ένα μεγάλο μέρος λογισμικού υπολογιστικής βιολογίας βασίζεται σε απλή διαχείριση αλφαριθμητικών (string handling). Παρά την μέχρι στιγμής καλή λειτουργία της ιδέας της αναπαράστασης μιας βιολογικής ακολουθίας σαν αλληλουχίες ASCII χαρακτήρων, ανακύπτουν μια σειρά προβλημάτων που ενδέχεται να δημιουργήσουν προβλήματα στον προγραμματιστή:

- Είναι δυνατό οποιοδήποτε string που εισάγεται σε μια ρουτίνα να θεωρηθεί βιολογική ακολουθία. Για αυτόν το λόγο κάθε αναγνώριση (validation) πρέπει να γίνεται σε ad hoc βάση.
- Η σημασία κάθε συμβόλου δεν είναι απαραίτητα ξεκάθαρη (ambiguity). Το σύμβολο T μπορεί να συμβολίζει τη θυμίνη σε ακολουθίες DNA, αλλά σε πρωτεϊνικές ακολουθίες το ίδιο σύμβολο αναπαριστά τη θρεονίνη.
- Παρά το ότι υπάρχουν συγκεκριμένες κωδικοποιήσεις για την αναπαράσταση σαν strings των βιολογικών ακολουθιών, η ίδια προσέγγιση δεν λειτουργεί πάντα το ίδιο αποτελεσματικά για διαφορετικά είδη δεδομένων που δημιουργούνται από λογισμικά ανάλυσης βιολογικών ακολουθιών. Οι κωδικοποιήσεις αυτές είναι επομένως περιοριστικές (limited alphabet).

Η Biojava προσεγγίζει με διαφορετικό τρόπο τα δεδομένα των βιολογικών ακολουθιών. Αντί να χρησιμοποιεί string ASCII χαρακτήρων, η ακολουθία μοντελοποιείται σαν μια λίστα Java αντικειμένων που υλοποιεί την διεπαφή με το όνομα Symbol (Symbol interface). Αυτή η κλάση είναι μέρος του πακέτου **org.biojava.bio.symbol**. Η βασικότερη διεπαφή για δεδομένα ακολουθιών στη Biojava είναι η SymbolList. Οι SymbolLists πρέπει να εξετάζονται σαν strings που δημιουργούνται από απλά αντικείμενα Symbol και όχι από χαρακτήρες. Πέρα από το πακέτο org.biojava.bio.symbol, άλλα δυο πακέτα είναι εξίσου σημαντικά για βασικές λειτουργίες διαχείρισης και ανάλυσης βιολογικών ακολουθιών μέσω της Biojava. Αυτά είναι το **org.biojava.bio.seq**, το οποίο είναι αυτό που είναι υπεύθυνο για τη διαχείριση ακολουθιών που προέρχονται από αρχεία ή βάσεις δεδομένων, και το **org.biojava.bio.seq.io**, με τη βοήθεια του οποίου μπορεί να εμφανιστούν ή να εισαχθούν κάποια βιολογικά δεδομένα και αποτελέσματα αυτών. Στο παράρτημα, παρουσιάζουμε ένα απλό παράδειγμα που αναδεικνύει τις δυνατότητες της Biojava. Εξάλλου, νέες δυνατότητες αναδεικνύονται όταν σε συνδυασμό με την γλώσσα Biojava χρησιμοποιούνται σχεσιακές βάσεις δεδομένων, με κύριο παράδειγμα αυτό της BioSQL, η οποία βασίζεται στην αντικειμενοστραφή σχεσιακή δομή της SQL και επεκτείνει τις δυνατότητές της σε βιολογικής μορφής δεδομένα.[20]



Σχήμα 3.8: Ο τρόπος «ανάγνωσης» μιας DNA ακολουθίας με την γλώσσα Biojava

3.4.3 Το πακέτο εργαλείων Βιοπληροφορικής της MATLAB

Η MATLAB είναι ένα πρόγραμμα υπολογιστών για ανθρώπους που χρησιμοποιούν αριθμητικούς υπολογισμούς, ειδικά στη γραμμική άλγεβρα (πίνακες). Ξεκίνησε ως ένα πρόγραμμα "Εργαστηρίου Πινάκων" ("MATrix LABoratory") που είχε σκοπό να παρέχει αλληλεπιδρώσα προσπέλαση στις βιβλιοθήκες Linpack και Eispack. Από τότε έχει αναπτυχθεί αρκετά, για να γίνει ένα ισχυρότατο εργαλείο στην οπτικοποίηση, στον προγραμματισμό, στην έρευνα, στην επιστήμη των μηχανικών και στις επικοινωνίες.

Στο δυναμικό της Matlab συμπεριλαμβάνονται μοντέρνοι αλγόριθμοι, δυνατότητες χειρισμού τεράστιων ποσοτήτων δεδομένων, και ισχυρά προγραμματιστικά εργαλεία. Η Matlab έρχεται ως πακέτο του βασικού προγράμματος, με πολλές "εργαλειοθήκες" (toolboxes), που πωλούνται ξεχωριστά. Θα καλύψουμε μόνο το πακέτο εργαλείων βιοπληροφορικής (bioinformatics toolbox). Η τρέχουσα έκδοση (Μάιος 2004) είναι η Matlab 7 έκδοση 14.

Το πακέτο εργαλείων βιοπληροφορικής της τελευταίας έκδοσης της MATLAB επεκτείνει τις δυνατότητες της MATLAB στην παροχή ενός ολοκληρωμένου προγραμματιστικού περιβάλλοντος για την γονιδιωματική και πρωτεϊνωματική ανάλυση. Με αυτόν τον τρόπο δίνεται η δυνατότητα, κυρίως σε συνδυασμό με το πακέτο εργαλείων στατιστικής (statistics toolbox), να λυθούν μια σειρά προβλημάτων και να δημιουργηθούν εφαρμογές στην ανακάλυψη φαρμάκων,

στη γενετική μηχανική και στην βιολογική ερευνά. Η βασική λογική που υπάρχει στην χρησιμοποίηση του Bioinformatics Toolbox είναι η χρήση βασικών συναρτήσεων βιοπληροφορικής που δίνονται στο συγκεκριμένο πακέτο για τη δημιουργία πολυπλοκότερων συναρτήσεων, εφαρμογών και αλγόριθμων. Οι κύριες δυνατότητες που δίνονται μέσω του νέου αυτού πακέτου εργαλείων είναι κομβικά οι εξής:

- Μορφές και βάσεις δεδομένων. Δίνεται η δυνατότητα σύνδεσης μέσω του διαδικτύου σε βάσεις βιολογικών δεδομένων, η ανάγνωση και η μετατροπή των δεδομένων αυτών σε άλλες μορφές (π.χ. από FASTA μορφή σε EMBL μορφή κοκ).
- Ανάλυση βιολογικών ακολουθιών. Καθορίζονται τα στατιστικά χαρακτηριστικά των δεδομένων. Διαχείριση και ευθυγράμμιση ακολουθιών. Ιδανικά μοτίβα σε βιολογικές ακολουθίες με χρήση Hidden Markov Model (HMM).
- Φυλογενετική ανάλυση. Δημιουργία και διαχείριση φυλογενετικών δέντρων δεδομένων.
- Δεδομένα φασματομετρία μάζας. Ανάλυση και εμπλουτισμός δεδομένων φασματομετρίας μάζας.
- Στατιστική μάθηση. Κατηγοριοποίηση και ταύτιση χαρακτηριστικών σε σύνολα δεδομένων με στατιστικά εργαλεία μάθησης.
- Προγραμματιστική διεπιφάνεια. Χρήση άλλου βιοπληροφορικού λογισμικού σε περιβάλλον MATLAB.

Με τις παραπάνω δυνατότητες που δίνονται μέσω του Bioinformatics Toolbox είναι δυνατή η:

- Ανάπτυξη αλγόριθμων
- Οπτικοποίηση αποτελεσμάτων και δεδομένων
- Ανάπτυξη και διανομή εφαρμογών

Παραθέτουμε στο παράρτημα 2 ένα παράδειγμα χρήσης του Bioinformatics Toolbox για τη στατιστική ανάλυση ακολουθιών DNA.[21]

---

[1] Περισσότερα για το Human Genome Project στο:
http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml

[2] J. Cohen - Bioinformatics-An introduction for Computer Scientists – ACM Computing Surveys, Vol.36,No.2, June 2004

[3] Σημειώσεις στη βιοπληροφορική http://eclass.duth.gr
Κ. Περδικούρη – Αλγόριθμοι Διαχείρισης & Ανάλυσης Ακολουθιών Βιολογικών Δεδομένων – ΔΠΜΣ, Πληροφορική Επιστημών Ζωής, Πανεπιστήμιο Πάτρας, 2003-2004
Introduction in Computational Biology – http://www.lisha.ufsc.br/~guto

[4] Π. Αντωνίου – Σημειώσεις στην Υπολογιστική βιολογία – Πανεπιστήμιο Κύπρου, http://www.cs.ucy.ac.cy/courses/EPL450/

[5] Gusfield D. – Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology – Cambridge University Press, 1997

[6] Rabiner, L. R. – A tutorial on hidden Markov models and selected applications in speech recognition – Proc. IEEE 77, 2, 257-286

[7] Borodovsky and McIninch, 1993

[8] Searls, D.B. – The language of genes – Nature 420 (November), 211 -217
Searls, D.B. – The linguistics of DNA – Amer. Sci. 80, 579-591, 1992

[9] Felsenstein, J. – Inferring Phylogenies – Sinauer Associates, 2003

[10] http://evolution.genetics.washington.edu/phylip.html

[11] Burge and Karlin (1997, 1998)

[12] Brunak et al. (1991); Hebsgaard et al. (1996)

[13] Guigo et al. (1992); Guigo 1998

[14] Lukashin and Borodovsky (1998); Borodovsky and McIninch (1993)

[15] Dong and Searls (1994)

[16] Martin G. Reese – Computational prediction og gene structure nad regulation in the genome of Drosophila melanogaster – 2000

[17] Prestridge (1995)

[18] Klingenhoff et al. (1999)

[19] Ο δικτυακός τόπος http://www.BioinformaticsOnline.org διαθέτει μια πλούσια συλλογή εργαλείων που χρησιμοποιούνται τόσο για εύρεση γονιδίων, όσο και για πρόβλεψη υποκινητών

[20] Για περισσότερες πληροφορίες: http://www.biogava.org

[21] Για περισσότερες πληροφορίες: http://www.mathworks.com

Κεφάλαιο 4

Τεχνικές Μηχανικής Μάθησης στη Βιοπληροφορική

4 Τεχνικές Μηχανικής Μάθησης στη Βιοπληροφορική

4.1 Εισαγωγή

Οι διάφορες μορφές τεχνητής νοημοσύνης έχουν συμβάλει καταλυτικά στην ανάπτυξη της βιοπληροφορικής. Μια σειρά είτε από θεωρητικές, είτε από πρακτικότερες προσεγγίσεις βιολογικών προβλημάτων που αντιμετωπίζονται με μεθόδους από την πληροφορική, μπορούν να εξετασθούν σε μια σειρά σχετικά πρόσφατων χρονικά βιβλίων[1]. Τα νευρωνικά δίκτυα, η ασαφής λογική, οι γενετικοί αλγόριθμοι, τα δέντρα αποφάσεων, τα hidden Markov Models και ο γενετικός προγραμματισμός είναι οι κυριότερες μέθοδοι για τον εντοπισμό γονιδίων, πρόβλημα με το οποίο ασχολούμαστε και εμείς. Οι τρεις πρώτες μέθοδοι αποτελούν και τα δομικά στοιχεία της υπολογιστικής νοημοσύνης[2]. Η υπολογιστική νοημοσύνη αποτελεί ένα από τους σημαντικότερους και ταχύτατα εξελισσόμενους κλάδους της πληροφορικής και ειδικότερα του κλάδου της τεχνητής νοημοσύνης. Σε αντίθεση με τις κλασικές μεθόδους τεχνητής νοημοσύνης, που βασίζονται στη λογική και στη συμβολική επεξεργασία, τα νευρωνικά δίκτυα χρησιμοποιούν αριθμητικά μοντέλα για να αντιμετωπίσουν προβλήματα Τεχνητής Νοημοσύνης. Οι αλγόριθμοι νευρωνικών δικτύων κατατάσσονται στις πιο δημοφιλείς τεχνικές αναζήτησης δεδομένων (data mining) και αυτοεκπαίδευσης μηχανής (machine learning), που χρησιμοποιούνται σήμερα. Καθώς η ταχύτητα των υπολογιστών αυξάνεται συνεχώς, η μεθοδολογία των νευρωνικών δικτύων έχει αρχίσει να αντικαθιστά πολλά παραδοσιακά «εργαλεία» στο χώρο της ανακάλυψης γνώσης (knowledge discovery).

4.2 Εισαγωγικά στοιχεία για τα Τεχνητά Νευρωνικά Δίκτυα

Χρησιμοποιώντας ως πρότυπο τη φύση αναπτύχθηκε το εμπνευσμένο από τη βιολογία υπολογιστικό παράδειγμα των τεχνητών νευρωνικών δικτύων. Ο όρος «τεχνητό» χρησιμοποιείται, διότι οι υπολογισμοί πραγματοποιούνται με τη χρήση ηλεκτρονικών και όχι βιολογικών στοιχείων. Ο εγκέφαλος λειτουργεί σαν μία πολυεπίπεδη δομή (6-7 επίπεδα νευρώνων), της οποίας η διαδικασία επεξεργασίας πληροφορίας έχει τα εξής χαρακτηριστικά:

- Οι υπολογισμοί είναι κατανεμημένοι παράλληλα
- Η ανάπτυξη ενός προγράμματος αντικαθίσταται από τη διαδικασία μάθησης

Έτσι ο εγκέφαλος είναι ικανός να μαθαίνει από την «ανατροφοδότηση» (feedback), από τις πληροφορίες που λαμβάνει, και να αλλάζει το σχέδιό του δημιουργώντας νέες νευρωνικές συνδέσεις μεταξύ των νευρώνων ή αλλάζοντας λειτουργίες των ήδη υπαρχόντων.

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

Εμπνευσμένοι λοιπόν από τα βιολογικά νευρικά συστήματα, πολλοί ερευνητές εξερευνούν την περιοχή των τεχνητών νευρωνικών δικτύων, μία καινοτόμα μη αλγοριθμική προσέγγιση στο πρόβλημα της επεξεργασίας της πληροφορίας. Ο εγκέφαλος μοντελοποιείται ως ένα συνεχούς-χρόνου μη γραμμικό δυναμικό σύστημα με ποικίλες αρχιτεκτονικές διασύνδεσης, το οποίο αναμένεται να μιμείται τους μηχανισμούς του εγκεφάλου και να προσεγγίζει ευφυή συμπεριφορά. Οι διασυνδέσεις υλοποιούνται ως κατανεμημένες αναπαραστάσεις με τη μορφή βαρών μεταξύ ενός μεγάλου αριθμού διασυνδεδεμένων νευρώνων[3].

4.2.1 Ιστορικά στοιχεία για τα Τεχνητά Νευρωνικά Δίκτυα

Η περιοχή των νευρωνικών δικτύων ξεκίνησε με την εργασία των ***McCulloch και Pitts*** το 1943, οι οποίοι μελέτησαν ένα μοντέλο του βασικού κυττάρου του ανθρώπινου εγκεφάλου, το οποίο ονόμασαν «νευρώνα» και σήμερα είναι γνωστό ως νευρώνας ***McCulloch και Pitts***. Το μοντέλο αυτό αποτελείται από μεταβλητές αντιστάσεις και αθροιστικούς ενισχυτές, οι οποίοι παριστούν τις συναπτικές διασυνδέσεις (ή συναπτικά βάρη) που συνδέουν τους νευρώνες μεταξύ τους και τη λειτουργία του σώματος του νευρώνα.

Το επόμενο μεγάλο βήμα στα ΤΝΔ έγινε το 1949 όταν ο ***Hebb*** στο βιβλίο του «*The Organization of Behavior*» διατύπωσε για πρώτη φορά ρητά την έννοια της μάθησης μέσω διαμόρφωσης των συναπτικών βαρών. Ο ***Hebb*** υποστήριξε ότι η συνδετικότητα του ανθρώπινου εγκεφάλου μεταβάλλεται συνεχώς όσο ο οργανισμός μαθαίνει διάφορες εργασίες και οι μεταβολές αυτές δημιουργούν τις νευρωνικές δομές. Αξιοσημείωτες εργασίες πάνω στα νευρωνικά δίκτυα και τη νευρωνική μάθηση είναι η διδακτορική διατριβή του ***Minsk***[4] και η εργασία του ***Gabor*** και των συνεργατών του (1954), στην οποία η μάθηση υλοποιήθηκε τροφοδοτώντας τη μηχανή με δείγματα μιας στοχαστικής ανέλιξης μαζί με τη συνάρτηση στόχου, την οποία έπρεπε να δώσει στην έξοδό της η μηχανή.

Δεκαπέντε χρόνια μετά τη δημοσίευση του κλασικού άρθρου των ***McCulloch και Pitts***[5], ο ***Rosenblatt*** ανέπτυξε το 1958 την έννοια του *Perceptron* ως μία νέα λύση στο πρόβλημα της αναγνώρισης προτύπων και απέδειξε το αντίστοιχο θεώρημα σύγκλισης του αλγορίθμου μάθησης του *Perceptron*[6]. Το 1960 οι ***Widrow και Hoff***[7] θεμελίωσαν τον αλγόριθμο μάθησης ελαχίστων μέσων τετραγώνων (*Least Mean Square / LMS*), τον οποίο χρησιμοποίησαν στο νευρωνικό μοντέλο τους ADALINE (**ADA**ptive **LIN**ear **E**lement).

Ο ***Minsky*** διατύπωσε το 1961[8] το *«πρόβλημα απόδοσης επαίνου»* (*credit assignment problem*) για το πολυστρωματικό *perceptron* υπό καθεστώς ενισχυτικής μάθησης. Το πρόβλημα αυτό, κατά τον συγγραφέα, μπορεί να επεξηγηθεί βάσει ενός πολύ απλού παραδείγματος. Ας υποθέσουμε ότι ένα εκατομμύριο αποφάσεις εμπλέκονται στην επίλυση ενός πολύπλοκου προβλήματος (όπως η νίκη σε μία παρτίδα σκάκι). Θα μπορούσαμε να αποδώσουμε σε κάθε απόφαση το ένα εκατομμυριοστό του επιλυμένου προβλήματος; Ο ***Minsky*** αναφέρει ότι για περισσότερο πολύπλοκα προβλήματα, όπου οι αποφάσεις λαμβάνονται ιεραρχικά και οι αυξήσεις είναι αρκετά μικρές ώστε να διασφαλίζεται πιθανή σύγκλιση, η επίλυση του «προβλήματος απόδοσης επαίνου» θα είχε ως αποτέλεσμα πολύ μικρότερους χρόνους επεξεργασίας.

Η δημοσίευση, το 1969, της μονογραφίας των ***Minsky και Papert***, στην οποία οι συγγραφείς αμφισβήτησαν τη σπουδαιότητα και χρησιμότητα του *Perceptron* ενώ παράλληλα έδωσαν έμφαση στους περιορισμούς που προβάλλει η χρήση του[9], είχε ως αποτέλεσμα τη σχεδόν ολοκληρωτική διακοπή των χρηματοδοτήσεων για έρευνα στα

πεδία της τεχνητής νοημοσύνης και των νευρωνικών δικτύων για μία περίοδο 12-15 ετών. Ολόκληρη τη δεκαετία του ’70, μόνο οι ερευνητές ψυχολογίας και νευρο-επιστήμης συνέχισαν να ενδιαφέρονται για τα νευρωνικά δίκτυα. Το 1973 ο ***Von der Malsburg***[10] ήταν ο πρώτος που κατέδειξε την «αυτοοργάνωση» και ανέπτυξε το αυτοοργανούμενο ΤΝΔ, το οποίο ονόμασε «αυτοοργανούμενη απεικόνιση» (*self - organizing map*).

Τη δεκαετία του ’80 έχουμε τις παρακάτω κύριες συμβολές στην ανάλυση και σχεδίαση ΝΔ:

- Ανάπτυξη της ανταγωνιστικής μάθησης ως μίας νέας αρχής αυτο-οργάνωσης (*Grossberg, 1980*).
- Ανάπτυξη των αναδρομικών ΝΔ **Hopfield** και χρήση της ιδέας της «ενεργειακής συνάρτησης» για την ανάλυσή τους.[11]
- Ανάπτυξη της διαδικασίας «προσομοιωμένης ανόπτησης» (*simulated annealing*) για την επίλυση προβλημάτων βελτιστοποίησης.[12]
- Νέα θεώρηση της ενισχυτικής μάθησης.[13]
- Ανάπτυξη του αλγορίθμου ανάστροφης διάδοσης σφάλματος (*back-propagation algorithm*) από τους ***Rumelhart, Hinton και Williams***.[14]
- Σπουδή των ΝΔ ακτινικών συναρτήσεων βάσης (*radial-basis functions / RBF*) από τους ***Broomhead και Lowe***.[15]
- Ανάμιξη εννοιών και μεθόδων νευροβιολογίας και τεχνολογίας VLSI από τον ***Mead*** στο βιβλίο του: «*Analog VLSI and Neural Systems*» (1989).

Σημειώνεται ότι την πιο σημαντική συμβολή για την αναζωογόνηση της έρευνας στα νευρωνικά δίκτυα μετά τη μονογραφία των ***Minsky και Papert*** είχαν το άρθρο του ***Hopfield*** (1982) και το δίτομο βιβλίο των ***Rumelhart και McLelland.***[16] Τα νευρωνικά δίκτυα έχουν αναμφίβολα φθάσει σε ένα θαυμάσιο επίπεδο ανάπτυξης και βεβαίως θα συνεχίσουν να αναπτύσσονται στο μέλλον ποικιλοτρόπως συνδυαζόμενα με τα ασαφή συστήματα, τους γενετικούς αλγορίθμους και τα έμπειρα συστήματα, αλλά και με άλλες κλασικές τεχνικές ανάλυσης και σχεδίασης συστημάτων απόφασης και ελέγχου. Στον πίνακα που ακολουθεί εμφανίζονται τα κυριότερα μοντέλα νευρωνικών δικτύων καθώς και οι ερευνητές που τα ανέπτυξαν.

Πίνακας 4.1: Κυριότερα μοντέλα ΤΝΔ

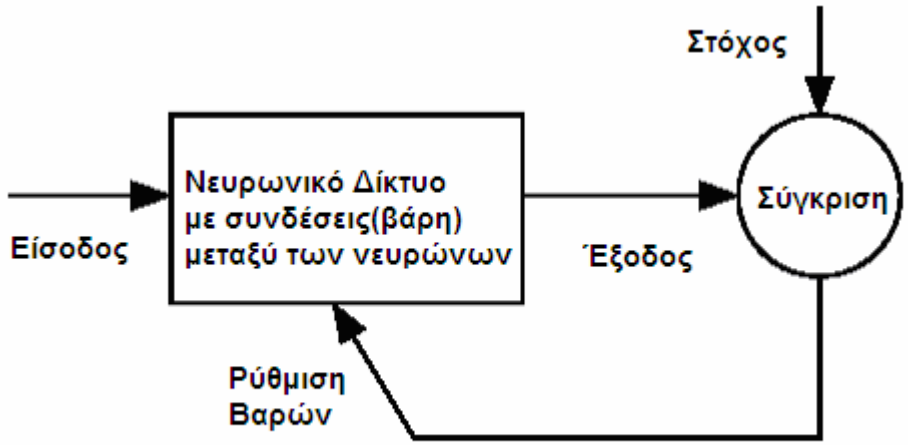
Έτος	Νευρωνικό Δίκτυο	Ερευνητές
1942	Νευρόνιο Mc Culloh / Pitts	Mc Culloh, Pitts
1957	Perceptron	Rosenblatt
1960	Adaline / Madaline	Widrow
1974	Backpropagation (BP)	Werbos, Parker, Rumelhart
1978	Neocognitron	Fukushima
1980	Adaptive Resonance Theory	Kohonen
1980	Self Organizing Map	Kohonen
1982	Νευρωνικό Δίκτυο Hopfield	Hopfield
1985	Μηχανή Boltzmann	Hinton, Sejnowsky
1988	Κυτταρικό Νευρωνικό Δίκτυο	Chua, Yang
1988	Δίκτυα Radial Basis Functions	Broohead, Lowe

4.2.2 Ορισμός του Τεχνητού Νευρωνικού Δικτύου

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

Η συνολική λειτουργικότητα ενός ΤΝΔ καθορίζεται από την τοπολογία του δικτύου, τα χαρακτηριστικά των νευρώνων, τη μέθοδο εκπαίδευσης και από τα δεδομένα με τα οποία γίνεται η εκπαίδευση. Παρότι δεν το απαιτεί ο προηγούμενος ορισμός, ο υπολογισμός που εκτελεί κάθε νευρώνας είναι απλός και κοινός για όλους τους νευρώνες. Επειδή οι νευρώνες λειτουργούν παράλληλα και ο αριθμός τους μπορεί να είναι πολύ μεγάλος, τα ΤΝΔ αποτελούν χαρακτηριστικό παράδειγμα μαζικά παράλληλου υπολογισμού (massively parallel computing).

Στο παράδειγμα που ακολουθεί, παρουσιάζεται πολύ συνοπτικά η λειτουργία ενός νευρωνικού δικτύου. Όπως γίνεται σαφές στο Σχήμα 4.1, αυτό το δίκτυο τροποποιείται σύμφωνα με μία σύγκριση της εξόδου και του στόχου (επιθυμητή έξοδος), μέχρις ότου η έξοδος να γίνει ίση με το στόχο. Στην πράξη, πολλά ζεύγη εισόδου-εξόδου χρησιμοποιούνται στην εκμάθηση με επίβλεψη, για να εκπαιδευτεί δηλαδή ένα δίκτυο.



Σχήμα 4.1: Λειτουργία Νευρωνικού Δικτύου

4.2.3 Βασικές Αρχές Νευρωνικών Δικτύων

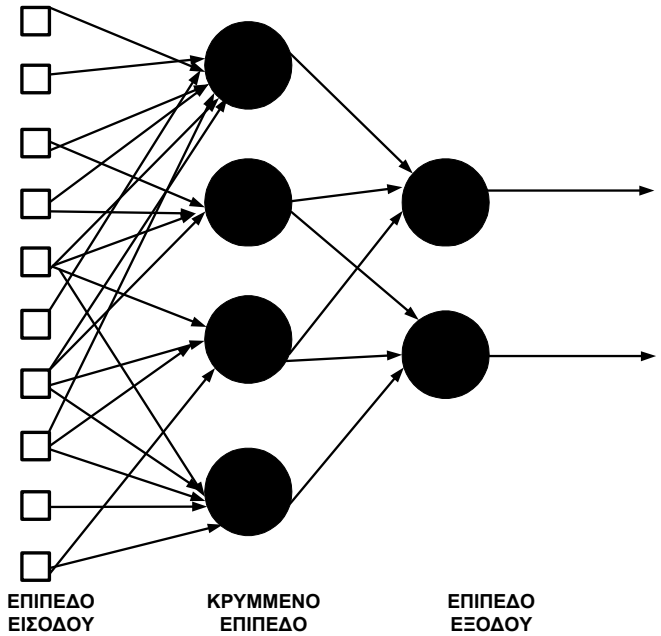
Οι βασικές αρχές των νευρωνικών δικτύων παρουσιάζονται αναλυτικά παρακάτω:

Στα σχήματα 4.2 και 4.3 απεικονίζονται δύο παραδείγματα δικτύων μικρής κλίμακας, στα οποία οι μονάδες παρουσιάζονται με κύκλους και οι συνδέσεις με

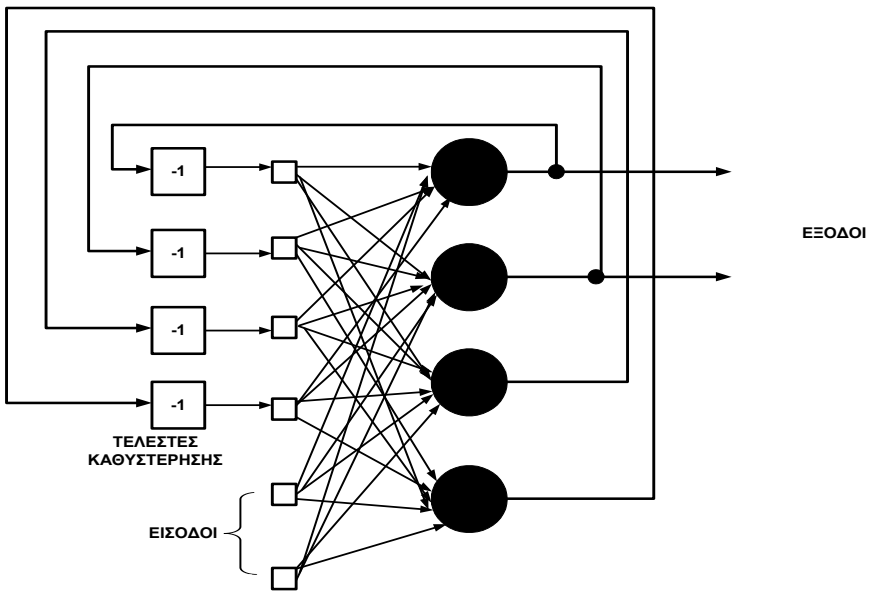
βέλη. Στο σχήμα 2 απεικονίζεται ένα μη-ανατροφοδοτούμενο δίκτυο, το οποίο δηλαδή δεν περιέχει κλειστά μονοπάτια από συνδέσεις. Όπως φαίνεται, οι μονάδες ομαδοποιούνται σε επίπεδα (ή στρώματα-layers). Αντίθετα, στο σχήμα 3 παρουσιάζεται η αρχιτεκτονική της ανατροφοδότησης, η οποία επιτρέπει την ύπαρξη κύκλων από συνδέσεις μεταξύ μονάδων. Αυτή η δεύτερη αρχιτεκτονική προσδίδει στο δίκτυο πολύ περισσότερες δυνατότητες, αλλά είναι πιο δύσκολο να αντιμετωπισθεί μαθηματικά. Επίσης, πρέπει να σημειωθεί ότι η τοπολογία των δικτύων μπορεί να είναι είτε στατική είτε δυναμική (μεταβαλλόμενη). Τέλος, οι μονάδες που δεν έχουν συνδέσεις με τον εξωτερικό κόσμο, λέμε ότι είναι «κρυμμένες» (hidden) ή εσωτερικές.

Κάθε μονάδα υλοποιεί μία συνάρτηση τοπικά και ολόκληρο το δίκτυο υλοποιεί μία συγκεκριμένη λειτουργία. Στις περισσότερες περιπτώσεις, η εφαρμογή είναι αυτή που καθορίζει μέσω των προδιαγραφών της την απαιτούμενη λειτουργία. Ο καθορισμός των παραμέτρων του δικτύου (τιμών και βαρών) που θα ικανοποιούν αυτές τις προδιαγραφές επιτυγχάνεται μέσω της διαδικασίας μάθησης.

Η γνώση, η εμπειρία και η εκπαίδευση του δικτύου αποθηκεύεται στις διασυνδέσεις των μονάδων και στις τιμές των βαρών. Η εκπαίδευση επιτυγχάνεται με την συνεχή τροποποίηση των βαρών.



Σχήμα 4.2. Δίκτυο Πρόσθιας Τροφοδότησης



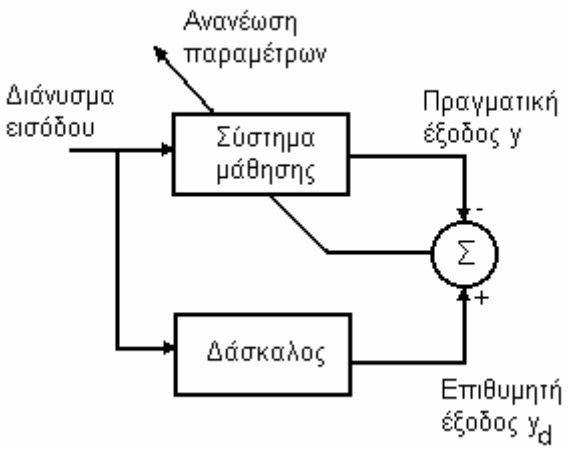
Σχήμα 4.3: Επαναληπτικό Δίκτυο

Τα περισσότερα τεχνητά νευρωνικά δίκτυα εκπαιδεύονται με την ελπίδα ότι θα παρουσιάσουν καλή γενικευτική ικανότητα, όταν θα τους ζητηθεί να αναγνωρίσουν ή να κατηγοριοποιήσουν καινούργια (άγνωστα) δεδομένα-πρότυπα. Αυτός είναι ο αντικειμενικός στόχος της διαδικασίας εκπαίδευσης, να αναπτύξει δηλαδή το ΤΝΔ κατάλληλη εσωτερική δομή, ώστε να αναγνωρίζει πρότυπα που θα μοιάζουν με αυτά με τα οποία εκπαιδεύτηκε. Τα ΤΝΔ εκπαιδεύονται τόσο με τεχνικές μάθησης με επίβλεψη, όσο και με τεχνικές μάθησης χωρίς επίβλεψη.

4.2.4 Εκπαίδευση Τεχνητών Νευρωνικών Δικτύων

Η έννοια της εκπαίδευσης είναι πολύ ευρεία και σε γενικές γραμμές μπορεί να οριστεί ως η κατάλληλη χρήση πληροφοριών για βελτίωση της συμπεριφοράς ενός συστήματος. Στην πιο συνηθισμένη περίπτωση των προβλημάτων απεικόνισης (συσχέτισης προτύπων εισόδου-εξόδου) η εκπαίδευση μπορεί να οριστεί ως η τροποποίηση των παραμέτρων (βαρών) του ΤΝΔ, ώστε χρησιμοποιώντας ένα σύνολο δεδομένων να πλησιάσουμε σταδιακά την επιθυμητή συμπεριφορά συγκρίνοντας την *τρέχουσα απόκριση* του δικτύου με την *επιθυμητή απόκριση* .

Έστω ότι έχουμε ένα σύνολο δεδομένων απεικόνισης εισόδου / εξόδου ή μονάχα εισόδου, τα οποία καθορίζουν την επιθυμητή συμπεριφορά του δικτύου. Το σύνολο αυτό το ονομάζουμε <u>σύνολο εκπαίδευσης Η</u> (*training set*). Στη μάθηση με επίβλεψη (supervised learning) του (*training set*) μας παρέχονται πληροφορίες για τη συσχέτιση των εισόδων *(I)* με τις εξόδους *(O)*, δηλαδή έχουμε ζεύγη της μορφής (I_i, O_i), $i=1,\ldots,n$. Στην περίπτωση αναγνώρισης εικόνας, για παράδειγμα, οι είσοδοι αποτελούνται από pixels ή κάποιες συγκεκριμένες μήτρες. Στο σημείο αυτό θα πρέπει να υπογραμμίσουμε τη σημασία της σωστής επιλογής των δεδομένων εισόδου. Θεωρητικά το νευρωνικό δίκτυο μπορεί να τροφοδοτηθεί με οτιδήποτε, αλλά η λανθασμένη επιλογή των δεδομένων εισόδου εισάγει θόρυβο στο σύστημα και δυσκολεύει τη διαδικασία της εκμάθησης.



Σχήμα 4.4: Δομή της επιβλεπόμενης μάθησης

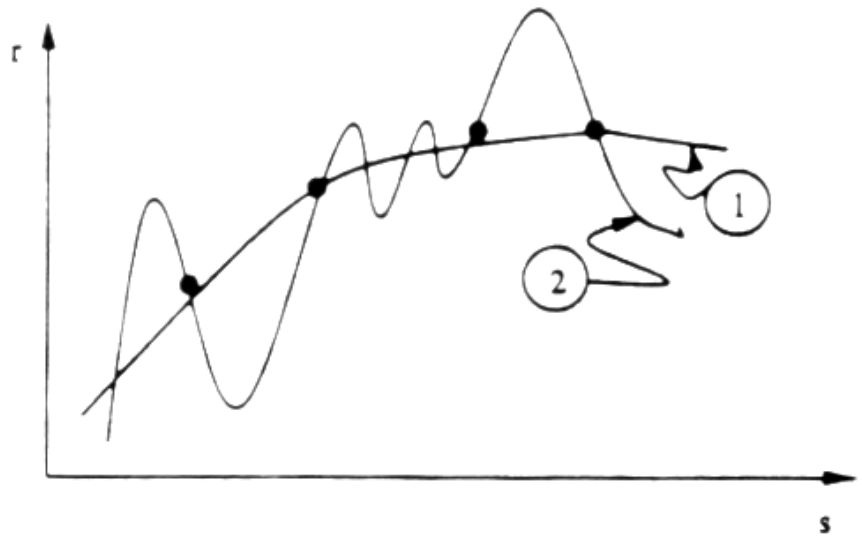
Στο τέλος της εκπαίδευσης, προκειμένου να διαπιστωθεί η ικανότητα γενίκευσης του ΤΝΔ, χρησιμοποιείται ένα άλλο σύνολο δεδομένων, που αποτελείται επίσης από ζεύγη της μορφής (I_i, O_i) και το οποίο ονομάζεται σύνολο ελέγχου Η'.

Τα σύνολα εκπαίδευσης Η και ελέγχου Η' πρέπει να είναι διαφορετικά, γιατί ο έλεγχος του νευρωνικού δικτύου με το ίδιο σύνολο δεδομένων που χρησιμοποιήθηκε κατά την εκπαίδευση δε μας επιτρέπει να γνωρίζουμε αν το δίκτυο έχει καταφέρει τελικά να «προβλέπει» *(να γενικεύει)* την σωστή απάντηση ή αν έχει «απομνημονεύσει» τα επιθυμητά αποτελέσματα. Ένας άλλος παράγοντας, στον οποίο πρέπει να δοθεί ιδιαίτερη σημασία είναι ο κίνδυνος *υπερεκπαίδευσης*. Κατά τη διάρκεια της εκπαίδευσης ενός νευρωνικού δικτύου μπορεί να παρατηρηθεί μείωση του λάθους στο σύνολο εκπαίδευσης, αλλά αύξηση του λάθους στο σύνολο ελέγχου. Αυτό συμβαίνει γιατί το νευρωνικό δίκτυο έχει υπερεκπαιδευτεί και τείνει να απομνημονεύει τις σωστές απαντήσεις παρά να είναι δημιουργικό.

Στη μάθηση χωρίς επίβλεψη τα δεδομένα του Η δεν είναι απεικονίσεις εισόδου / εξόδου αλλά μόνο δεδομένα εισόδου I_i. Σε αυτήν την περίπτωση, το δίκτυο πρέπει να καταλήξει στην εξαγωγή κάποιων βασικών ιδιοτήτων των δεδομένων του Η.

4.2.5 Γενίκευση

Στην περίπτωση της μάθησης με επίβλεψη, μετά το τέλος της εκπαίδευσης το ΤΝΔ θα πρέπει να παρέχει για κάθε είσοδο την αντίστοιχη επιθυμητή έξοδο. Το ερώτημα που προκύπτει είναι κατά πόσον αυτό επιτυγχάνεται για εισόδους διαφορετικές από αυτές του συνόλου εκπαίδευσης. Το ερώτημα αυτό αποτελεί και το στόχο της γενίκευσης. Έστω ότι έχουμε ένα ΤΝΔ που επιτελεί απεικόνιση μιας εισόδου σε μία έξοδο και έχουμε 4 ζευγάρια της μορφής (x, f(x)) για το σύνολο Η. Όπως φαίνεται στο σχήμα που ακολουθεί, η γενίκευση μετά την εκπαίδευση μπορεί να έχει διάφορες μορφές. Παρότι δηλαδή δεν υπάρχει λάθος στην εκπαίδευση, είναι δυνατόν η συνάρτηση απεικόνισης να έχει πολλές διαφορετικές μορφές. Δύο από αυτές τις μορφές παρουσιάζονται στο σχήμα 4.5. Το σύνολο εκπαίδευσης από μόνο του μπορεί να μας οδηγήσει σε οποιαδήποτε από αυτές τις λύσεις. Επομένως, το σύνολο ελέγχου είναι απαραίτητο σε τέτοιες περιπτώσεις, γιατί θα εκτιμήσει την ικανότητα γενίκευσης και θα μας οδηγήσει στην επιλογή του καλύτερου μοντέλου.



Σχήμα 4.5: Καμπύλες Γενίκευσης.

4.2.6 Εφαρμογές

Η διαρκώς αναπτυσσόμενη τεχνολογία των ΤΝΔ περιλαμβάνει ένα ευρύ σύνολο τεχνικών, το οποίο προσφέρει μια πρακτική προσέγγιση, εναλλακτική των συμβατικών και σε πολλές περιπτώσεις παρέχει τη δυνατότητα αντιμετώπισης δύσκολων προβλημάτων. Αυτό επιτυγχάνεται γιατί τα Νευρωνικά Δίκτυα είναι εργαλεία, ικανά να μαθαίνουν από το περιβάλλον τους και να βρίσκουν εξαρτήσεις μη εμφανείς μεταξύ δεδομένων. Πολλά προβλήματα που χαρακτηρίζονται από τον πολυδιάστατο χώρο τους, την πολυπλοκότητά τους και τις άγνωστες συσχετίσεις μεταξύ των μεταβλητών τους, έχουν πολύ καλά αποτελέσματα χρησιμοποιώντας ΤΝΔ. Μερικά από αυτά είναι:

- Προβλήματα αναγνώρισης(π.χ. αναγνώριση φωνής, εικόνας κλπ.).
- Προβλήματα ελέγχου των οποίων τα δεδομένα είναι ελλιπή, ασαφή και στοχαστικά.
- NP-complete προβλήματα τα οποία περιλαμβάνουν προβλήματα δρομολόγησης, αναζήτησης κλπ.

Παρακάτω αναφέρονται ορισμένοι τομείς, στους οποίους παρουσιάζονται αυτά τα προβλήματα και άρα η χρήση της μεθόδου των ΤΝΔ μπορεί να αποβεί ιδιαίτερα χρήσιμη:

- Επεξεργασία εικόνας και μηχανική όραση.
- Επεξεργασία σήματος (ανάλυση και μορφολογία σεισμικού σήματος).
- Αναγνώριση Προτύπων (εξαγωγή χαρακτηριστικών, ανάλυση και κατηγοριοποίηση σήματος radar, αναγνώριση φωνής, κειμένου, ταυτότητας).
- Ιατρική (ανάλυση ηλεκτροκαρδιογραφήματος, ιατρική διάγνωση και επεξεργασία ιατρικής εικόνας).
- Αμυντικά Συστήματα.
- Οικονομία.
- Δυναμικά εξελισσόμενα συστήματα, πρόβλεψη χρονοσειρών.

- Επικοινωνία ανθρώπου-υπολογιστή.

Ειδικότερα στον τομέα της βιοπληροφορικής, ορισμένες από τις εφαρμογές των ΤΝΔ φαίνονται στον παρακάτω πίνακα:

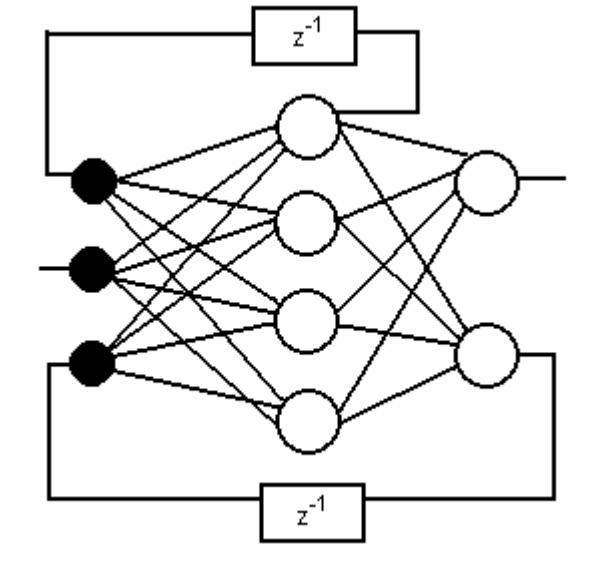
Πίνακας 4.2. Εφαρμογές ΤΝΔ στον τομέα της βιοπληροφορικής

Εφαρμογή	**Χρησιμοποιούμενη Τεχνική**	**Αναφορές**
Πρόβλεψη δευτεροταγούς δομής πρωτεϊνών	Τεχνητά Νευρωνικά Δίκτυα	Holley and Karplus, 1989; Kneller *et al.*, 1990; Qian and Sejnowski, 1988; Rost *et al.*, 1994; Vieth and Kolinski, 1991
Ομαδοποίηση προτύπων γονιδιακής έκφρασης	Τεχνητά Νευρωνικά Δίκτυα	Bicciato *et al.*, 2003; Herrero *et al.*, 2001; Vohradsky, 2001
Προσδιορισμός δομικών ιδιοτήτων νουκλεϊκών οξέων	Τεχνητά Νευρωνικά Δίκτυα	Alvager *et al.*, 1997; Cheng *et al.*, 2001
Προσδιορισμός της θέσης κωδικών περιοχών σε ακολουθίες γονιδιωματικού DNA	Hidden Markov Models Τεχνητά Νευρωνικά Δίκτυα	Alexandersson *et al.*, 2003; Pachter *et al.*, 2002; Pedersen and Hein, 2003; Salzberg *et al.*, 1999; Kulp *et al.*1996; Xu *et. al* 1996
Πρόβλεψη τοπολογίας διαμεμβανικών πρωτεϊνών	Hidden Markov Models	Krogh *et al.*, 2001; Sonnhammer *et al.*, 1998
Υποκινητές σε ακολουθίες DNA	Τεχνητά Νευρωνικά Δίκτυα Δέντρα Αποφάσεων	Opitz *et.al.,* 1997; Hirsh *et. al.,* 1994

4.2.7 Κατηγορίες ΤΝΔ

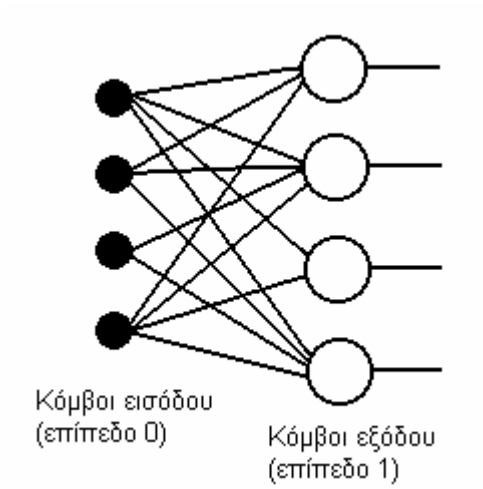
Η αρχιτεκτονική των νευρωνικών δικτύων αφορά την τοπολογική διάταξη αλλά και τη μεθοδολογία δόμησης πολλαπλών νευρώνων. Τα χαρακτηριστικά που καθορίζουν την αρχιτεκτονική ενός ΝΔ είναι το πλήθος των στρωμάτων (*layers*) και οι συνδέσεις ανάμεσα στους νευρώνες. Ένα επιπρόσθετο χαρακτηριστικό, το οποίο σχετίζεται άμεσα με τον τρόπο δόμησης των νευρώνων, είναι ο αλγόριθμος μάθησης που χρησιμοποιείται για την εκπαίδευση του νευρωνικού δικτύου. Τα ΤΝΔ μπορούν να κατηγοριοποιηθούν βάση της τοπολογίας τους στους παρακάτω τύπους:

1. Δίκτυα Ανατροφοδότησης (με συνδέσεις ανάδρασης).



Σχήμα 4.6: Πολυστρωματικό ΝΔ ανατροφοδότησης

2. Δίκτυα πρόσθιας τροφοδότησης.



Σχήμα 4.7: Μονοστρωματικό ΝΔ πρόσθιας τροφοδότησης

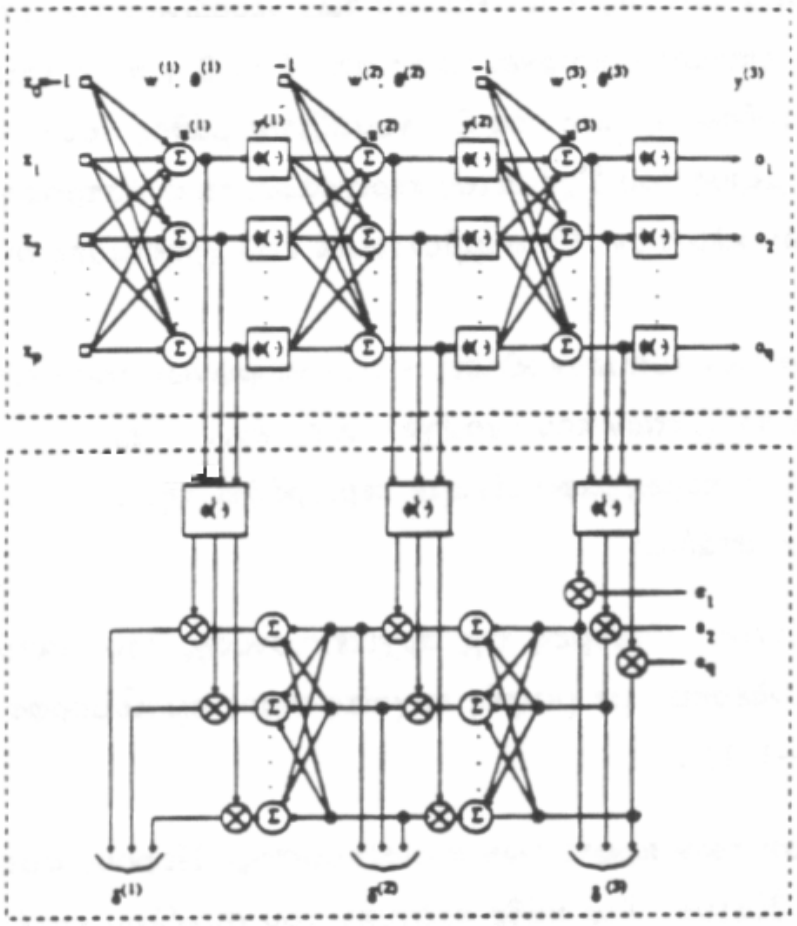
3. Δίκτυα με δομή επιπέδων ή ιεραρχική.
4. Δίκτυα με ανταγωνιστικές συνδέσεις.

4.2.7.1 Ανάστροφη Διάδοση Σφάλματος (Error Back Propagation)

Στην κατηγορία των δικτύων πρόσθιας τροφοδότησης, τα οποία περιλαμβάνουν ένα επίπεδο εισόδου, ένα ή περισσότερα κρυμμένα επίπεδα και ένα επίπεδο εξόδου, εφαρμόζεται η εκπαίδευση με επίβλεψη βάση του αλγορίθμου ***Ανάστροφης Διάδοσης Σφάλματος (Error Back Propagation)***. Ο αλγόριθμος

ανάστροφης διάδοσης είναι γενίκευση του αλγορίθμου *μέσου ελαχίστου τετραγώνου* (*Least Mean Square*) και αποτελεί ίσως τον πιο διαδεδομένο αλγόριθμο επιβλεπόμενης μάθησης.

Ο όρος back propagation εμφανίστηκε μετά το 1985, αλλά η ιδέα περιγράφηκε για πρώτη φορά από τον Werbos στη διδακτορική διατριβή του το 1974. Η ανάπτυξη του αλγορίθμου backpropagation αποτελεί ορόσημο στην εξέλιξη των νευρωνικών δικτύων, διότι παρέχει μια υπολογιστικά αποτελεσματική μέθοδο εκπαίδευσης. Στο Σχήμα 4.8 που ακολουθεί, παρουσιάζεται ένα τέτοιο Νευρωνικό Δίκτυο με χρήση αλγορίθμου Ανάστροφης Διάδοσης Σφάλματος



Σχήμα 4.8: Χρήση αλγορίθμου Ανάστροφης Διάδοσης Σφάλματος

Η διαδικασία εκπαίδευσης στον αλγόριθμο Backpropagation περιλαμβάνει υπολογισμούς που πραγματοποιούνται σε δύο περάσματα μέσω των επιπέδων (στρωμάτων) του δικτύου: ένα πέρασμα κατά την ευθεία φορά (από την είσοδο προς την έξοδο) και ένα πέρασμα κατά την ανάστροφη φορά (από την έξοδο προς την είσοδο).

Στο ευθύ πέρασμα εφαρμόζεται ένα πρότυπο στις εισόδους του δικτύου, πραγματοποιούνται οι υπολογισμοί κατά την ορθή φορά και στο τέλος παράγεται ένα σύνολο από εξόδους που αποτελούν και την πραγματική έξοδο του δικτύου. Κατά τη διάρκεια αυτού του περάσματος, τα βάρη των συνδέσεων του δικτύου παραμένουν σταθερά.

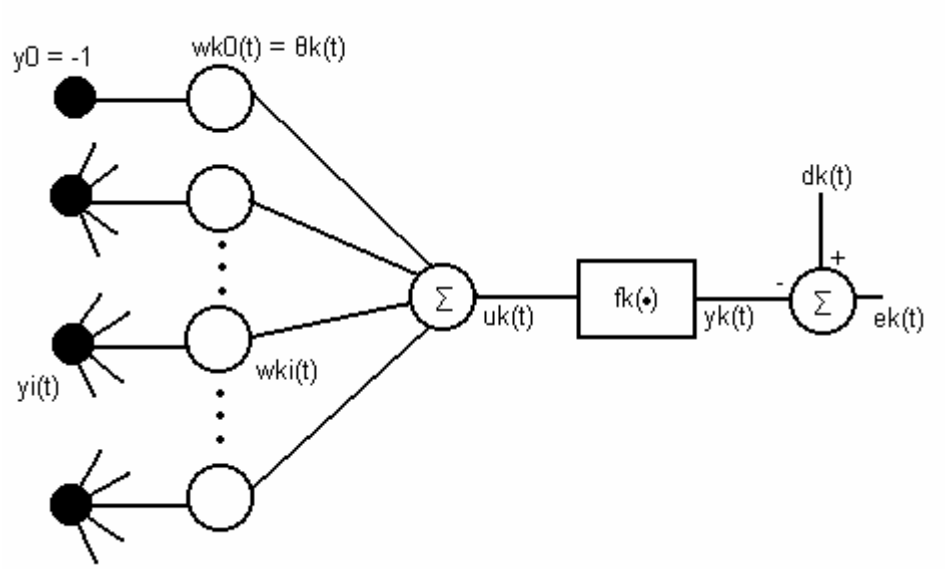
Αντίθετα, κατά τη διάρκεια του αντίστροφου περάσματος τα βάρη προσαρμόζονται σύμφωνα με τον κανόνα διόρθωσης σφάλματος. Συγκεκριμένα, η πραγματική τιμή εξόδου αφαιρείται από την αντίστοιχη επιθυμητή και παράγεται το

σήμα σφάλματος. Στη συνέχεια, αυτό το σήμα προωθείται στο δίκτυο κατά την ανάστροφη κατεύθυνση, δηλαδή από την έξοδο προς την είσοδο (από τη λειτουργία αυτή προέκυψε και το όνομα του αλγορίθμου). Στη διαδικασία αυτή, τα βάρη των συνδέσεων αναπροσαρμόζονται, ώστε να πλησιάσει η απόκριση του δικτύου την επιθυμητή τιμή.

Δηλαδή, προσαρμόζει τα συναπτικά βάρη έτσι, ώστε να ελαχιστοποιηθεί το μέσο τετραγωνικό σφάλμα μεταξύ των επιθυμητών και πραγματικών αποκρίσεων μετά από την παρουσίαση του κάθε προτύπου στην είσοδο του νευρωνικού δικτύου. Η επέκταση από τον αλγόριθμο μέσου ελαχίστου τετραγώνου συνίσταται στη χρήση της πραγματικής εξόδου κάθε νευρώνα μετά την εφαρμογή της συνεχούς συνάρτησης ενεργοποίησης. Έτσι, τα συναπτικά βάρη ενημερώνονται με στόχο την ελαχιστοποίηση του κριτηρίου:

$$E_p(t) = \frac{1}{2}\sum_k e^2{}_k(t) \ , \ e_k(t) = d_k(t)-y_k(t) \qquad (4.2.7.1.1)$$

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



Σχήμα 4.9: Ο k-οστός νευρώνας εξόδου

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

$$u_k = \sum_{i=0}^{n} w_{ki}(t)y_i(t) \qquad (4.2.7.1.2)$$

$$y_k(t) = f_k(u_k(t)) \qquad (4.2.7.1.3)$$

$$e_k(t) = d_k(t)-f_k(u_k(t)) \qquad (4.2.7.1.4)$$

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

$$\Delta w_{ki}(t) = -\gamma\frac{\partial E_p(t)}{\partial w_{ki}(t)} \qquad (4.2.7.1.5)$$

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

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

$= e_k(t)(-1)f'_k(u_k(t))y_i(t)$ (4.2.7.1.6)

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

επομένως τελικά προκύπτει ο γενικευμένος κανόνας δέλτα:

$\Delta w_{ki}(t) = \gamma\delta_k(t)y_i(t)$ (4.2.7.1.7)

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

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

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

$\Delta w_{ki}(t) = \gamma[d_k(t)-y_k(t)]f'_k(u_k(t))$ (4.2.7.1.8)

όπου ο δείκτης k εκτείνεται σε όλους τους νευρώνες εξόδου και ο δείκτης i σε όλους τους νευρώνες του τελευταίου κρυμμένου στρώματος.

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

$$\delta_j(t) = -\frac{\partial E_p(t)}{\partial y_j(t)} \cdot \frac{\partial y_j(t)}{\partial u_j(t)} = \frac{\partial E_p(t)}{\partial y_j(t)} f'_j(u_j(t)) \quad (4.2.7.1.9)$$

όπου
$$\frac{\partial E_p(t)}{\partial y_j(t)} = \sum_m e_m(t)\frac{\partial e_m(t)}{\partial y_j(t)}$$
$$= \sum_m e_m(t)\frac{\partial e_m(t)}{\partial u_m(t)}\frac{\partial u_m(t)}{\partial y_j(t)}$$
$$= \sum_m e_m(t)f'_m(u_m(t))w_{mj}(t) \quad (4.2.7.1.10)$$

και ο δείκτης m εκτείνεται σε όλους τους νευρώνες του στρώματος εξόδου. Η σχέση (3.2.2.1.10) προκύπτει ομοίως με προηγούμενα (σχέσεις (4.2.7.1.1) → (4.2.7.1.4)):

$$E_p(t) = \frac{1}{2}\sum_m e^2{}_m(t)\ ,\ e_m(t) = d_m(t)-y_m(t) \quad (4.2.7.1.11)$$

$$u_m = \sum_{j=0}^{n} w_{mj}(t)y_j(t) \quad (4.2.7.1.12)$$

$$y_m(t) = f_m(u_m(t)) \quad (4.2.7.1.13)$$

όπου n είναι ο συνολικός αριθμός εισόδων. Βάσει της (4.2.7.1.7), η (4.2.7.1.10) γράφεται:

$$\frac{\partial E_p(t)}{\partial y_j(t)} = -\sum_m \delta_m(t)w_{mj}(t) \quad (4.2.7.1.14)$$

άρα το δέλτα του j κρυμμένου νευρώνα θα είναι:

$$\delta_j(t) = f'_j(u_j(t))\sum_m \delta_m(t)w_{mj}(t) \quad (4.2.7.1.15)$$

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

$$\Delta w_{ji}(t) = \gamma\delta_j(t)y_i(t) = \gamma y_i(t) \sum_m \delta_m(t)w_{mj}(t) \qquad (4.2.7.1.16)$$

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

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

   $$u_j = \sum_i w_{ji}(t)y_i(t) \qquad (4.2.7.1.17)$$

   $$y_j(t) = f_j(u_j(t)) \qquad (4.2.7.1.18)$$

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

   $$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t) \qquad (4.2.7.1.19)$$

   όπου το $\Delta w_{ji}(t)$ δίνεται από την (4.2.7.1.8) όταν ο θεωρούμενος νευρώνας ανήκει στο στρώμα εξόδου και από την (4.2.7.1.16) όταν ανήκει σε κρυμμένο στρώμα. Σημειώνεται ότι $w_{ji}(t)$ είναι το συναπτικό βάρος που συνδέει τον νευρώνα j ενός στρώματος με τον νευρώνα i του αμέσως προηγούμενου στρώματος.
5. Επαναλαμβάνεται η όλη διαδικασία από το βήμα 2

4.2.7.2 Χρήσιμες παρατηρήσεις και ορισμός παραμέτρων για τον Αλγόριθμο Ανάστροφης Διάδοσης

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

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

$$\Delta \mathbf{w}^{(\tau)} = -\eta \nabla E|_{\mathbf{w}}^{(\tau)} \qquad (4.2.7.2.1)$$

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

$\Delta \boldsymbol{w}^{(\tau)} = -\eta \nabla E^{n}|_{\boldsymbol{w}}^{(\tau)}$ (4.2.7.2.2)

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

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

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

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

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

$\Delta w_{ji}(t) = \alpha \Delta w_{ji}(t-1) + \eta \delta_j(t) y_i(t)$ (4.2.7.2.3)

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

$$\Delta w_{ji}(t) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t) \qquad (4.2.7.2.4)$$

ισοδύναμα από την σχέση (3.1.8.5) προκύπτει ότι:

$$\Delta w_{ji}(t) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E_p(t)}{\partial w_{ji}(t)} \qquad (4.2.7.2.5)$$

Βάσει αυτής της σχέσης μπορούν να γίνουν ορισμένες παρατηρήσεις.

- Η τρέχουσα μεταβολή $\Delta w_{ji}(t)$ αντιπροσωπεύει το άθροισμα μιας εκθετικής χρονικής ακολουθίας. Για να συγκλίνει αυτή η ακολουθία, πρέπει η σταθερά *α* να ανήκει στο διάστημα $0 \le |\alpha| < 1$. Όταν α = 0 ο αλγόριθμος λειτουργεί χωρίς *momentum*. Η σταθερά *α* μπορεί να παίρνει και αρνητικές τιμές, αν και αυτό δεν συνηθίζεται.
- Όταν η μερική παράγωγος $\frac{\partial E_p(t)}{\partial w_{ki}(t)}$ έχει το ίδιο πρόσημο σε συνεχόμενες επαναλήψεις, τότε το $\Delta w_{ji}(t)$ μεγαλώνει σημαντικά και το βάρος $w_{ji}(n)$ μεταβάλλεται σε μεγάλο βαθμό. Επομένως η εισαγωγή του παράγοντα ορμής τείνει να επιταχύνει σε σταθερή κατεύθυνση την κάθοδο.
- Όταν η μερική παράγωγος $\frac{\partial E_p(t)}{\partial w_{ki}(t)}$ έχει αντίθετο πρόσημο σε συνεχόμενες επαναλήψεις, τότε το $\Delta w_{ji}(t)$ μειώνεται σημαντικά και το βάρος $w_{ji}(n)$ μεταβάλλεται σε μικρό βαθμό. Επομένως η εισαγωγή του παράγοντα ορμής έχει σταθεροποιητικό αποτέλεσμα.

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

4.2.8 Πλεονεκτήματα και μειονεκτήματα των ΤΝΔ

Τα πλεονεκτήματα και τα μειονεκτήματα των νευρωνικών δικτύων σε σχέση με τις κλασικές μεθόδους υπολογισμού δεν είναι πλήρως καθορισμένα, γιατί τα ΤΝΔ αποτελούν ένα σχετικά πρόσφατο μοντέλο υπολογισμού. Οι προσδοκίες είναι μεγάλες αν και η αντίληψη ότι τα ΤΝΔ μπορούν να λύσουν όλα τα προβλήματα αυτόματης απόκτησης γνώσης είναι μη ρεαλιστική. Κάποια από τα πλεονεκτήματα και τα μειονεκτήματα που έχουν ήδη αποσαφηνισθεί είναι τα ακόλουθα:

Πλεονεκτήματα:

- Έμφυτος παραλληλισμός.
- Ανοχή σε βλάβες (λόγω του παραλληλισμού).
- Σχεδιασμός ώστε να είναι προσαρμοσμένα.
- Δεν υπάρχει η ανάγκη για χαρακτηρισμό του προβλήματος πέρα από το σύνολο των προτύπων εκπαίδευσης.

Μειονεκτήματα:

- Δεν υπάρχουν σαφείς κανόνες για την ανάπτυξη ενός ΤΝΔ για οποιαδήποτε εφαρμογή.

- Δεν υπάρχει γενικός τρόπος ερμηνείας της εσωτερικής λειτουργίας του δικτύου.
- Η εκπαίδευση μπορεί να είναι δύσκολη ή πολύ χρονοβόρα.
- Η ικανότητα γενίκευσης είναι δύσκολα προβλέψιμη.

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

4.3 Εισαγωγικά στοιχεία για την ασαφή λογική και τα ασαφή συστήματα

Η έννοια της ασάφειας συναντιέται σε πολλούς τομείς της ζωής μας και αποτελεί ακόμα περισσότερο ένα χαρακτηριστικό της γλώσσας, αφού η ανακρίβεια που ενυπάρχει στη συντριπτική πλειοψηφία των (γλωσσικών) ορισμών και τη χρήση εννοιών και συμβόλων θεωρείται δεδομένη. Η θεωρία της ασαφούς λογικής βρήκε αρχικά μεγάλη αντίδραση από τους οπαδούς της κλασσικής (Αριστοτελικής) λογικής και τους πιθανοθεωρητικούς επιστήμονες, έφτασε σήμερα όμως στο σημείο να αποτελεί ένα από τα ισχυρότερα καθολικά εργαλεία λήψης αποφάσεων και ανάπτυξης αλγορίθμων ελέγχου και έμπειρων συστημάτων μέσα σε αβεβαιότητα. Μαθηματικά ο όρος ασάφεια (fuzziness) σημαίνει πολλαπλότητα τιμών και πηγάζει από την αρχή αβεβαιότητας του Heisenberg.

4.3.1 Ιστορικά στοιχεία για την ασαφή λογική

Η θεωρία ασαφών συνόλων θεμελιώθηκε στην παρούσα της μορφή από το μηχανικό αυτομάτου ελέγχου Lofti Zadeh to 1965. Η ιδέα όμως της χρήσης ασαφών μαθηματικών δομών έχει χρησιμοποιηθεί από τον Poincare τον 19° αιώνα στο πλαίσιο της "οπτικής αντίληψης". Η ιδέα αυτή δεν αναπτύχθηκε μέχρι τις αρχές της δεκαετίας του '70 όπου χρησιμοποιήθηκε από τον Zeeman. Η ασαφής γεωμετρία του Zeeman, που έχει πολύ μικρή σχέση με τη θεωρία του Zadeh, βρήκε εφαρμογή σε ορισμένες περιοχές της Κβαντικής Θεωρίας, αλλά σε σύγκριση με τη θεωρία του Zadeh η διάδοσή της ήταν πολύ περιορισμένη. Η θεωρία του Zadeh βρήκε εφαρμογές στην τεχνολογία, στην οργάνωση επιχειρήσεων, την ψυχολογία, αλλά και σε εξωτικές περιοχές όπως η "σημειοτική" (semiotics). Τα ασαφή σύνολα και η ασαφής συλλογιστική χρησιμοποιήθηκαν για την επίλυση πολλών πρακτικών προβλημάτων από την τιμολόγηση προϊόντων μέχρι τον αυτόματο έλεγχο της τσιμεντοβιομηχανίας και των τραίνων. Έτσι μέσα σε 25-30 χρόνια το νέο αυτό μαθηματικό εργαλείο χειρισμού της αβεβαιότητας είχε μια σημαντική επίδραση στη ζωή του ανθρώπου. Μερικές πρόσφατες εξελίξεις στη θεωρία των ασαφών συνόλων είναι χρήση της στην τεχνολογία γνώσης, την ψυχολογία αντίληψης και σε διάφορες εφαρμογές της επιστήμης και τεχνολογίας. Ασαφή έμπειρα συστήματα υπάρχουν σήμερα με επιτυχή λειτουργία σε ποικιλία πεδίων όπως η ιατρική και η πρόβλεψη των σεισμών.

4.3.2 Βασικές έννοιες και ορισμοί στα ασαφή σύνολα

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

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

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

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

Δεδομένου ότι ένα κλασικό σύνολο Χ είναι ένα σύνολο με σαφή όρια, τότε αν Α ένα υποσύνολο του Χ, το Α καλείται ασαφές υποσύνολο του Χ του υπερσυνόλου αναφοράς Α τότε και μόνο τότε αν:

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

όπου μ_A μία συνάρτηση από το Χ στο [0,1] η οποία ονομάζεται συνάρτηση συμμετοχής. Όπως προκύπτει από τον ορισμό της, η συνάρτηση συμμετοχής (ΣΜ) καθορίζει ένα ασαφές σύνολο πλήρως. Επειδή τα περισσότερα ασαφή σύνολα που χρησιμοποιούνται έχουν ως υπερσύνολο αναφοράς το σύνολο των πραγματικών αριθμών, θα ήταν ανέφικτο να καταγραφούν όλα τα ζευγάρια που καθορίζουν μια ΣΜ. Ένας πιο συνοπτικός τρόπος για να οριστεί μια ΣΜ είναι να εκφραστεί με το μαθηματικό της τύπο. Με αυτόν τον τρόπο οι κυριότερες συναρτήσεις συμμετοχής είναι: η τριγωνική, η τραπεζοειδής και η γκαουσσιανή. Παράλληλα, τα ασαφή σύνολα αντίστοιχα με τα κλασσικά σύνολα έχουν παρόμοιες πράξεις[17].

4.3.3 Ασαφής Συλλογιστική

Βάσει των ασαφών συνόλων η ασαφής, ή αλλιώς προσεγγιστική, συλλογιστική κωδικοποιεί τον μηχανισμό με τον οποίο μπορούν να εξαχθούν ασαφή ή προσεγγιστικά συμπεράσματα. Υπάρχουν τρεις βασικές έννοιες σε αυτήν:

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

'Χ είναι Α'

- ✓ Ο γενικευμένος κανόνας του Θέτειν και του Αναιρείν. Αντίστοιχα με την «λογική ταυτολογία» που ισχύει στην κλασσική λογική και τις κυριότερες από αυτές που είναι οι Modus Ponens (κανόνας του Θέτειν), Modus tollens (κανόνας του Αναιρείν) μπορούν να προκύψουν οι γενικευμένοι κανόνες του θέτειν και του αναιρείν, οι οποίοι συμπεριλαμβάνουν και τα ασαφή σύνολα. Ο τροποποιημένος (ασαφής) κανόνας Modus Ponens (MP) που αναφέρεται και ως **Generalized Modus Ponens (GMP) (γενικευμένος κανόνας του θέτειν**) έχει την ακόλουθη δομή:

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

  όπου τα A, B, A′ και B′ είναι ασαφή σύνολα. Αντίστοιχα ο **γενικευμένος (ασαφής) κανόνας του αναιρείν** (**Generalized Modus Tollens GMT**) έχει τη μορφή:

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

- ✓ Οι ασαφείς σχέσεις. Οι ασαφείς σχέσεις αποτελούν γενίκευση των συνήθων (κοφτών) σχέσεων και μας δίνουν τη δυνατότητα να χειρισθούμε προβλήματα στα οποία υπάρχει αβεβαιότητα ή αμφιβολία. Από μαθηματική σκοπιά οι ασαφείς σχέσεις χρησιμοποιούνται για τη μοντελοποίηση (παράσταση) ασαφών συνεπαγωγών καθώς και τη συλλογιστική με αυτές. Βρίσκουν εφαρμογή σε όλες τις επιστημονικές περιοχές ασαφούς / προσεγγιστικής συλλογιστικής. Οι κοφτές (συνήθεις) σχέσεις αποτελούν στοιχεία του Καρτεσιανού γινομένου ορισμένων χώρων. Με τον όρο «ασαφής σχέση» εννοούμε ένα ασαφές σύνολο στο Καρτεσιανό γινόμενο $X \times Y = \{(x,y)/ x\in X, y\in Y\}$ το οποίο σύνολο χαρακτηρίζεται από τη συνάρτηση συμμετοχής μ_R:

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

  Στην περίπτωση όπου X =Y έχουμε μια ασαφή σχέση επί του X. Η συνάρτηση συμμετοχής $\mu_R(x ,y)$ παριστά για κάθε ζεύγος (x ,y) το βαθμό σύνδεσης ανάμεσα στα x και y. Οι πράξεις των ασαφών σχέσεων ορίζονται κατά αναλογία των πράξεων των ασαφών συνόλων.

4.3.4 Ασαφή Συστήματα

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

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

1. Μία βάση ασαφών κανόνων της μορφής EAN-TOTE (Ασαφής βάση Γνώσης).

2. Μία ασαφή συλλογιστική μηχανή (μηχανισμό εξαγωγής ασαφών συμπερασμάτων).
3. Μία μονάδα ασαφοποίησης (ασαφοποιητική μονάδα διεπαφής), η οποία μετατρέπει τα δεδομένα εισόδου σε ασαφή σύνολα.
4. Μία μονάδα απο-ασαφοποίησης (από-ασαφοποιητική μονάδα διεπαφής) η οποία μετατρέπει τα ασαφή συμπεράσματα / αποφάσεις σε σαφώς καθορισμένη μορφή.

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

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

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

- Μετράει (παραλαμβάνει) τις (μη ασαφείς) τιμές των εισόδων του συστήματος
- Απεικονίζει τις περιοχές μεταβολής των τιμών εισόδου σε κατάλληλα υπερσύνολα αναφοράς
- Ασαφοποιεί τις εισερχόμενες τιμές των εισόδων, δηλαδή τις μετατρέπει σε ασαφή γλωσσική μορφή.

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

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

Τα ασαφή συστήματα βρίσκουν και στον χώρο της βιοπληροφορικής αρκετά μεγάλη εφαρμογή ιδιαίτερα στην δημιουργία υβριδικών μοντέλων σε συνδυασμό κυρίως με νευρωνικά δίκτυα (Neuro-Fuzzy Networks)[18].

4.4 Εισαγωγικά στοιχεία για τους Γενετικούς Αλγόριθμους

Τα τελευταία τριάντα χρόνια παρατηρείται ένα συνεχώς αυξανόμενο ενδιαφέρον για την ανάπτυξη συστημάτων επίλυσης προβλημάτων βασισμένα στις αρχές της Φυσικής Εξέλιξης[19]. Τα συστήματα αυτού του είδους λειτουργούν διατηρώντας έναν πληθυσμό κωδικοποιημένων πιθανών λύσεων του προβλήματος που προσπαθούμε να επιλύσουμε και εφαρμόζοντας πάνω σε αυτόν διάφορες διαδικασίες εμπνευσμένες από τη βιολογική εξέλιξη. Έτσι, περνώντας από γενιά σε γενιά, τα συστήματα αυτά δημιουργούν συνεχώς νέους πληθυσμούς πιθανών λύσεων εξελίσσοντας τους προηγούμενους πληθυσμούς. Η βασική ιδέα που κρύβεται πίσω από τους ΓΑ είναι η μίμηση των μηχανισμών της βιολογικής εξέλιξης που απαντώνται στη φύση. Βασίζονται στην έννοια του πληθυσμού ατόμων, στα οποία εφαρμόζονται οι γενετικοί τελεστές της διασταύρωσης, μετάλλαξης και αντιστροφής και λειτουργούν υπακούοντας στην αρχή της φυσικής επιλογής του Δαρβίνου («επιβίωση του ισχυρότερου»).

Μία διαδικασία εξέλιξης που εφαρμόζεται πάνω σε έναν πληθυσμό αντιστοιχεί σε μία εκτενή διερεύνηση στον χώρο των πιθανών λύσεων. Απαραίτητη προϋπόθεση για την επιτυχημένη έκβαση μίας τέτοιας διερεύνησης αποτελεί η εξισορρόπηση δύο διαδικασιών που είναι προφανώς αντικρουόμενες, της εκμετάλλευσης και διατήρησης των καλύτερων λύσεων και της όσο το δυνατόν καλύτερης διερεύνησης ολόκληρου του διαστήματος.

Οι ΓΑ διατηρούν έναν πληθυσμό πιθανών λύσεων του προβλήματος που μας ενδιαφέρει, τον οποίο επεξεργάζονται, σε αντίθεση με άλλες μεθόδους αναζήτησης που επεξεργάζονται ένα μόνο σημείο του διαστήματος αναζήτησης. Έτσι, ένας ΓΑ πραγματοποιεί αναζήτηση σε πολλές κατευθύνσεις και υποστηρίζει καταγραφή και ανταλλαγή πληροφοριών μεταξύ αυτών των κατευθύνσεων. Ο πληθυσμός υφίσταται μία προσομοιωμένη γενετική εξέλιξη. Σε κάθε γενιά, οι σχετικά «καλές» λύσεις αναπαράγονται ενώ οι σχετικά «κακές» απομακρύνονται. Ο διαχωρισμός και η αποτίμηση των διαφόρων λύσεων γίνεται με τη βοήθεια μίας αντικειμενικής συνάρτησης ή συνάρτησης ποιότητας (*objective* ή *fitness function*), η οποία παίζει το ρόλο του περιβάλλοντος μέσα στο οποίο εξελίσσεται ο πληθυσμός.

Μέσω των ΓΑ, έννοιες από τον χώρο της Βιολογίας μεταφέρονται στον χώρο επιστημονικών και τεχνικών προβλημάτων διατηρώντας μία αναλογία στις ιδιότητές τους[20]. Έτσι, η έννοια του πληθυσμού αναφέρεται στο σύνολο των υποψήφιων λύσεων του προβλήματος της επιλογής χαρακτηριστικών. Το χρωμόσωμα, το οποίο για τη Βιολογία αντιστοιχεί σε μία αλυσίδα DNA, αναφέρεται σε μία υποψήφια λύση του προβλήματος, η οποία κωδικοποιείται με μία ακολουθία δυαδικών ψηφίων. Το γονίδιο αναφέρεται σε ένα τμήμα της ακολουθίας δυαδικών ψηφίων του χρωμοσώματος. Ο γονότυπος αναφέρεται στη διάταξη των ψηφίων στο χρωμόσωμα που αντιστοιχεί στην υποψήφια λύση του προβλήματος, ενώ ο φαινότυπος, όταν αυτός αναφέρεται, στην ίδια τη λύση (π.χ. αλυσίδα που κωδικοποιεί ένα ΝΔ και το ίδιο το ΝΔ). Επίσης, όπως στη Βιολογία η προσαρμογή ή ποιότητα ενός οργανισμού (*fitness*) ορίζεται τυπικά ως η πιθανότητα που έχει ο οργανισμός να αναπαραχθεί, στους ΓΑ η προσαρμογή ή ποιότητα μίας λύσης είναι η πιθανότητά της να επικρατήσει μέσα σε ένα ανταγωνιστικό περιβάλλον. Από τα άτομα του πληθυσμού, τα οποία διασταυρώνονται μεταξύ τους αλλά και μεταλλάσσονται, προκύπτουν οι απόγονοι, οι οποίοι συμμετέχουν σε νέες διασταυρώσεις, υπόκεινται σε νέες μεταλλάξεις και δίνουν νέους απογόνους. Τελικά, μετά την ολοκλήρωση ενός αριθμού γενεών προκύπτει ο απόγονος που αντιστοιχεί στη βέλτιστη λύση του προβλήματος.

Οι γενετικοί αλγόριθμοι (*Genetic Algorithms*) είναι ένα παράδειγμα τέτοιου συστήματος που μαζί με τον εξελικτικό προγραμματισμό (*Evolutionary Programming*), τις στρατηγικές εξέλιξης (*Evolution Strategies*), τα συστήματα ταξινόμησης (*Classifier Systems*) και το γενετικό προγραμματισμό (*Genetic Programming*) αποτελούν μία κατηγορία συστημάτων επίλυσης προβλημάτων που είναι ευρύτερα γνωστή με τον όρο Εξελικτικοί Αλγόριθμοι (*Evolutionary Algorithms*).

4.4.1 Ιστορικά στοιχεία για τους Γενετικούς Αλγόριθμους

Η πρώτη εμφάνιση των Γενετικών Αλγορίθμων (ΓΑ) χρονολογείται στις αρχές του 1950, όταν διάφοροι επιστήμονες από τον χώρο της βιολογίας αποφάσισαν να χρησιμοποιήσουν υπολογιστές στην προσπάθειά τους να προσομοιώσουν πολύπλοκα βιολογικά συστήματα. Το 1966 οι Fogel, Owens και Walsh ανέπτυξαν τον

καλούμενο *"εξελικτικό προγραμματισμό"* (evolutionary programming), μια τεχνική στην οποία οι υποψήφιες λύσεις σε δεδομένα προβλήματα (καθήκοντα ή ενέργειες) παραστάθηκαν ως μηχανές πεπερασμένων καταστάσεων, οι οποίες εξελίχθηκαν μεταλλάσοντας (mutating) τυχαία τα διαγράμματα μετάπτωσης των καταστάσεών τους και επιλέγοντας το "πλέον ταιριαστό" (fittest). Η συστηματική τους ανάπτυξη όμως, που οδήγησε στη μορφή με την οποία είναι γνωστοί σήμερα, πραγματοποιήθηκε στις αρχές του 1970 από τον John Holland[21] και τους συνεργάτες του στο Πανεπιστήμιο του Michigan, αρχικά για τη μελέτη του φαινομένου της «φυσικής προσαρμογής» και στη συνέχεια για την επίλυση πρακτικών προβλημάτων βελτιστοποίησης.

4.4.2 Βασικός Γενετικός Αλγόριθμος

Η απλούστερη μορφή ΓΑ περιλαμβάνει τρεις τελεστές: Επιλογή, διασταύρωση και μετάλλαξη. Οι τελεστές αυτοί περιγράφονται παρακάτω:

- ✓ Επιλογή. Ο τελεστής αυτός επιλέγει χρωμοσώματα στον πληθυσμό προς αναπαραγωγή. Όσο μεγαλύτερη είναι η ποιότητα του χρωμοσώματος, τόσο αυξάνεται η πιθανότητά του να επιλεγεί περισσότερες φορές προς αναπαραγωγή.
- ✓ Διασταύρωση. Ο τελεστής αυτός επιλέγει τυχαία μία θέση και ανταλλάσσει τις υποακολουθίες πριν και μετά από αυτή τη θέση μεταξύ δύο χρωμοσωμάτων για να παραγάγει δύο απογόνους. Ο τελεστής διασταύρωσης μιμείται χονδρικά τη βιολογική διασταύρωση μεταξύ δύο απλοειδών (μονού χρωμοσώματος) οργανισμών.
- ✓ Μετάλλαξη. Ο τελεστής αυτός, ο οποίος εφαρμόζεται μετά τη διασταύρωση, αλλάζει τυχαία μερικά από τα ψηφία ενός χρωμοσώματος. Η μετάλλαξη μπορεί να λάβει χώρα σε κάθε θέση του χρωμοσώματος με κάποια πιθανότητα, η οποία είναι συνήθως πολύ μικρή (π.χ. 0.001).

Ας δούμε λίγο αναλυτικότερα τα παραπάνω στον τρόπο λειτουργίας του ΓΑ: Κατά τη διάρκεια της γενιάς t, ο ΓΑ διατηρεί έναν πληθυσμό P(t) από n πιθανές λύσεις (*individuals*):

$$P(t) = \{x_1(t), \ldots, x_n(t)\}$$

Κάθε άτομο (*individual*) $x_i(t)$ αποτιμάται και δίνει ένα μέτρο της καταλληλότητας και ορθότητάς του. Αφού ολοκληρωθεί η αποτίμηση όλων των μελών του πληθυσμού, δημιουργείται ένας νέος πληθυσμός (γενιά t+1), ο οποίος προκύπτει από την επιλογή των πιο κατάλληλων στοιχείων του πληθυσμού της προηγούμενης γενιάς. Μερικά μέλη από τον καινούριο αυτό πληθυσμό υφίστανται αλλαγές με την βοήθεια των γενετικών διαδικασιών της διασταύρωσης και της μετάλλαξης σχηματίζοντας νέες πιθανές λύσεις. Η διασταύρωση συνδυάζει τα στοιχεία των χρωμοσωμάτων δύο γονέων για να δημιουργήσει δύο νέους απογόνους ανταλλάσσοντας κομμάτια από τους γονείς. Για παράδειγμα, έστω ότι δύο γονείς αναπαριστώνται με χρωμοσώματα πέντε γονιδίων (a_1, b_1, c_1, d_1, e_1) και (a_2, b_2, c_2, d_2, e_2) αντίστοιχα, τότε οι απόγονοι που θα προκύψουν από διασταύρωση με σημείο διασταύρωσης (*crossover point*) το σημείο 2 είναι οι (a_1, b_1, c_2, d_2, e_2) και (a_2, b_2, c_1, d_1, e_1). Διαισθητικά, μπορούμε να πούμε ότι η διασταύρωση εξυπηρετεί την ανταλλαγή πληροφοριών μεταξύ διαφορετικών πιθανών λύσεων. Η διαδικασία της

μετάλλαξης, όπως είπαμε και παραπάνω, αλλάζει αυθαίρετα ένα ή περισσότερα γονίδια ενός συγκεκριμένου χρωμοσώματος. Πραγματοποιείται με τυχαία αλλαγή γονιδίων με πιθανότητα ίση με το ρυθμό μετάλλαξης (*mutation rate*). Για παράδειγμα, έστω ότι ένα άτομο (*individual*) αναπαρίσταται με το διάνυσμα πέντε διαστάσεων (a_1, b_1, c_1, d_1, e_1). Τότε το άτομο που θα προκύψει με μετάλλαξη στη δεύτερη και στην τέταρτη διάσταση είναι το (a_1, b_1*, c_1, d_1*, e_1). Διαισθητικά, μπορούμε να πούμε ότι η μετάλλαξη εξυπηρετεί την εισαγωγή νέων πιθανών λύσεων, διαφορετικών από τις υπάρχουσες, στον ήδη υπάρχοντα πληθυσμό.

Η πιθανότητα διασταύρωσης P_δ έχει συνήθως τιμή μεταξύ 0.6 και 1.0. Αντίστοιχα, η πιθανότητα μετάλλαξης P_μ είναι μικρότερη από 0.1. Η επιλογή των παραμέτρων P_δ και P_μ απαιτεί την επίλυση ενός πολύπλοκου προβλήματος βελτιστοποίησης.

Ενδεικτικές προτεινόμενες τιμές της πιθανότητας διασταύρωσης P_δ και της πιθανότητας μετάλλαξης P_μ είναι οι εξής :

Για μεγάλους πληθυσμούς:
Ρυθμός διασταύρωσης: P_δ =0.6
Ρυθμός μετάλλαξης: P_μ =0.001

Για μικρούς πληθυσμούς:
Ρυθμός διασταύρωσης: P_δ = 0.9
Ρυθμός μετάλλαξης: P_μ =0.01

Ο εξελικτικός κύκλος επαναλαμβάνεται μέχρι να φθάσει σε κάποιο επιθυμητό κριτήριο τερματισμού. Το όριο αυτό μπορεί να είναι ο αριθμός των γενεών ή το πλήθος των μεταβολών των ατόμων ανάμεσα στις διάφορες γενιές ή μία προκαθορισμένη τιμή της συνάρτησης ποιότητας.

Συνοψίζοντας, μπορούμε να πούμε ότι ένας ΓΑ που χρησιμοποιείται για την επίλυση ενός συγκεκριμένου προβλήματος πρέπει να αποτελείται από τα παρακάτω πέντε στοιχεία:

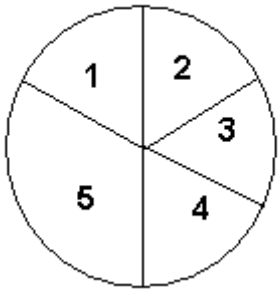
1. Μία γενετική αναπαράσταση των πιθανών λύσεων του προβλήματος.
2. Έναν τρόπο δημιουργίας ενός αρχικού πληθυσμού από πιθανές λύσεις (αρχικοποίηση).
3. Μία αντικειμενική συνάρτηση (*fitness function*) αξιολόγησης των μελών του πληθυσμού, η οποία παίζει τον ρόλο του περιβάλλοντος.
4. Γενετικούς τελεστές για τη δημιουργία νέων μελών (λύσεων).
5. Τιμές για τις διάφορες παραμέτρους που χρησιμοποιεί ο ΓΑ (μέγεθος πληθυσμού, πιθανότητες εφαρμογής των γενετικών τελεστών, κλπ.).

4.4.3 Επιλογή «τροχού ρουλέτας»

Ένα από τα συνηθέστερα σχήματα επιλογής γονέων είναι ο μηχανισμός «τροχού ρουλέτας», του οποίου τα βήματα περιγράφονται παρακάτω:

1. Αθροίζονται οι τιμές της συνάρτησης ποιότητας όλων των μελών του πληθυσμού για να βρεθεί η ολική τιμή της $f_{ολικό}$.
2. Επιλέγεται ένας τυχαίος αριθμός (n) μεταξύ 0 και $f_{ολικό}$.
3. Βρίσκεται το πρώτο μέλος του πληθυσμού, του οποίου η τιμή της συνάρτησης ποιότητας, προστιθέμενη στις τιμές της συνάρτησης ποιότητας των προηγούμενων μελών του πληθυσμού υπερβαίνει το n ή είναι ίση με αυτό.

Για παράδειγμα, στο παρακάτω σχήμα η περιφέρεια του τροχού της ρουλέτας είναι ίση με το $f_{ολικό}$ των πέντε χρωμοσωμάτων. Το χρωμόσωμα 5 είναι το πιο ποιοτικό (*fittest*) και καταλαμβάνει τον μεγαλύτερο τομέα του κύκλου, ενώ το χρωμόσωμα 1 είναι το λιγότερο ποιοτικό χρωμόσωμα και καταλαμβάνει τον μικρότερο τομέα στον τροχό. Για να επιλεχθεί ένα χρωμόσωμα, παράγεται ένας τυχαίος αριθμός στο διάστημα $[0, f_{ολικό}]$ και επιλέγεται το χρωμόσωμα του οποίου ο τομέας, προστιθέμενος στους τομείς των προηγούμενων χρωμοσωμάτων, καλύπτει τον τυχαίο αυτό αριθμό.



Σχήμα 4.10: Επιλογή γονέων με το μηχανισμό της ρουλέτας

4.4.4 Πλεονεκτήματα και μειονεκτήματα των Γενετικών Αλγόριθμων

Οι γενετικοί αλγόριθμοι και γενικότερα ο εξελικτικός προγραμματισμός αναπτύσσεται ιδιαίτερα έντονα τα τελευταία χρόνια, ενώ σε πολλές περιπτώσεις ειδικά για προβλήματα βιοπληροφορικής προτιμάται μιας και οι επιστήμονες που ασχολούνται με τη γενετική κατανοούν πιο εύκολα τις έννοιες και τη λογική που απαιτείται για να αναπτύξουν γενετικούς αλγόριθμους[22]. Πιο συγκεκριμένα, τα πλεονεκτήματα των γενετικών αλγόριθμων θα μπορούσαν να συνοψιστούν στα εξής:

- Μπορούν να επιλύουν δύσκολα προβλήματα γρήγορα και αξιόπιστα.
- Μπορούν εύκολα να συνεργαστούν με τα υπάρχοντα μοντέλα και συστήματα.
- Είναι εύκολα επεκτάσιμοι και εξελίξιμοι.
- Μπορούν να συμμετέχουν σε υβριδικές μορφές με άλλες μεθόδους.
- Εφαρμόζονται σε πολύ περισσότερα πεδία από κάθε άλλη μέθοδο.
- Δεν απαιτούν περιορισμούς στις συναρτήσεις που επεξεργάζονται.
- Δεν ενδιαφέρει η σημασία της υπό εξέταση πληροφορίας.
- Διαθέτουν από τη φύση τους το στοιχείο του παραλληλισμού.
- Είναι μία μέθοδος που πραγματοποιεί ταυτόχρονα διερεύνηση του χώρου αναζήτησης και εκμετάλλευση της ήδη επεξεργασμένης πληροφορίας[23].

Από την άλλη πλευρά, μια σειρά από μειονεκτήματα χαρακτηρίζουν τους γενετικούς αλγόριθμους:

- Προβλήματα εξοικείωσης με τη Γενετική. Όσο εξοικειωμένοι

είναι οι βιολόγοι και οι γενετιστές με την ορολογία και την λογική των γενετικών αλγόριθμων, άλλο τόσο δύσκολο είναι να την αντιληφθούν οι προγραμματιστές και όσοι προσεγγίζουν το συγκεκριμένο πεδίο χωρίς άμεση επαφή με την βιολογία.

- Το πρόβλημα του χρόνου. Το ερώτημα που τίθεται από πολλούς είναι απλό: μπορεί να προσομοιωθεί η γενετική εξέλιξη που έλαβε χώρα σε τεράστιο χρονικό διάστημα με αυτήν που λαμβάνει χώρα σε έναν υπολογιστή σε πολύ μικρά χρονικά διαστήματα;

4.5 Μέθοδοι Μηχανικής Μάθησης εκτός του χώρου της υπολογιστικής νοημοσύνης

Δεν θα ήταν δυνατό στα πλαίσια μιας διπλωματικής εργασίας να παρουσιάσουμε αναλυτικά το σύνολο των τεχνικών μηχανικής μάθησης που εφαρμόζονται σε προβλήματα βιοπληροφορικής. Πέρα από τις τεχνικές που περιγράψαμε παραπάνω, τεχνικές που αποτελούν άλλωστε και τις δομικές μονάδες για τον χώρο που ορίζεται σαν υπολογιστική νοημοσύνη, μια σειρά από άλλες χρησιμοποιούνται με πολύ καλά σε πολλές περιπτώσεις αποτελέσματα. Τα δέντρα αποφάσεων (DT-GBI, C4.5)[24], ο EM-αλγόριθμος [25], η τεχνική των k κοντινότερων γειτόνων[26] αποτελούν τις κυριότερες τεχνικές που χρησιμοποιούνται πέραν αυτών της υπολογιστικής νοημοσύνης. Η πιο συστηματική κάλυψη των μεθόδων αυτών από εργαλεία όπως η Matlab και το WEKA[27] δίνουν την ευκαιρία σε όσους επιθυμούν, να εξετάσουν την επίδοση των μεθόδων αυτών αλλά και συνδυασμών τους.

---

[1] P. Baldi and S. Brunak – Bioinformatics: The machine learning approach – MIT press, 2002
A.M. Campbell, L. Heyer – Discovering Genomics, Proteomics and Bioinformatics – Benjamin Cummings, 2002
D. Krane and M.Raymer – Fundamental Concepts Of Bioinformatics – Benjamin Cummings, 2003

[2] Σπύρος Γ. Τζαφέστας - Υπολογιστική Νοημοσύνη – Έκδοση ΕΜΠ, τόμος Α, 2002

[3] Κώστας Α. Τσιώλης & Χρήστος Ι. Φερλές - Ασύρματη Κατανεμημένη Υλοποίηση Νευρο-Ασαφούς Συστήματος Ταξινόμησης, 2003

[4] Marvin Minsky – Neural Nets and the Brain model problem – 1954

[5] Warren McCulloch and Walter Pitts - A logical calculus of the ideas immanent in nervous activity - 1943

[6] Frank Rosenblatt - the Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain - 1958

[7] Widrow B. and Hoff M. - Adaptive switching circuits – 1960

[8] Minsky, M. L. - Steps toward artificial intelligence - 1961

[9] Marvin Minsky and Seymour Papert - Perceptrons: An introduction to Computational Geometry - 1969

[10] Von der Malsburg, C. - Self-organization of Orientation Sensitive Cells in the striate Cortex - 1973

[11] Hopfield, J. - Neural networks and physical systems with emergent collective computational properties - 1982

[12] S. Kirkpatrick, C. D. Galatti, and M. P. Vecchi - Optimization by simulated annealing – 1983

[13] Barto A. G., Sutton R. S. & Anderson C. W. - Neuronlike elements that can solve difficult learning control problems - 1983

[14] Rumelhart D.E., Hinton G. E. and Williams R. J. - Learning internal representations by error propagation – 1986

[15] Broomhead D.S. and Lowe D. - Multivariable functional interpolation and adaptive networks - 1988

[16] D Rumelhart and J McClelland - Parallel Distributed Processing: Explorations in the Microstructure of Cognition – 1986

[17] Για περισσότερα στοιχεία όσον αφορά τα ασαφή σύνολα: J.S.R. Lang, C.T.Sun, E.Mituzani – Neuro-Fuzzy and Soft Computing – Prentice Hall, 1997

[18] Για μια σειρά συγκεκριμένων εφαρμογών, περισσότερα στο κεφάλαιο 4

[19] Ευστράτιος Φ. Γεωργόπουλος & Σπυρίδων Δ. Λυκοθανάσης,- Σημειώσεις Πανεπιστημίου Πατρών - Εισαγωγή στους Γενετικούς Αλγορίθμους, 1999

[20] Ιωάννης Βαλαβάνης - Ανάπτυξη υπολογιστικού συστήματος υποστήριξης διάγνωσης εστιακών βλαβών ήπατος με χρήση χαρακτηριστικών υφής και τεχνικών πολλαπλής ταξινόμησης - 2003

[21] Jerome H. Friedman, Jon Louis Bentley & Raphael A. Finkel - An Algorithm for Finding Best Matches in Logarithmic Expected Time – 1977

[22] Μια σειρά από ενδιαφέρουσες εφαρμογές γενετικών αλγόριθμων στη βιοπληροφορική: Gary B. Fogel and David W. Corne – Evolutionary Computation in Bioinformatics – Morgan Kaufmann Publishers, 2003

[23] Ε. Α. Παππάς – Ανάπτυξη συστήματος πρόβλεψης δευτεροταγούς δομής πρωτεϊνών με χρήση τεχνικών κοντινότερων γειτόνων και πιθανοτικών νευρωνικών δικτύων – 2004

[24] Geamsakul, W. et al. – Constructing a Decision Tree for graph structured data
J. R. Quinlan - C4.5: Programs for machine learning – Morgan Kaufmann Publishers, 1993

[25] Lawrence et al. – An expectation-maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences – Proteins: Structure, Function, and Genetics, 7:41-51, 1990

[26] Peter Waiganjo Wagacha - Instance-Based Learning: *k*-Nearest Neighbor - 2003

[27] http://www.cs.waikato.ac.nz/ml/weka

Κεφάλαιο 5

Η θεωρία των υποκινητών και το βακτήριο E.Coli

5 Η θεωρία των υποκινητών και το βακτήριο E. Coli

5.1 Η θεωρία των υποκινητών

Τόσο τα προκαρυωτικά, όσο και τα ευκαρυωτικά κύτταρα κατά το στάδιο της μεταγραφής ακολουθούν τα όσα συμπυκνώνονται στην θεωρία των υποκινητών (promoters theory). Κομβικά η θεωρία αυτή περιγράφει, με σχετικά διαφορετικό τρόπο για τις δύο κατηγορίες κυττάρων, την κατηγοριοποίηση συγκεκριμένων κομματιών της ακολουθίας DNA ενός γονίδιου με βάση τη λειτουργία που έχουν στη διαδικασία της πρωτεϊνοσύνθεσης. Πιο συγκεκριμένα, για τα ευκαρυωτικά κύτταρα το γονίδιο μπορεί χονδρικά να χωριστεί σε πέντε μέρη: τον υποκινητή (promoter), την περιοχή '5 UTR (untranslated region), τα εξόνια (exons), τα ιντρόνια (introns) και την απολη κτική ακολουθία (terminator). Από την άλλη πλευρά, τα προκαρυωτικά κύτταρα χωρίζονται σε τρία μέρη: τον υποκινητή, την κωδικοποιούμενη περιοχή (coding region) και την απολη κτική ακολουθία όπως και στα ευκαρυωτικά κύτταρα[1]. Για την πρώτη περίπτωση, η πλειοψηφία των υποκινητών είναι υποακολουθίες οι οποίες μπορούν να περιγραφούν με βάση τέσσερις παράγοντες:

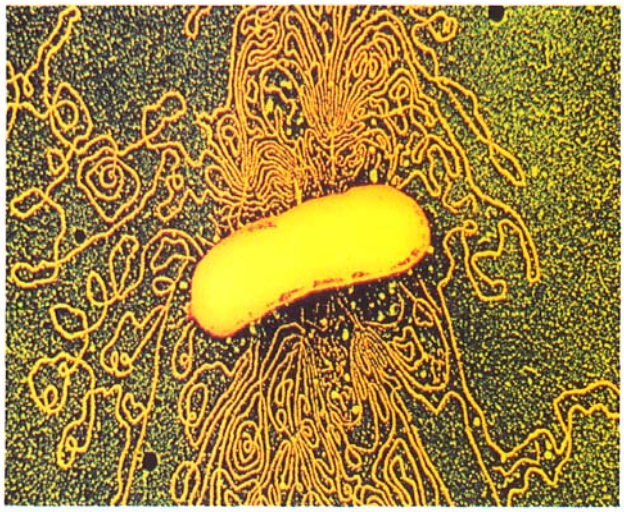
1. το σημείο εκκίνησης της μεταγραφής (transcription start site - TSS). Σε γενικές γραμμές είναι πυριμιδίνη (C ή T) που ακολουθείται από πουρίνη (A ή G). Στις περισσότερες περιπτώσεις η πουρίνη αποτελεί και την αρχή της αρίθμησης (+1).
2. TATAAT: η συγκεκριμένη πλειοψηφική υποακολουθία (consensus box) 6 νουκλεοτιδίων τοποθετείται 10 περίπου bp στην κατεύθυνση 3'→5' (upstream) από το TSS. Ονομάζεται και υποακολουθία -10 ή κουτί Pribnow.
3. TTGACA: και αυτό το consensus box αποτελείται από 6 νουκλεοτίδια και τοποθετείται 35 περίπου bp στην κατεύθυνση 3'→5' (upstream) από το TSS. Ονομάζεται και υποακολουθία -35.
4. Απόσταση (TATAAT- TTGACA). Η απόσταση μεταξύ των υποακολουθιών -35 και -10 είναι καθοριστική στην διατήρηση των δυο περιοχών σε συγκεκριμένη απόσταση για την γεωμετρία της RNA-πολυμεράσης. Η απόσταση κυμαίνεται μεταξύ 15 και 21 bp.

Η διαφοροποίηση μεταξύ των προκαρυωτικών και ευκαρυωτικών κυττάρων έγκειται στην πιο περίπλοκη διαδικασία της μεταγραφής στα ευκαρυωτικά κύτταρα σε σύγκριση με τα προκαρυωτικά.

5.2 Το βακτήριο E. Coli

Η E.Coli αποτελεί συντόμευση του βακτηρίου το οποίο ανήκει στην οικογένεια Enterobacteriaceae και ονομάζεται Escherichia (γένος) coli (είδος) (Σχήμα 5.1). Η E.Coli βρίσκεται στο έντερο των ενήλικων ανθρώπων και σύμφωνα με τελευταίες δημοσιεύσεις αποτελεί το 0.1% του συνολικού αριθμού βακτηρίων που υπάρχουν σε αυτό. Η E.Coli μεταδίδεται στο νεογέννητο κατά τις πρώτες εβδομάδες πιθανότατα από την μητέρα χωρίς μέχρι σήμερα να είναι απολύτως γνωστοί οι μηχανισμοί. Το όνομα Escherichia προέρχεται από τον Escherich, ο οποίος για πρώτη φορά απομόνωσε και χαρακτήρισε το συγκεκριμένο βακτήριο το 1885. Η παρουσία του E.Coli είναι απαραίτητη στον ανθρώπινο οργανισμό, όπως άλλωστε και μιας

σειρά άλλων βακτηρίων, αφού συμμετέχει καταλυτικά στην παραγωγή μιας σειράς απαραίτητων βιταμινών. Ειδικότερα για την E.Coli, συμμετέχει στην παραγωγή βιταμίνης-K και στην ομάδα B-βιταμινών. Για αυτόν ακριβώς το λόγο η παρουσία της E.Coli είναι απαραίτητη στον ανθρώπινο οργανισμό από τις πρώτες μέρες ύπαρξής του.



Εικόνα 5.1: Η E.Coli σε φάση ανάπτυξης του DNA της.

Παρά την αναγκαία ύπαρξη της E.Coli στον ανθρώπινο οργανισμό για να πραγματοποιηθούν εξαιρετικά σημαντικές λειτουργίες του, υπάρχουν περιπτώσεις όπου μεταλλάξεις της E.Coli έχουν αρνητικό ρόλο. Μια αντιστοιχία που θα μπορούσαμε να κάνουμε για να κατανοήσουμε το συγκεκριμένο θέμα είναι αυτή της ποικιλίας των δυνατών συνδυασμών για τον Homo Sapiens. Παρά δηλαδή το γεγονός του ότι όλοι οι ανθρώπινοι οργανισμοί ανήκουν στο ίδιο γένος (Homo) και στο ίδιο είδος (Sapiens) το γενετικό υλικό του κάθε ανθρώπου είναι διαφορετικό από οποιουδήποτε άλλου. Αντίστοιχα, παρά το γεγονός ότι το σύνολο των E.Coli ανήκουν στο ίδιο γένος και το ίδιο είδος υπάρχει η δυνατότητα ενός μεγάλου αριθμού δυνατών συνδυασμών, όπου κάποιοι από αυτούς είναι πιθανόν να είναι βλαβεροί για τον ανθρώπινο οργανισμό. Η εισαγωγή στον ανθρώπινο οργανισμό ενός βακτηρίου E.Coli πέρα από αυτά που ήδη ενυπάρχουν στο έντερό του μπορεί να αποτελέσει και την αιτία μόλυνσης του από το βακτήριο. Εξάλλου η γενικευμένη ύπαρξη της E.Coli στο περιβάλλον (σε άλλους ζώντες οργανισμούς, στο νερό ή το χώμα) ενισχύει κατά πολύ μια τέτοια πιθανότητα. Ένα χαρακτηριστικό παράδειγμα μιας ομάδας E.Coli που μπορεί να αποτελέσει αιτία ήπιας αιματηρής διάρροιας είναι η EHEC (enterohemorrhagic E.Coli group). Η είσοδός της στον ανθρώπινο οργανισμό μπορεί να γίνει με την τροφή ή με την πόση μολυσμένου νερού. Μία από τις πιο επικίνδυνες E.Coli που ανήκουν στην κατηγορία EHEC είναι και η E.Coli O157:H7 (ο συνδυασμός γραμμάτων και αριθμών στο ακριβές όνομα ενός βακτηρίου παραπέμπει σε συγκεκριμένους γενετικούς σημειωτές που έχουν βρεθεί στην επιφάνειά του και που το διαχωρίζουν από άλλους τύπους E.Coli). Η συγκεκριμένη E.Coli έχει μολυνθεί από έναν βακτηριακό ιό, ο οποίος έχει την ικανότητα να εισάγει το δικό του DNA στο χρωμόσωμα της E.Coli παραμένοντας εκεί χωρίς να αλλοιώνει τον ρόλο της. Παράλληλα όμως, καθώς το DNA της E.Coli περιέχει και το γενετικά διαφοροποιημένο DNA του ιού, με την διαίρεση του κυττάρου και την αριθμητική του αύξηση, μια σειρά νέων κυττάρων που δημιουργούνται εμπεριέχουν το διαφοροποιημένο DNA. Η νέα γενετική πληροφορία που περιέχεται στην νέα ακολουθία του DNA της E.Coli προβλέπει και την παραγωγή μιας τοξίνης με την ονομασία Vero toxin. Η τοξίνη αυτή (μια μορφή πρωτεΐνης) προκαλεί σοβαρές βλάβες στο εσωτερικό του εντέρου, ειδικότερα στα επιθυλιακά κύτταρά του (κύτταρα τα οποία οριοθετούν το τοίχωμα του εντέρου). Οι συνέπειες μπορούν να είναι εξαιρετικά σοβαρές, ιδιαίτερα σε παιδιά μικρής ηλικίας που δεν έχουν αναπτύξει ακόμα δυνατότητες αντιμετώπισης μεγάλης τάξης

αιμορραγίας. Οι ουρολοιμώξεις σε εξωνοσοκομειακούς ασθενείς σε αναλογία 80% προκαλούνται από E.Coli. Χαρακτηριστικό είναι το γεγονός ότι κατά μέσο όρο κάθε χρόνο στις ΗΠΑ παρατηρούνται 73000 περιπτώσεις μόλυνσης με 61 θανάτους. Η λοίμωξη μπορεί να μεταδοθεί εύκολα μέσα από άβραστη τροφή και μη παστεριωμένο γάλα. Επίσης μπορεί να μεταδοθεί μέσα σε κλειστό περιβάλλον στο οποίο βρίσκεται ήδη μολυσμένο άτομο[2].

Πέρα όμως από τις λοιμώξεις που είναι πιθανό να προκαλέσει η E.Coli σε ανοιχτό περιβάλλον, έχει διαπιστωθεί ότι είναι η αιτία για μια σειρά κλινικών νοσημάτων εξαιτίας της αξιοσημείωτης αντοχής της σε αντιβιοτικά. Στις εκδηλώσεις συγχέεται με άλλα είδη εντεροβακτηρίων όπως Salmonella spp, Shighella spp κα. Η πιο κοινή εκδήλωση που προκαλεί είναι σε εντερικές και ουρογεννητικές οδούς (προκαλώντας διάρροια, δυσεντερία κα.) Μπορεί όμως και να εισέλθει βαθύτερα στους ιστούς προκαλώντας σηψαιμία στο αίμα ανοσοκατεσταλμένων και μηνιγγίτιδες στα παιδιά.

Πίνακας 5.1: Οι νοσοκομειακές λοιμώξεις και η παρουσία της E.Coli σε αυτές.

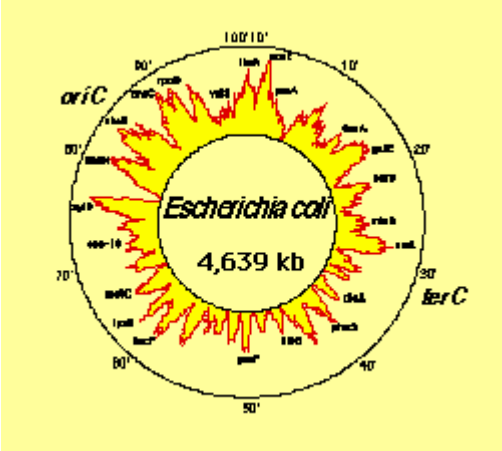
Λοιμώξεις	Βακτήρια
Ουρολοιμώξεις	*Escherichia coli* *Klebsiella spp* *Proteus spp* *Pseudomonas aeruginosa*
Χειρουργικές λοιμώξεις	*Staphylococcus aureus* *Pseudomonas spp* *Escherichia coli* *Klebsiella spp*
Λοιμώξεις του αναπνευστικού συστήματος	*Pseudomonas spp* *Klebsiella spp* *Staphylococcus aureus* *Escherichia coli*
Βακτηριαιμίες	*Escherichia coli* *Pseudomonas spp* *Klebsiella spp* *Enterobacter spp*

Η δομή και το χρωμόσωμα της E.Coli ακολουθεί τη γενικότερη μορφή που έχουν τα βακτήρια. Τα βακτηριακά χρωμοσώματα αποτελούνται από αδιάσπαστα κυκλικά μόρια με διπλή έλικα DNA. Το μέγεθος του χρωμοσώματος της E.Coli είναι $4.7*10^6$ bp. Το μέσο μέγεθος για ένα βακτηριακό γονίδιο είναι 1000 bp και ο αριθμός τους για την E.Coli είναι περίπου 3500 (ο αντίστοιχος αριθμός για τον άνθρωπο είναι της τάξης των 30000 γονιδίων)[3]. Η δομή του DNA στην E.Coli είναι με την γνωστή μορφή της διπλής έλικας και έχει δυο σημαντικές λειτουργίες:

1. Το γενετικό υλικό υποβάλλεται σε αντιγραφή πριν την κυτταρική διαίρεση.
2. Το γενετικό υλικό καθοδηγεί την πρωτεϊνοσύνθεση.

Η αντιγραφή του DNA ξεκινάει από ένα μοναδικό σημείο της ακολουθίας του DNA (origin of replication) και λήγει στο μέσον περίπου της διαδρομής μέχρι το τέλος του χρωμοσώματος (terminus of replication). Οι διχάλες αντιγραφής (replication forks) αποτελούν τα σημεία όπου η αντιγραμμένη ακολουθία DNA διαχωρίζεται σε δυο διπλές αλυσίδες, καθώς η αντιγραφή προχωράει από το σημείο έναρξης της αντιγραφής. Η αντιγραφή του DNA της E.Coli είναι διπλής κατεύθυνσης, δηλαδή θα έχουμε δυο διχάλες αντιγραφής που θα κινούνται με αντίθετη κατεύθυνση γύρω από το χρωμόσωμα. Η γενετική πληροφορία που μεταφέρεται μέσα από τις νέες διπλές έλικες του DNA αποκωδικοποιείται από τον

μηχανισμό του κυττάρου και αποτελεί το νέο πρόγραμμα δράσης του για την δημιουργία πρωτεϊνών.



Εικόνα 5.2: Με την επιλογή αυτής της εικόνας στον δικτυακό τόπο **http://genolist.pasteur.fr/Colibri/** δίνεται η δυνατότητα μελέτης όλου του γονιδιώματος της E.Coli.

5.3 Η θεωρία υποκινητών στην E.Coli

Ο ρόλος των υποκινητών στην E.Coli και τα χαρακτηριστικά τους έρχονται σε αντιστοιχία με την γενικότερη λειτουργία των υποκινητών στα βακτηριακά κύτταρα. Πιο συγκεκριμένα, οι υποκινητές της E.Coli χαρακτηρίζονται από σχετικά μεγάλη ποικιλομορφία, η οποία εξάλλου αποδεικνύεται από τον αριθμό των υποκινητών που έχουν βρεθεί στο βακτήριο μέχρι σήμερα. Κατά την ολοένα και πιο αναλυτική έρευνα της E.Coli μια σειρά από ερευνητές δημιούργησαν συλλογές υποκινητών, οι οποίες και δημοσιεύθηκαν σε ένα μεταγενέστερο στάδιο[4]. Από αυτές τις συλλογές, μετά από στατιστική ανάλυσή τους, δημιουργήθηκαν και οι «πλειοψηφικές» ακολουθίες (consensus boxes). Η πρώτη συλλογή που ανέδειξε σε έναν βαθμό το consensus box των υποκινητών της E.Coli είναι αυτή των Calvin Harley και Robert Reynolds το 1987 όπου ανέλυσαν 263 ακολουθίες υποκινητών της E.Coli. Το consensus box που εμφανίστηκε ήταν το εξής:

$$T_{78}T_{82}G_{68}A_{58}C_{52}A_{54} \text{ -- } 16_{21}17_{52}18_{19} \text{ -- } T_{82}A_{89}T_{52}A_{59}A_{49}T_{89}$$

Μια δεύτερη δημοσίευση συλλογής E.Coli υποκινητών έγινε το 1993 από τους Shlommit Lisser και Hanah Margalit [4]. Τα αποτελέσματα δεν ήταν ακριβώς τα ίδια με αυτά της προηγούμενης συλλογής αλλά δεν εμφανίστηκαν μεγάλες διαφορές. Το consensus box σε αυτήν την περίπτωση ήταν:

$$T_{69}T_{79}G_{61}A_{56}C_{54}A_{54} \text{ -- } 16_{17}17_{43}18_{17} \text{ -- } T_{77}A_{76}T_{60}A_{61}A_{56}T_{82}$$

Παρατηρούμε ότι δεν παρουσιάζονται σημαντικές διαφορές μεταξύ των consensus box στις δυο περιπτώσεις. Παρ' όλα αυτά παρατηρούμε ότι η συχνότητα εμφάνισης νουκλεοτιδίων που είχαν υψηλό ποσοστό στην πρώτη περίπτωση εμφανίζουν πτωτική τάση συχνότητας εμφάνισης στην δεύτερη, ενώ αντίθετα νουκλεοτίδια που είχαν στην πρώτη περίπτωση χαμηλή συχνότητα εμφάνισης εμφανίζουν αυξητική τάση εμφάνισης. Η βιολογική εξήγηση αυτού του φαινομένου είναι η ύπαρξη διαφορετικών υποκινητών για διαφορετικές μορφές RNA-

πολυμεράσης και διαφορετικά μεγέθη της. Αυτόν το ρόλο έχουν οι λεγόμενες υπομονάδες σ (σ-factors) οι οποίες αναγνωρίζουν διαφορετικά είδη υποκινητών[5]. Ειδικά για την E.Coli, θα μπορούσαμε να πούμε ότι υπάρχουν ισχυροί και ασθενείς υποκινητές (strong and weak promoters). Χαρακτηριστικό παράδειγμα ισχυρού υποκινητή είναι ο υποκινητής με κωδική ονομασία recA. Οι περιοχές -35, -10 καθώς και η απόσταση μεταξύ τους είναι η παρακάτω:

TTGATA -- 16 – TATAAT

Η διαφορά μεταξύ της συγκεκριμένης ακολουθίας και του consensus box είναι η εξής:

TTGATA -- 16 -- TATAAT
TTGACA -- 17 – TATAAT

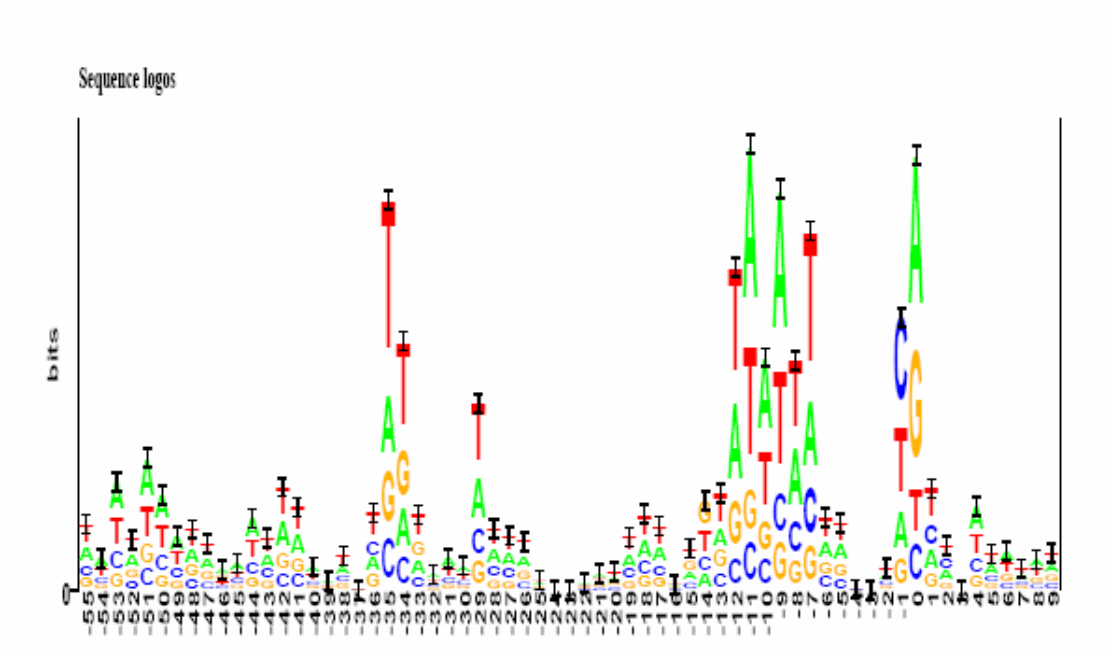
Από την άλλη πλευρά ένα παράδειγμα ασθενούς υποκινητή είναι ο araBAD ο οποίος σε αντιστοιχία με την εμφάνιση του recA είναι ο εξής:

CTGACG -- 18 – TACTGT

Η διαφορά μεταξύ της συγκεκριμένης ακολουθίας και του consensus box είναι η εξής:

CTGACG -- 18 -- TACTGT
TTGACA -- 17 – TATAAT

Συμπεραίνουμε, λοιπόν, ότι οι ισχυροί υποκινητές έχουν μεγαλύτερο βαθμό ομοιότητας με το consensus box σε σχέση με τους ασθενείς. Το βακτήριο E.Coli διαθέτει πολλούς σ παράγοντες που αναγνωρίζουν διαφορετικά σετ γονιδίων. Χαρακτηριστικές υπομονάδες είναι για την E.Coli οι σ^{70},σ^{32} και σ^{54}, με διαφορετικές λειτουργίες η καθεμία. Πέρα από τα χαρακτηριστικά που εμφανίζονται στις συγκεκριμένες περιοχές κάποια άλλα ασθενή μοτίβα εμφανίζονται στην βιβλιογραφία που αναφέρεται στην μελέτη της E.Coli[6]. Τα ασθενή αυτά μοτίβα εμφανίζονται άλλωστε και στα λεγόμενα sequence logos. Τα sequence logos είναι γραφικές αναπαραστάσεις ευθυγραμμισμένων συνόλων ακολουθιών. Στα sequence logos εμφανίζεται για κάθε θέση μια στοίβα (stack) όπου το συνολικό ύψος αναπαριστά την συσσώρευση της πληροφορίας (information content) που μας δίνεται για κάθε θέση, ενώ παράλληλα το ύψος κάθε γράμματος που αντιπροσωπεύει ένα συγκεκριμένο νουκλεοτίδιο δείχνει την συχνότητα εμφάνισης αυτού σε κάθε συγκεκριμένη θέση[7]. Η ουσιαστική διαφορά μεταξύ των consensus boxes και των sequence logos είναι ότι τα δεύτερα δείχνουν τον συνολικό βαθμό ουσιαστικής πληροφορίας που έχουμε σε κάθε θέση. Παρατηρούμε για παράδειγμα στο sequence logo της E.Coli ότι το ύψος της στοίβας στις θέσεις -35, -10 και στο TSS είναι πολύ υψηλότερο από ότι στις υπόλοιπες θέσεις. Ο κάθετος άξονας δείχνει την ποσότητα της πληροφορίας που έχουμε σε bits, ενώ ο οριζόντιος τα bp που οριοθετούν σε ένα βαθμό την περιοχή του υποκινητή. Στο Σχήμα 5.1 εμφανίζεται το sequence logo μιας συλλογής 438 υποκινητών της E.Coli, οι οποίοι έχουν ευθυγραμμιστεί σύμφωνα με το σημείο εκκίνησης της μεταγραφής.



Σχήμα 5.1: Το sequence logo μιας συλλογής 438 υποκινητών E.Coli. Να σημειωθεί ότι η θέση 0 στο γράφημα αντιστοιχεί στη θέση +1 των δημοσιεύσεων που έχουμε μελετήσει.

Μια δεύτερη παρατήρηση που μπορούμε να κάνουμε είναι ότι στην περίπτωση των υποκινητών της E.Coli η ορθότητα της πληροφορίας που μας δίνεται από το consensus box επιβεβαιώνεται και από το sequence logo που αναπαριστά την δομή της ακολουθίας των 438 υποκινητών E.Coli.

Συνοψίζοντας το πρόβλημα των υποκινητών στην E.Coli τα στοιχεία που περιγράφουν το πρόβλημα είναι τα εξής:

- Απαιτείται ευθυγράμμιση (alignment) των δοσμένων ακολουθιών βάσει του σημείου εκκίνησης της μεταγραφής (transcription start site – TSS).
- Η περιοχή -35 (35 bp στην κατεύθυνση 3'→ 5' – upstream) δίνει το πρώτο εξαμερές consensus box που είναι το: TTGACA.
- Η περιοχή -10 (10 bp στην κατεύθυνση 3'→ 5' – upstream) δίνει το δεύτερο εξαμερές consensus box : TATAAT.
- Η απόσταση μεταξύ των περιοχών -35 και -10 κυμαίνεται μεταξύ 15 και 21 bp και στις περισσότερες περιπτώσεις 16, 17 ή 18 bp.
- Οι υπομονάδες σ αναγνωρίζουν τους υποκινητές και το μέγεθός τους, ενώ παράλληλα δίνουν μια σαφή εικόνα του ποιοι είναι «ισχυροί» και ποιοι «ασθενείς» υποκινητές.
- Πρέπει να λαμβάνεται υπόψιν η απουσία κάποιων υποκινητών, αφού μέχρι σήμερα δεν έχει ακόμα βρεθεί το σύνολο τους στην E.Coli.

5.4 Μέθοδοι επίλυσης του προβλήματος υποκινητών στην E.Coli

Η περίπτωση του βακτηρίου E.Coli και η μελέτη σε αυτό της θεωρίας υποκινητών έχει γίνει με μια σειρά μεθόδων που εμφανίζεται σε πλειάδα άρθρων. Οι κυριότερες εφαρμογές που έχουν γίνει, έχουν βασιστεί σε μεγάλο βαθμό σε διαφορετικά μοντέλα τεχνητής νοημοσύνης. Όπως σε γενικότερο επίπεδο η βιοπληροφορική χρησιμοποιεί διάφορες μορφές νευρωνικών δικτύων[8], εξελικτικούς αλγόριθμους[9], δέντρα αποφάσεων[10], στατιστικές μεθόδους όπως τα hidden Markov Models[11], γενετικό προγραμματισμό[12] και ασαφή λογική[13] και πρόσφατα συνδυασμούς κάποιων από αυτές τις μεθόδους, έτσι και για το πρόβλημα πρόγνωσης υποκινητών οι ερευνητές χρησιμοποιούν μια σειρά τέτοιων μεθόδων για να το προσεγγίσουν με όσο το δυνατόν καλύτερο τρόπο. Το συγκεκριμένο πρόβλημα είναι κομβικής σημασίας για την γονιδιωματική έρευνα. Οι τεράστιες ποσότητες νέων ακολουθιών DNA που ανακαλύπτονται σε όλο τον κόσμο απαιτούν αποδοτικές από άποψης ταχύτητας και ακριβείς μεθόδους, για να είναι δυνατή μια όσο το δυνατόν περισσότερο αυτοματοποιημένη ανάλυση. Ένα από τα σημαντικότερα προβλήματα είναι αν μια ακολουθία περιέχει ένα ή περισσότερα γονίδια, και, αν ναι, ποια είναι η ακριβής περιοχή όπου αυτά υπάρχουν. Στο προηγούμενο κεφάλαιο κάναμε μια όσο το δυνατόν πιο περιεκτική και ακριβή περιγραφή των κυριότερων μεθόδων. Παραθέτουμε στη συνέχεια τις εφαρμογές που έχουν γίνει στην E.Coli πάνω στο πρόβλημα που στη συνέχεια προσπαθήσαμε και εμείς να αντιμετωπίσουμε.

5.5 Εφαρμογές μεθόδων μηχανικής μάθησης για την πρόγνωση υποκινητών στην E.Coli

Οι P.Horton και M.Kanehisa[14] το 1992 υλοποίησαν μια από τις απλούστερες εφαρμογές νευρωνικού δικτύου για τους υποκινητές της E.Coli. Η δημιουργία ενός απλού perceptron νευρωνικού δικτύου, το οποίο βελτιώνει την ικανότητα γενίκευσης με μια νέα τεχνική επιλογής των χαρακτηριστικών των ακολουθιών που εξετάζονται κατά τη διάρκεια της εκπαίδευσης του δικτύου, δίνει αρκετά καλά ποσοστά επιτυχίας της τάξης του 80.6%.

Ένας από τους βασικότερους αλγόριθμους που χρησιμοποιούνται στο πρόβλημα πρόγνωσης υποκινητών με νευρωνικά δίκτυα είναι ο αλγόριθμος ανάστροφης διάδοσης (backpropagation neural network). Οι I.Mahadevan και I.Ghosh παρουσίασαν μια από τις πιο σημαντικές εφαρμογές πάνω στο πρόβλημα πρόγνωσης υποκινητών στην E.Coli [15]. Στο συγκεκριμένο άρθρο παρουσιάζεται ένας συνδυασμός τριών «υπο-νευρωνικών δικτύων» που να αρθρώνουν ένα συνολικό νευρωνικό δίκτυο, το οποίο τελικά θα αναγνωρίζει ακολουθίες υποκινητών. Σε ένα επόμενο στάδιο εισάγεται στο νευρωνικό δίκτυο για να εκπαιδευτεί και ένα μικρό σύνολο δεδομένων μεταλλαγμένων υποκινητών και εξετάζεται η δυνατότητα του νευρωνικού δικτύου στο να αναγνωρίζει και μεταλλαγμένους υποκινητές. Το νευρωνικό δίκτυο ανάστροφης διάδοσης αναπτύχθηκε στην γλώσσα προγραμματισμού C σε περιβάλλον εργασίας UNIX. Κάθε γράμμα που αντιπροσωπεύει ένα συγκεκριμένο νουκλεοτίδιο αναπαραστάθηκε με 4 bits (A = 0001, T = 0010, C = 1000, G = 0100). Η έξοδος του νευρωνικού είναι 1.0 αν μια ακολουθία είναι υποκινητής και 0.0 αν μια ακολουθία δεν είναι υποκινητής. Αναλυτικότερα, στο πρώτο κομμάτι του συνολικού νευρωνικού δικτύου αναγνωρίζονται τα consensus boxes των ακολουθιών της E.Coli που απαρτίζουν το σύνολο δεδομένων που εισάγεται στο νευρωνικό δίκτυο. Το πρώτο κομμάτι

αποτελείται από δυο νευρωνικά δίκτυα που αναγνωρίζουν τις περιοχές -35 και -10 αντίστοιχα. Το επίπεδο εισόδου (input layer) κάθε επιμέρους νευρωνικού δικτύου του πρώτου κομματιού έχει 24 νευρώνες (6*4 = 24), 2 νευρώνες στο κρυμμένο επίπεδο (hidden layer), και 1 νευρώνα στο επίπεδο εξόδου (output layer), ο οποίος έχει έξοδο 1.0 για πιθανά boxes υποκινητών και 0.0 για πιθανά boxes μη-υποκινητών. Στο δεύτερο κομμάτι του συνολικού νευρωνικού δικτύου ο στόχος είναι με βάση τα αποτελέσματα του πρώτου κομματιού να γίνει ευθυγράμμιση (alignment) των πιθανών υποκινητών. Τα αποτελέσματα του πρώτου μέρους της εφαρμογής θεωρούνται πιθανοί υποκινητές όταν η τιμή εξόδου είναι μεγαλύτερη του 0.8 (cut-off value = 0.8) και παράλληλα η απόσταση μεταξύ της περιοχής -35 και -10 είναι μεταξύ 15 και 21 bp (15<distance(-35,-10)<21). Στο σχήμα 5.2 φαίνεται ο τρόπος με τον οποίο γίνεται η ευθυγράμμιση των ακολουθιών. Τα κενά είναι τα bp όπου το νευρωνικό δίκτυο δεν έχει εκπαιδευτεί και συμβολίζονται με παύλα (-). Η τελεία (.) δείχνει τα 6 νουκλεοτίδια που υπάρχουν μετά το TSS (Transcription Start Site). Με αυτόν τον τρόπο όλες οι ακολουθίες γίνονται ισομήκεις, με μήκος 65 bp. Το τρίτο κομμάτι του συνολικού νευρωνικού δικτύου εκπαιδεύεται στην αναγνώριση των υποκινητών με μήκος 65 bp. Το επίπεδο εισόδου έχει 260 νευρώνες (65*4 =260) και το επίπεδο εξόδου 1 νευρώνα. Στο κρυφό επίπεδο οι νευρώνες κυμαίνονταν μεταξύ 2 και 12, ενώ στις περισσότερες περιπτώσεις χρησιμοποιήθηκαν 7 νευρώνες. Τα ποσοστά επιτυχίας κυμαίνονται σε υψηλά επίπεδα, 98% για τους υποκινητές και 90.2% για μη υποκινητές.

Μια άλλη προσέγγιση που έχει χρησιμοποιηθεί σε πολλές περιπτώσεις είναι η μέθοδος υβριδικής μάθησης, με χαρακτηριστικότερο παράδειγμα το KBANN (Knowledge-Based Artificial Neural Networks)[16]. Στην συγκεκριμένη εφαρμογή, χρησιμοποιείται ένας συνδυασμός δύο αλγόριθμων. Σε πρώτη φάση, ο αλγόριθμος με την ονομασία "Rules-To-Network" εισάγει σχεδόν σωστούς (approximately correct)

```
                -35        Spacer          -10
CTGTTTCCGAC CTGACA CCTGCGTGAGTTGTTCACG TATTTT TTCACTATgTCTTAC

↓     14     ↓ -35  ▌ 5 ▌▲Rest of Spacer▲ -10  |3|         • 6  •
---CTGTTTCCGACCTGACACCTGC--GTGAGTTGTTCACGTATTTTTTC---ACTATgTCTTAC
```

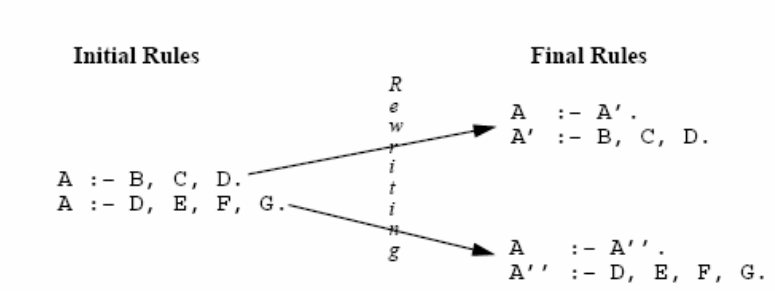
Σχήμα 5.2: Αναπαράσταση μιας ακολουθίας έτσι όπως συμβολίζεται στο άρθρο [14]

συμβολικούς κανόνες στο νευρωνικό δίκτυο. Τα νευρωνικά δίκτυα που δημιουργούνται στο στάδιο αυτό κάνουν τις ίδιες κατηγοριοποιήσεις με αυτές που κάνουν οι κανόνες πάνω στους οποίους βασίζονται. Ο αλγόριθμος "Rules-To-Network" ολοκληρώνεται σε 7 βήματα:

1. Η αναδιαμόρφωση των κανόνων (rewriting). Αυτό γίνεται έτσι ώστε, αν υπάρχουν περισσότεροι από ένας προηγούμενοι κανόνες για έναν παραγόμενο κανόνα, τότε κάθε κανόνας για τον συγκεκριμένο παραγόμενο κανόνα αναδιαμορφώνεται με την μορφή δυο κανόνων. Η αναγκαιότητα για αυτό το βήμα του αλγορίθμου αναδεικνύεται παρακάτω (σχήμα 5.3).
2. Η αντιστοίχιση του συνόλου κανόνων που έχουμε στο νευρωνικό δίκτυο (mapping). Με αυτόν τον τρόπο έχουμε μια γραμμική σχέση λειτουργίας του νευρωνικού δικτύου σε σχέση με το σύνολο κανόνων που διαθέτουμε, με το οποίο το τροφοδοτούμε.
3. Αρίθμηση (numbering). Το βήμα αυτό είναι απαραίτητη προϋπόθεση για την ορθή λειτουργία των επόμενων βημάτων του αλγόριθμου. Πιο συγκεκριμένα, σε αυτό το βήμα το KBANN, αριθμεί τα KBANN-δίκτυα με βάση το «επίπεδό» τους ή, απλούστερα, απαριθμεί τις

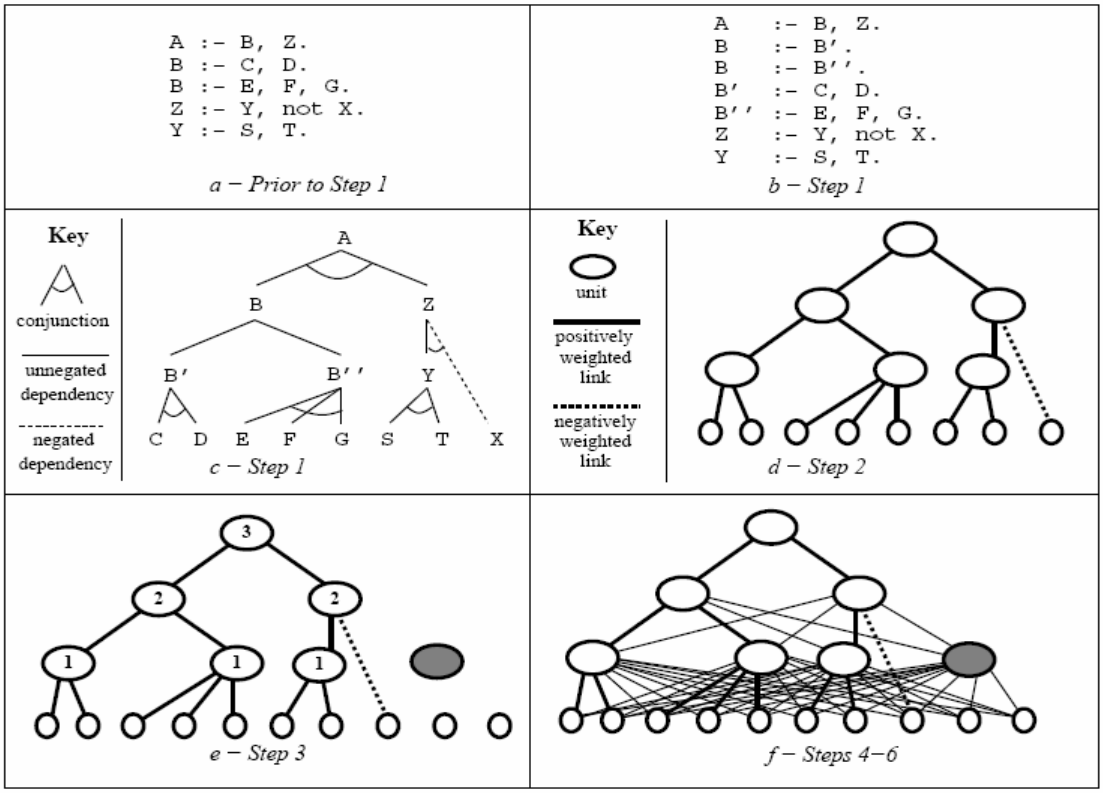
δυνατές διαδρομές των κανόνων από την είσοδο μέχρι την έξοδο, ακόμα και αν οι διαδρομές είναι ανολοκλήρωτες.

4. Πρόσθεση κρυμμένων μονάδων (adding hidden units). Με αυτό το βήμα γίνεται εφικτή η προσθήκη συγκεκριμένων κανόνων που δεν έχουν εισαχθεί σαν αρχικοί κανόνες στο νευρωνικό δίκτυο. Το βήμα αυτό είναι προφανώς προαιρετικό.
5. Εισαγωγή μονάδων εισόδου (adding input units). Σε αυτό το βήμα εισάγεται πληροφορία που είναι σχετική με το πρόβλημα, καθώς το σύνολο κανόνων με το οποίο έχουμε τροφοδοτήσει το νευρωνικό δίκτυο δεν είναι ικανό για να αναγνωρίσει κάθε πρότυπο εισόδου.
6. Εισαγωγή συνδέσμων (adding links). Ο αλγόριθμος συνεχίζεται στο σημείο αυτό προσθέτοντας συνδέσμους με μηδενικά βάρη μεταξύ κάθε n-1 μονάδας με την n μονάδα.
7. Διατάραξη βαρών (perturbing). Με την εισαγωγή ενός μικρού τυχαίου αριθμού σε κάθε βάρος αποφεύγονται προβλήματα που οφείλονται στην συμμετρία.



Σχήμα 3.3: Αναδιαμόρφωση των κανόνων για το KBANN

Βλέπουμε στο Σχήμα 5.4 τα βήματα μέχρι την ολοκλήρωσης του πρώτου από τους δύο αλγόριθμους του KBANN.



Σχήμα 5.4: Η διαδικασία για την ολοκλήρωση του αλγόριθμου “Rules-To-Network”.

Ο δεύτερος αλγόριθμος του KBANN ονομάζεται “Neural Learning” και βελτιώνει τα νευρωνικά δίκτυα που έχουν δημιουργηθεί βάσει του αλγόριθμου ανάστροφης διάδοσης (backpropagation algorithm). Παρά το ότι ο αλγόριθμος ανάστροφης διάδοσης δεν παρέχει μεγάλες βελτιώσεις σε περιπτώσεις όπου η έξοδος είναι αρκετά «ισχυρή» (τιμές κοντά στο 0 ή στο 1), η χρήση διαφορετικής συνάρτησης λάθους (cross-entropy error function), σε σχέση με αυτή που χρησιμοποιείται σε άλλες περιπτώσεις, και η παράλληλη χρήση ενός «ομαλοποιητή» (regularizer) ο οποίος κάνει αλλαγές στην αρχική θεωρία που κωδικοποιείται στη συνέχεια στους κανόνες, δίνουν μεγαλύτερη δυνατότητα γενίκευσης (generalization) και ερμηνείας (interpretability) στο νευρωνικό δίκτυο. Χρησιμοποιώντας κανόνες οι οποίοι περιγράφονται σε Prolog-μορφή στον Πίνακα 5.2, και με σύνολο δεδομένων 53 υποκινητές και 53 μη-υποκινητές με μήκος 57 bp το καθένα, το KBANN έδινε τέσσερις λάθος προβλέψεις στα 106 πρότυπα εισόδου. Συγκριτικά, δηλαδή, με ένα απλό νευρωνικό δίκτυο ανάστροφης διάδοσης έχουμε καλύτερο αποτέλεσμα, πάντα για το ίδιο σύνολο δεδομένων, καθώς ένα απλό backpropagation νευρωνικό δίκτυο δίνει οκτώ λάθος προβλέψεις στα 106 πρότυπα εισόδου.

```
promoter  :- contact, conformation.
contact   :- minus-35, minus-10.

minus-35 :- @-37 'CTTGAC'.            minus-10 :- @-13  'TA*A*T'.
minus-35 :- @-36  'TTGACA'.           minus-10 :- @-12   'TA***T'.
minus-35 :- @-36  'TTG*CA'.           minus-10 :- @-14 'TATAAT'.
minus-35 :- @-36  'TTGAC'.            minus-10 :- @-13  'TATAAT'.

conformation  :-  @-45     'AA**A'.
conformation  :-  @-45     'A***A', @-28 'T***T*AA**T', @-04 'T'.
conformation  :-  @-49 'A****T',    @-27  'T****A**T*TG', @-01 'A'.
conformation  :-  @-47   'CAA*TT*AC', @-22 'G***T*C', @-08 'GCGCC*CC'
```

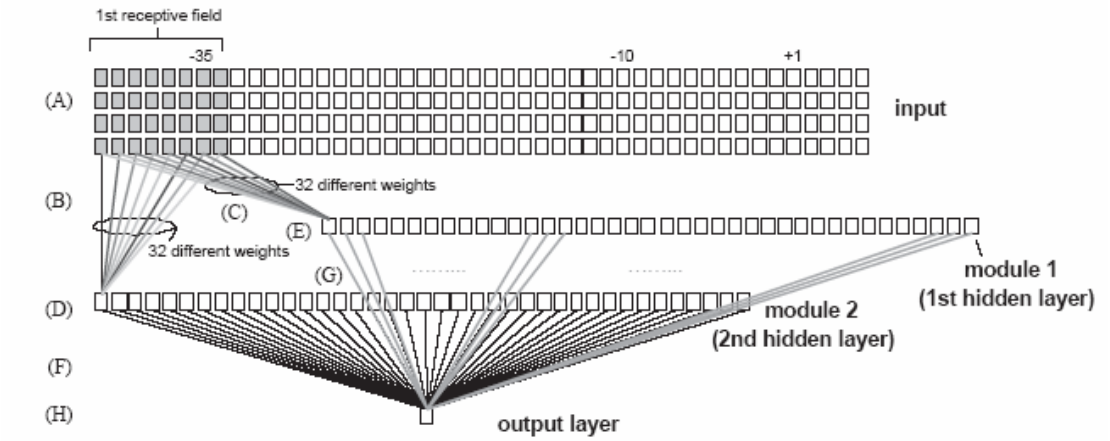
Πίνακας 5.2: Οι κανόνες που χρησιμοποιήθηκαν στο KBANN σε Prolog-μορφή. Το ‘*’ συμβολίζει αδιάφορο χαρακτήρα για τη συγκεκριμένη θέση.

Μια τρίτη προσέγγιση του προβλήματος που συνδυάζει τα νευρωνικά δίκτυα με την ασαφή λογική (fuzzy logic) και τους εξελικτικούς αλγόριθμους είναι το HPAM (Hybrid Promoter Analysis Methodology)[17]. Το HPAM αποτελείται από την σειριακή εφαρμογή τριών διαφορετικών μεθόδων:

1. Αρχικά έχουμε ένα νευρωνικό δίκτυο με χρονική καθυστέρηση (time delayed neural network – TDNN). Το TDNN είναι νευρωνικό δίκτυο προσοτροφοδότησης (feed-forward) και παίρνει σαν εισόδους τις ακολουθίες DNA (Σχήμα 5.5). Χρησιμοποιείται ορθογωνική μέθοδος αναπαράστασης εισόδου (orthogonal input representation), αντιστοιχούμε δηλαδή κάθε νουκλεοτίδιο με τέσσερα bits (A = 1000, C = 0100, G = 0010, T = 0001) έτσι ώστε να διατηρείται η ίδια απόσταση Hamming μεταξύ κάθε ζεύγους ανυσμάτων. Υπολογίζεται ότι τα συνολικά bp που απαιτούνται για να εξεταστεί με ορθό τρόπο το πρόβλημα είναι:
   6 (για την περιοχή -35)
   +6 (για την περιοχή -10)
   +21 (για την max[απόσταση(-35,-10)]
   +13 (υπόλοιπα bp γύρω από το TSS και την περιοχή -35) = 46 bp.

Επομένως απαιτούνται για το επίπεδο εισόδου του TDNN 46*4 = 184 μονάδες εισόδου. Για κάθε μονάδα (module) των υπαρχόντων υπομοτίβων (submotifs) προσθέτουμε και ένα κρυμμένο επίπεδο. Για παράδειγμα, για τα υπομοτίβα των περιοχών -35 και -10 προσθέτονται 2 κρυμμένα επίπεδα όπου το καθένα έχει 46 – 8 + 1 = 39 μονάδες, καθεμία από τις οποίες συνδέεται με 8 bp του επίπεδου εισόδου και κατευθύνεται στο κρυμμένο επίπεδο μετά από τη διαμόρφωση 8*4 = 32 διαφορετικών βαρών. Με αυτόν τον τρόπο έχουμε την εισαγωγή του παράγοντα της απόστασης μεταξύ των υπομοτίβων. Το επίπεδο εξόδου έχει ένα νευρώνα, ο οποίος συνδέεται με όλα τα κρυμμένα επίπεδα και έχει πραγματική τιμή. Παράλληλα χρησιμοποιείται ένας συντελεστής συσχέτισης (correlation coefficient – CC) ο οποίος καθορίζει την κατάλληλη τιμή που πρέπει να επιλεχθεί σαν οριακή τιμή για το διαχωρισμό μεταξύ υποκινητών και μη-υποκινητών. Ο αλγόριθμος του TDNN είναι μια διαφοροποίηση του αλγόριθμου ανάστροφης διάδοσης.

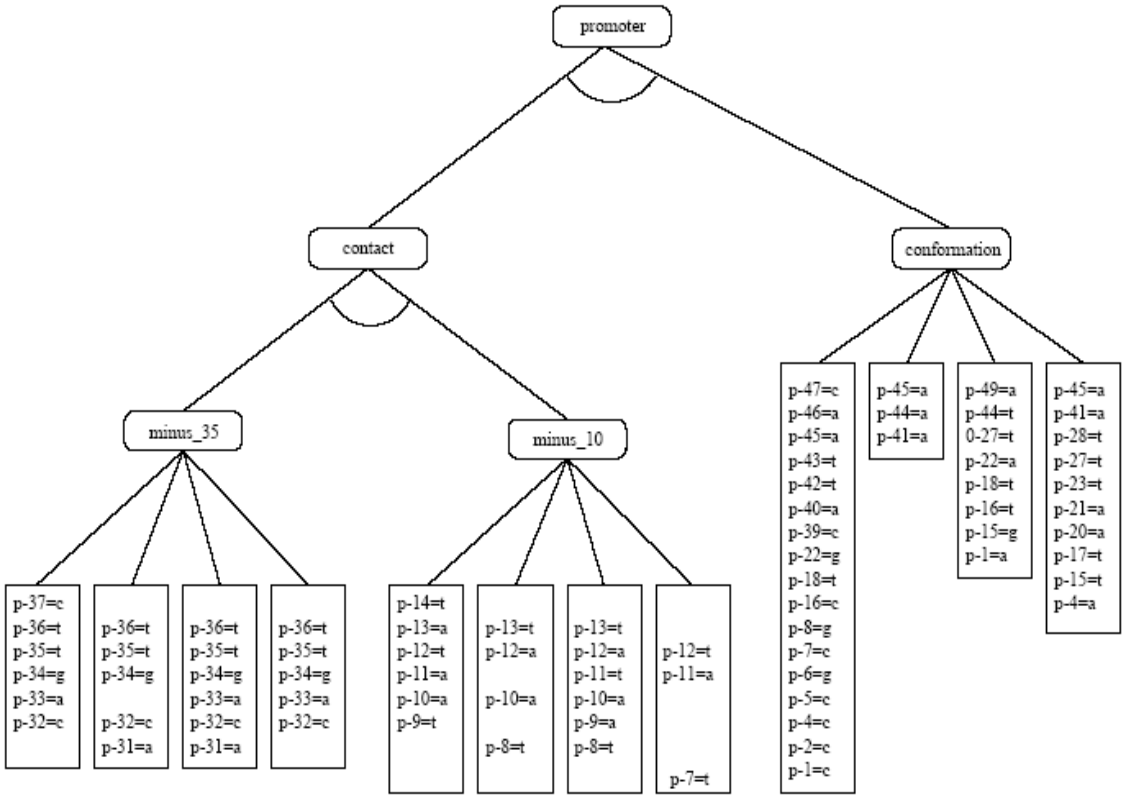
2. Στη συνέχεια γίνεται ανάλυση της εξόδου του TDNN νευρωνικού δικτύου έτσι ώστε να εξαχθεί γνώση για να δημιουργηθεί ένα αποσαφηνισμένο μοντέλο υποκινητών βάσει ασαφών συνόλων. Έτσι εξάγονται συναρτήσεις συμμετοχής για τις περιοχές -35, -10 και την απόσταση μεταξύ τους.
3. Το τελευταίο βήμα είναι μια μέθοδος αναγνώρισης προτύπων (pattern recognition method). Ονομάζεται MOSS (Multi-Objective Scatter Search) και βασίζεται στον εξελικτικό αλγόριθμο scatter search με την διαφορά ότι έχει πολλαπλούς στόχους (αντιστοίχιση μιας DNA ακολουθίας στα υπομοτίβα και την απόσταση μεταξύ τους).



Σχήμα 5.5: Η αρχιτεκτονική του TDNN

Τα αποτελέσματα του HPAM είναι ιδιαίτερα υψηλά για όλα τα μέρη από τα οποία αποτελείται (για το TDNN έχουμε ποσοστά επιτυχούς πρόβλεψης 95.83%, ενώ για τον εξελικτικό αλγόριθμο MOSS έχουμε ποσοστά επιτυχίας 93.1%).

Μια άλλη προσέγγιση του συγκεκριμένου προβλήματος είναι αυτή της δημιουργίας συγκεκριμένων κανόνων που έχουν εξαχθεί για την E.Coli με βιολογική έρευνα[18] . Χαρακτηριστική εφαρμογή αποτελεί η "M-of-N concepts"[19], η οποία, αφού παρουσιάσει το σύνολο των κανόνων που μπορούν να συνδυαστούν σε μορφή AND-OR δέντρου υποστηρίζει ότι εξαιτίας της «αυστηρότητας» των κανόνων, ένας μικρότερος αριθμός αυτών σαν ισχύοντες κανόνες δίνει καλύτερα αποτελέσματα. Όπως παρατηρούμε στο Σχήμα 5.6 από το δέντρο που αναπαριστά το σύνολο των



Σχήμα 5.6: Το δέντρο που περιγράφει τους ισχύοντες κανόνες για υποκινητές E.Coli

κανόνων που πρέπει αρχικά να συνδυαστούν με όλους τους δυνατούς τρόπους, έχουμε $4^3 = 64$ δυνατούς συνδυασμούς. Στην συγκεκριμένη εφαρμογή επιλέγεται η απομάκρυνση του κλαδιού conformation και η σταδιακή μείωση των κανόνων που βρίσκονται σε ισχύ από N σε N-1, N-2 κοκ. Τα τελικά αποτελέσματα αποδεικνύουν ότι, βάσει του συνόλου δεδομένων από το UCI[20], τα μεγαλύτερα ποσοστά επιτυχίας δίνονται στην N-2 περίπτωση με το κλαδί conformation απενεργοποιημένο και μπορούν να φτάσουν μέχρι και 93.4%.

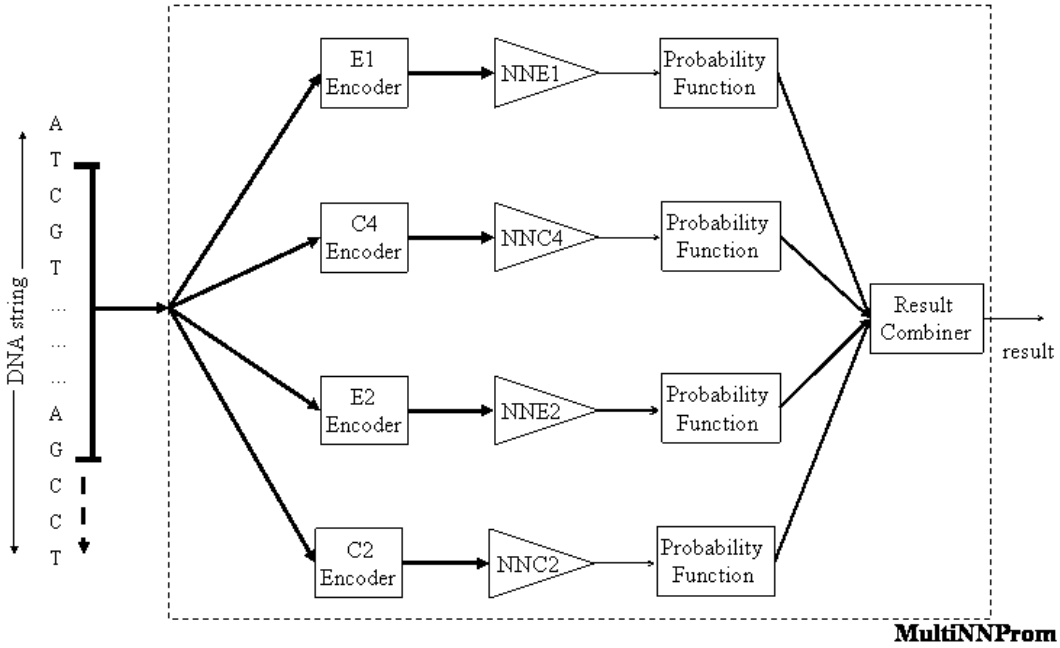
Μια άλλη ενδιαφέρουσα εφαρμογή που συνδυάζει τα νευρωνικά δίκτυα με τους γενετικούς αλγόριθμους είναι το MultiNNProm[21]. Το MultiNNProm συνδυάζει αρχικά τέσσερις ταξινομητές (classifiers). Ο συνδυασμός πολλαπλών ταξινομητών έχει αποδειχθεί ότι δίνει σε πολλές περιπτώσεις πολύ μεγαλύτερα ποσοστά επιτυχίας σε σύγκριση με ένα μόνο ταξινομητή. Σε κάθε ένα από τους τέσσερις ταξινομητές εισάγεται η ίδια ακολουθία DNA αλλά με διαφορετική κωδικοποίηση των νουκλεοτιδίων η οποία φαίνεται στον Πίνακα 5.3.

Ο συνδυασμός των αποτελεσμάτων μπορεί να γίνει με δυο διαφορετικές μεθόδους την LAP (Linear Average Prediction) και την LOP2 (τροποποίηση της μεθόδου LOP – Logarithmic Opinion Pool). Στην τελική έξοδο έχουμε το αποτέλεσμα που είναι ο συνδυασμός των τεσσάρων εξόδων των ταξινομητών. Το ενδιαφέρον σημείο στην συγκεκριμένη περίπτωση είναι η σχετική διαφοροποίηση που εμφανίζεται μεταξύ των

Πίνακας 5.3: Η διαφορετικές μορφές κωδικοποίησης που χρησιμοποιούνται στους 4 ταξινομητές του MultiNNProm

Encoding Method	Originally used by	A	T	C	G
E1		-2	-1	1	2
E2		-1	-1	1	1
C2	Demeler and Zhou, 1991	00	01	11	10
C4	Brunak et al., 1991	1000	0100	0010	0001

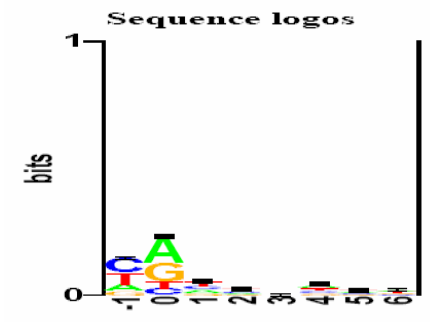
επιτυχών προγνώσεων που παρουσιάζουν οι τέσσερις ταξινομητές μεταξύ τους. Παρά το ότι εισάγουμε τις ίδιες ακολουθίες DNA υπάρχουν σαφώς καλύτερα αποτελέσματα στο νευρωνικό δίκτυο με το συμβολικό όνομα NNC4, αυτό δηλαδή με κωδικοποίηση τεσσάρων bits για το κάθε νουκλεοτίδιο. Τελικά το MultiNNProm (η αρχιτεκτονική



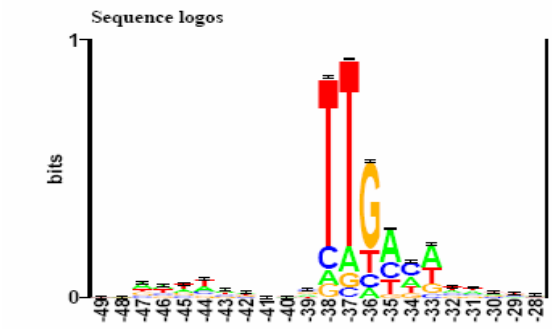
Σχήμα 5.7: Η αρχιτεκτονική του MultiNNProm

του παρουσιάζεται στο σχήμα 5.7) εμφανίζει ελαφρώς μεγαλύτερα ποσοστά σωστής πρόγνωσης σε σχέση με έναν ταξινομητή που χρησιμοποιεί κωδικοποίηση τεσσάρων bits.

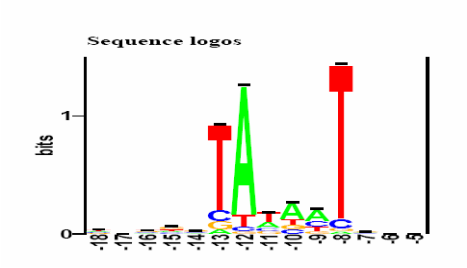
Σε προηγούμενο κεφάλαιο αναφέραμε τη χρήση στατιστικών μεθόδων για το πρόβλημα των υποκινητών στην E.Coli. Μια τέτοια προσέγγιση είναι και η χρήση του αλγόριθμου expectation-maximization (EM-algorithm). Στην εφαρμογή που εξετάζουμε παρακάτω ο EM-αλγόριθμος υλοποιείται σε συνδυασμό με νευρωνικά δίκτυα[22]. Αρχικά, ο EM-αλγόριθμος χρησιμοποιείται για να βρεθούν οι ακριβείς περιοχές -35 και -10. Πιο συγκεκριμένα, χρησιμοποιείται ένας μέγιστης τιμής εκ των υστέρων μπεϋζιανός αλγόριθμος (Bayesian maximum a posteriori (MAP)) EM-αλγόριθμος, ο οποίος μελετά τις δυο περιοχές ανεξάρτητα από τις αποστάσεις τόσο μεταξύ τους όσο και από το σημείο εκκίνησης της μεταγραφής (TSS). Το πλεονέκτημα αυτής της επιλογής είναι ότι δεν απαιτείται το μέγεθος της απόστασης να είναι σταθερό. Μετά τον καθορισμό των περιοχών -35 και -10 γίνεται η ευθυγράμμιση των ακολουθιών βάσει των περιοχών -35,-10 και του TSS (με χρήση sequence logo που εμφανίζονται στα σχήματα 5.8, 5.9, 5.10).



Σχήμα 5.8: Sequence logo στο TSS



Σχήμα 5.9: Sequence logo στην περιοχή -35



Σχήμα 5.10: Sequence logo στην περιοχή -10

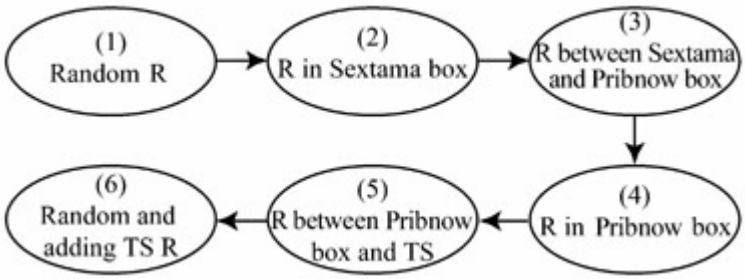
Στη συνέχεια η κωδικοποίηση που γίνεται είναι των τεσσάρων bits, ενώ για μια θέση, στην οποία δεν είναι γνωστό το νουκλεοτίδιο, αναπαρίσταται με X (X=1111). Επιλέγονται να κωδικοποιηθούν συνολικά 35 νουκλεοτίδια. Τα 17 από αυτά βρίσκονται στην περιοχή -35, τα 11 στην περιοχή -10 και τα άλλα 7 στην περιοχή του TSS.

Το νευρωνικό δίκτυο που δημιουργείται έχει ένα κρυμμένο επίπεδο με σιγμοειδείς συναρτήσεις ενεργοποίησης και είναι πλήρως συνδεδεμένο και εκπαιδευμένο σύμφωνα με τον scaled conjugate gradient αλγόριθμο. Έπειτα από δοκιμές οι νευρώνες του κρυμμένου επιπέδου επιλέχθηκαν να είναι 20. Το νευρωνικό δίκτυο εκπαιδεύτηκε 10 φορές. Τα πειραματικά αποτελέσματα είναι σε επίπεδο ακρίβειας 96.29%, και σε επίπεδο ευαισθησίας 91.78%.

Μια εφαρμογή η οποία αναλύει άγνωστης κατηγορίας υποκινητές E.Coli και η οποία βασίζεται σε Hidden Markov Models (HMM) παρουσιάζεται από τους Pederesen et. al[23]. Αναλυτικότερα, δεδομένου ότι οι υποκινητές μπορούν να κατηγοριοποιηθούν σε πολλές τάξεις ανάλογα με το ποια RNA-πολυμεράση ενεργοποιείται από αυτούς, δημιουργούν HMMs τα οποία κατηγοριοποιούν άγνωστης τάξης υποκινητές σε ήδη γνωστές για το HMM τάξεις αλλά παράλληλα έχει τη δυνατότητα να αναγνωρίζει νέες τάξεις υποκινητών. Τα HMMs που χρησιμοποιούνται ονομάζονται χαρακτηριστικά “super-HMMs” και αποτελούνται από HMMs παράλληλα δομημένα, το πλήθος των οποίων είναι ίσο με τον αριθμό των τάξεων των υποκινητών. Το super-HMM εκπαιδεύεται με ανταγωνιστική μάθηση. Το κυριότερο πλεονέκτημα της συγκεκριμένης μεθόδου είναι ότι τα HMMs δίνουν την δυνατότητα σωστής εκπαίδευσης με μικρά σύνολα δεδομένων.

Επέκταση της παραπάνω εφαρμογής αποτελεί η εργασία των L.Jian-Cheng et. al.[24]. Στο άρθρο αυτό ένα HMM με διάρκεια καταστάσεων (state duration density) συνδυάζεται τόσο με στατιστικές μεθόδους για τον υπολογισμό των αποστάσεων μεταξύ των περιοχών που μας ενδιαφέρουν, όσο και με εργαλεία και βάσεις βιολογικών δεδομένων που μπορούν να χρησιμοποιηθούν μέσω του διαδικτύου. Η διαδικασία που ακολουθήθηκε ήταν η εξής: αρχικά θεωρείται ότι δεν πρέπει να ακολουθηθεί η λογική της αντιστοιχίας 1:1 μεταξύ υποκινητή – γονιδίου, όπως στις περισσότερες περιπτώσεις μέχρι σήμερα. Αυτό γιατί, όπως αποδεικνύεται, μια σειρά από γονίδια ενεργοποιούνται μόνο από έναν υποκινητή. Για αυτόν τον λόγο παρουσιάζεται η έννοια της μονάδας μεταγραφής (Transcription Unit - TU). Το πρόβλημα λοιπόν μετατίθεται στην σωστή πρόβλεψη μιας τέτοιας περιοχής μέσω του υποκινητή, αφού η νέα αντιστοιχία είναι μεν 1:1 αλλά μεταξύ υποκινητή – TU. Μέσω του διαδικτυακού εργαλείου predictionRegulonDB[25] εκτιμήθηκαν τα αποτελέσματα που είχε το πείραμα. Στη συνέχεια ανακτήθηκαν 469 υποκινητές E.Coli από το predictionPromEC[26] και κατηγοριοποιήθηκαν βάσει των σ υπομονάδων(σ-factors). Για την πρόγνωση των περιοχών TU χρησιμοποιήθηκαν στατιστικές μέθοδοι οι οποίες βασίστηκαν στην απόσταση των γονιδίων και της λειτουργίας αυτών. Όσον αφορά την πρόγνωση των υποκινητών, μετά από διαχωρισμό σε δυο ομάδες των υποκινητών του συνόλου δεδομένων σε σ^{70} υποκινητές και μη σ^{70} υποκινητές και την δημιουργία ισομηκών ακολουθιών, πραγματοποιήθηκε στατιστική μελέτη για την συχνότητα εμφάνισης των νουκλεοτιδίων σε όλα τα bp (-75 έως +25 από το TSS). Τα αποτελέσματα ήταν προφανώς όμοια με αυτά των consensus boxes και των sequence logos, κυρίως στην ομάδα των σ^{70} υποκινητών. Τα αποτελέσματα αυτά

χρησιμοποιήθηκαν σε ένα HMM το οποίο δεν ακολουθεί καμία από τις παραδοσιακές μεθόδους εφαρμογής του σε βιολογικά προβλήματα (Forward-Backward procedure, Viterbi algorithm, Baum- Welch method), αλλά είναι ένα HMM με διάρκεια καταστάσεων (with state duration), του οποίου η αρχιτεκτονική φαίνεται στο Σχήμα 5.11.



Σχήμα 5.11: HMM με διάρκεια καταστάσεων (with state duration)

Τα πειραματικά αποτελέσματα της εφαρμογής αυτής σε επίπεδο πρόγνωσης υποκινητών κυμαίνονται μεταξύ 72 και 75.8%.

Πέρα από την παραπάνω αρθρογραφία, εξετάσαμε και μια σειρά άλλων άρθρων που προτείνουν μια σειρά παραπλήσιων ή σχετικά διαφοροποιημένων μεθόδων για το πρόβλημα της πρόβλεψης υποκινητών σε DNA ακολουθίες της E.Coli. Οι λογικές προσέγγισης του προβλήματος με δέντρα αποφάσεων[27], η χρήση της μεθόδου των k-πλησιέστερων γειτόνων (knn) και οι μηχανές υποστήριξης διανύσματος[28] παρουσιάζονται στις δημοσιεύσεις και τα άρθρα που παραθέτουμε στη βιβλιογραφία του κεφαλαίου.

---

[1] Becker, Kleinsmith, Hardin – The world of the cell – Benjamin Cummings, 2002

[2] Περίσσότερα για τους κινδύνους που έχουν μεταλλαγμένα στελέχη της E. Coli στο: http://www.cdc.gov/ncidod.dbmd/diseaseinfo/escherichiacoli.htm

[3] Για περισσότερα στοιχεία που αφορούν την E.Coli: http://ecoli.bham.ac.uk/title.html

[4] Shlomit Lisser and Hanah Margalit – Compilation of E.Coli mRNA promoter sequences – Nucleic Acids Research, Vol.21, No.7, 1507-1516, 1993

[5] Durbin R., Eddy S., Krogh A., Mitchison G. – Biological sequence analysis – Cambridge University Press, 1998

[6] Qicheng Ma, Jason T.L. Wang, James R. Gattiker – Mining biomolecular data using background knowledge and artificial neural networks

[7] Περισσότερα για τα sequence logos στον δικτυακό τόπο: http://www-lmmb.ncifcrf.gov/~toms/just.say.no/sequence.logo.html

[8] Ma Q., Wang J.T.L. – Recognizing promoters in DNA using Bayesian Neural Networks – Proceedings of the IASTED International Conference, Artificial Intelligence and Computing, August 9-12, 1999
Snyder E.E. , Stormo G. - Identification of Protein Coding Regions in Genomic DNA - Journal of Molecular Biology(1995) 248, 1-18
Uberbacher E.C., Mural R. J. - Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach - Proc. Natl. Acad. Sci. USA, Vol 88, 11261-11265, 1991

[9] Gary B. Fogel and David W. Corne – Evolutionary Computation in Bioinformatics – Morgan Kaufmann Publishers, 2003

10 Salzberg S., Delcher A.L., Fasman K.H. , Henderson J. - A Decision Tree System for Finding Genes in DNA - Journal of Computational Biology, 1997

11 Henderson, J., Salzberg, S. and Fasman, K. - Finding Genes in DNA with a Hidden Markov Model - Journal of Computational Biology, Vol. 4, No.2 (1997), pp. 127-141
Krogh, A. - Two Methods for Improving Performance of a HMM and Their Application for Gene Finding - Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology, AAAI Press, Menlo Park, CA,1997, pp. 179-186
Kulp, D., Haussler, D., Reese, M. G. and Eeckman, F. H. - A Generalized Hidden Markov Model for the Recognition of Human Genes in DNA - Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, AAAI Press, Menlo Park, CA, 1996, pp.134-142
Birney, E. - Hidden Markov Models in Biological Sequence Analysis - IBM Journal of Research and Development Volume 45, Numbers 3/4, 2001

[12] Koza J.R, Andre D. - Automatic Discovery of Protein Motifs Using Genetic Programming - Evolutionary Computation: Theory and Applications, 1995

[13] Ohno-Machado L., Vinterbo S., Webber G. - Classification of Gene Expression Data Using Fuzzy Logic - Decision Systems Group, …
Woolf P.J., Wang Y. - A Fuzzy Logic Approach to Analysing Gene Expression Data - Physiol Genomics, 3: 9-15, 2000

[14] P.Horton and M.Kanehisa – An assessment of neural network and statistical approaches for prediction of E.Coli promoter sites – Nucleic Acids Research, Vol.20, No.16, 4331-4338

[15] I.Mahadevan and I.Ghosh - Analysis of E.Coli promoter structures using neural networks – Nucleic Acids Research,1994, Vol.22, No. 11]

[16] G.G.Towell, J.W.Shavlik – Knowledge-Based Artificial Neural Networks – Artificial Intelligence 70 (1994), 119-165

[17] V.Cotik, R.Zalic, I.Zwir - A hybrid promoter analysis methodology for prokaryotic genomes – Fuzzy sets and systems, Elsevier
Το HPAM είναι ελεύθερο προς χρήση στον δικτυακό τόπο:
http://soar-tools.wustl.edu

[18] O' Neill, M.C. (1989) - Escherichia Coli promoters: Consensus as it relates to spacing class, specificity, repeat structure and three dimensional organization – Journal of Biological Chemistry, 264, 5522 – 5530

[19] J.Ortega – On the Informativeness of the DNA Promoter Sequences Domain Theory – Journal of Artificial Intelligence Research 2 (1995) 361 – 367

[20] http://www.ics.uci.edu/~mlearn/mlrepository.htm

[21] R.Ranawana – MultiNNProm: A Multi-Classifier System for Gene Finding

[22] Q.Ma, J.Wang, D.Shasha, C.Wu – DNA Sequence Classification via an Expectation Maximization Algorithm and Neural Networks: A Case Study – IEEE TRANSACTION ON SYSTEMS, MAN AND CYBERNETICS – PART C: APPLICATIONS AND REVIEWS – Vol.31, No.4, November 2001
Παρόμοια εφαρμογή περιγράφεται και στο άρθρο των: Q.Ma, J.Wang, J.Gattiker - Mining biomolecular data using background knowledge and artificial neural networks

[23] A.Pedersen, P.Baldi, S.Brunak, Y.Chauvin – Characterization of Prokaryotic and Eukaryotic Promoters Using Hidden Markov Models – Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology (ISMB98), 1998.

[24] LIN J.C., XU J.L., LUO J.H., LI Y.X. – Prediction of Prokaryotic Promoters Based on Prediction of Transcriptional Units – ACTA BIOCHIMICA et BIOPHYSICA SINICA 2003, 35(4):317-324

[25] http://kinich.cifn.unam.mx:8850/db/regulondb

[26] http://bioinfo.md.huji.ac.il/marg/promec/

[27] Jih –Wei Huang - Promoter Prediction in DNA sequences – 2003
Takashi Matsuda et al. – Constructing a decision tree for graph structured data..

[28] Nikola Kasabov, Shaoning Pang - Transductive support vector machines and applications in bioinformatics for promoter recognition

Κεφάλαιο 6

Εφαρμογή αναγνώρισης ακολουθιών υποκινητών στο DNA του βακτηρίου E.Coli

6 Εφαρμογή αναγνώρισης ακολουθιών υποκινητών στο DNA του βακτηρίου E.Coli

6.1 Γενικός σχεδιασμός

Το αντικείμενο της παρούσας διπλωματικής εργασίας στρέφεται γύρω από την αναγνώριση βιολογικών ακολουθιών υποκινητών (promoters), από την DNA ακολουθία του βακτηρίου της Escherichia Coli, με την δημιουργία νευρωνικού δικτύου, το οποίο λειτουργεί με τον αλγόριθμο ανάστροφης διάδοσης (backpropagation). Ο γενικός σχεδιασμός για την λήψη των πειραματικών μετρήσεων είναι ο εξής:

1. Δημιουργία συνόλων δεδομένων που να περιέχουν τόσο θετικά όσο και αρνητικά πρότυπα εισόδου.
2. Δοκιμές στις παραμέτρους του νευρωνικού δικτύου για την επιλογή εκείνου του συνδυασμού που δίνει τα καλύτερα δυνατά αποτελέσματα.

6.1.1 Δημιουργία συνόλων δεδομένων (data sets)

Η αρχική κωδικοποίηση στην οποία οι DNA ακολουθίες βρίσκονταν ήταν προφανώς μη δυαδική. Στο παράρτημα 3 παραθέτουμε την προέλευση των θετικών πρότυπων εισόδου (υποκινητές). Για την δημιουργία των αρνητικών πρότυπων εισόδου επιλέξαμε τρεις διαφορετικούς τρόπους που εμφανίζονται στη βιβλιογραφία[1].

Η πρώτη επιλογή ήταν η δημιουργία αρνητικών πρότυπων εισόδου βάσει του συνολικού αριθμού γονιδίων του συγκεκριμένου βακτηρίου. Η λογική ήταν η εξής: δεδομένου ότι κάθε υποκινητής σηματοδοτεί ένα γονίδιο και ακόμα περισσότερο την κωδικοποιούμενη περιοχή του, αν διαγράψουμε από κάθε γονίδιο τις περιοχές των υποκινητών είναι βέβαιο ότι δεν θα υπάρχει στην νέα ακολουθία DNA υποκινητής. Με αυτήν τη λογική δημιουργήθηκαν οι συναρτήσεις που παρουσιάζουμε στο παράρτημα 4 έτσι ώστε να δημιουργηθούν τελικά 1802 DNA ακολουθίες μη-υποκινητών E.Coli.

Η δεύτερη επιλογή ήταν οι ακολουθίες μη-υποκινητών να δημιουργηθούν μετά από την ανάγνωση ολόκληρου του γονιδιώματος της E.Coli, την επιλογή κάποιων υποακολουθιών από ένα ή περισσότερα σημεία του γονιδιώματος και τον μετέπειτα έλεγχο των υποακολουθιών αυτών έτσι ώστε να εξασφαλίζεται ότι η νέα ακολουθία δεν είναι συγχρόνως θετικό και αρνητικό πρότυπο εισόδου. Οι συναρτήσεις που δημιουργήθηκαν για την περίπτωση αυτή εμφανίζονται στο παράρτημα 4.

Η τρίτη και τελευταία επιλογή για τη δημιουργία αρνητικών προτύπων εισόδου ήταν η τυχαία δημιουργία ακολουθιών, οι οποίες να έχουν συχνότητα εμφάνισης αδενίνης και θυμίνης πάνω από 60% (60% AT-rich). Οι νέες ακολουθίες ελέγχθηκαν και πάλι για να εξασφαλίσουμε διαφορετικά θετικά και αρνητικά πρότυπα εισόδου. Οι συναρτήσεις που χρησιμοποιήθηκαν για αυτήν τη μέθοδο δημιουργίας αρνητικών πρότυπων εισόδου βρίσκονται στο παράρτημα 4.

Οι νέες DNA ακολουθίες που έχουμε πρέπει να μετασχηματιστούν με τέτοιον τρόπο ώστε να είναι δυνατή η ορθή ανάγνωση της πληροφορίας που αυτές δίνουν από το νευρωνικό δίκτυο. Σε μια σειρά από άρθρα[2] υποστηρίζεται ότι η καλύτερη αναπαράσταση σε δυαδική μορφή μιας ακολουθίας DNA είναι η ορθογώνια

κωδικοποίηση (orthogonal encoding), η οποία αναπαριστά κάθε νουκλεοτίδιο / γράμμα με τέσσερα bits, με ένα bit σε κάθε διαφορετική περίπτωση νουκλεοτιδίου στο 1 και τα υπόλοιπα στο 0. Πιο συγκεκριμένα, επιλέξαμε (βάσει της αρθρογραφίας που εξετάστηκε) A = 1000, C = 0100, G = 0010, T = 0001. Η αιτία όπου προτιμάται μια τέτοια αναπαράσταση και όχι κάποια άλλη (όπως για παράδειγμα με ακέραιους αριθμούς) είναι ότι σε αυτήν διατηρείται η απόσταση Hamming μεταξύ κάθε ζεύγους ανυσμάτων. Επιπλέον, εξαιτίας του συνόλου των ακολουθιών υποκινητών που χρησιμοποιήσαμε, όπου η κάθε ακολουθία έχει μήκος 101 bp, είναι προφανές ότι, μετά την ορθογωνική αναπαράσταση που επιλέξαμε να κάνουμε, κάθε ακολουθία θα έχει μήκος 101*4 = 404 bits. Η μετατροπή και η δημιουργία των αριθμητικών πια συνόλων δεδομένων ολοκληρώνεται με το «ανακάτεμα» των θετικών και αρνητικών πρότυπων εισόδου σε ένα σύνολο δεδομένων και με την εμφάνιση της κατηγορίας υποκινητής – μη υποκινητής (αντίστοιχα κατηγορία 1, κατηγορία 2). Όλες οι τελευταίες συναρτήσεις που ολοκληρώνουν τη δημιουργία των τριών διαφορετικών τελικών συνόλων δεδομένων που χρησιμοποιήσαμε στη συνέχεια, παρουσιάζονται στο παράρτημα 4. Τα σύνολα δεδομένων ονομάστηκαν DataX όπου X ο τρόπος που δημιουργήθηκαν τα αρνητικά πρότυπα εισόδου σύμφωνα με την παραπάνω περιγραφή (Data1, Data2, Data3). Κάθε σύνολο δεδομένων από αυτά είχε τελικά 471 ακολουθίες υποκινητών και 1802 ακολουθίες μη-υποκινητών (συνολικά 2273 ακολουθίες με μια αναλογία υποκινητών – μη-υποκινητών 1:4).

Τέλος, σημειώνουμε ότι εκτός από τα παραπάνω σύνολα δεδομένων χρησιμοποιήθηκε και αυτό από τη βάση δεδομένων UCI[3]. Διαφέρει σε σχέση με τα άλλα στο ότι έχει μήκος 57 bp, στο ότι η αναλογία μεταξύ υποκινητών – μη υποκινητών είναι 1:1, ενώ το συνολικό πλήθος των προτύπων εισόδου είναι 106. Εξαιτίας του μικρού αριθμού των προτύπων αλλά και του μικρότερου μήκους της ακολουθίας, προτιμήσαμε οι πειραματικές μετρήσεις να γίνουν στα 3 σύνολα δεδομένων που παρουσιάσαμε παραπάνω, ενώ ο προσδιορισμός των παραμέτρων του νευρωνικού δικτύου έγινε κυρίως με αυτό το σύνολο δεδομένων αφού έχει μικρότερο μέγεθος με αποτέλεσμα να έχουμε και ταχύτερη διαδικασία εκπαίδευσης ώστε να εκτιμήσουμε τα αποτελέσματα που είχε το νευρωνικό δίκτυο. Πρέπει επίσης να σημειώσουμε ότι όλες οι ακολουθίες που χρησιμοποιήθηκαν σαν θετικά πρότυπα εισόδου ήταν στοιχισμένες εκ των προτέρων, γι' αυτό και δεν χρειάστηκε να χρησιμοποιήσουμε κάποιο εργαλείο ευθυγράμμισης (alignment tool).

6.1.2 Επιλογή παραμέτρων

Πέρα από τα σύνολα δεδομένων, υπάρχει μια σειρά από μεταβλητούς παράγοντες που καθορίζουν τα πειραματικά αποτελέσματα που θα έχουμε στη συνέχεια. Ο αριθμός των κρυμμένων επιπέδων του νευρωνικού δικτύου, το πλήθος των νευρώνων που υπάρχουν σε κάθε ένα από τα επίπεδα του νευρωνικού δικτύου, οι συναρτήσεις εκπαίδευσης που χρησιμοποιούνται όπως επίσης και το πλήθος των εποχών εκπαίδευσης καθορίζουν σε μεγάλο βαθμό την απόδοση και ακρίβεια πρόβλεψης του συστήματος.

Μετά από μια σειρά δοκιμών κυρίως στο σύνολο δεδομένων με μήκος ακολουθίας 57 bp καταλήξαμε στις εξής επιλογές παραμέτρων:

- Η συνάρτηση με την καλύτερη συμπεριφορά σε σχέση με τις υπόλοιπες είναι η traingda
- Ο βέλτιστος αριθμός των νευρώνων με 1 κρυμμένο επίπεδο είναι 16

- Με την εισαγωγή και δεύτερου κρυμμένου επιπέδου ο βέλτιστος αριθμός νευρώνων για αυτό είναι 6, άρα τελικά ο βέλτιστος αριθμός νευρώνων για τα δύο κρυμμένα επίπεδα είναι 16-6
- Ο αριθμός αυτός των νευρώνων δοκιμάστηκε σε όλες τις προσφερόμενες από την Matlab συναρτήσεις εκπαίδευσης εκτός των trainlm και trainbfg όπου στο σύστημα στο οποίο έγιναν οι δοκιμές, έδινε μήνυμα σφάλματος λόγω έλλειψης μνήμης

6.2 Εκτέλεση της εφαρμογής και πειραματικά αποτελέσματα

Το τελικό στάδιο της εφαρμογής μας είναι ο καθορισμός των συναρτήσεων μεταφοράς των νευρώνων και το πλήθος των εποχών εκπαίδευσης. Στη συνέχεια μέσω γραφικών παραστάσεων εξετάζουμε τα αποτελέσματα με κριτήρια το μέσο τετραγωνικό σφάλμα καθώς και την ακρίβεια κατηγοριοποίησης στα άλλα σύνολα δεδομένων από αυτό στο οποίο το ΝΔ εκπαιδεύτηκε (εξετάζεται ουσιαστικά η ικανότητα γενίκευσης του υλοποιηθέντος νευρωνικού δικτύου).

6.2.1 Τεχνικά χαρακτηριστικά και επιλογή παραμέτρων για το backpropagation νευρωνικό δίκτυο

Τα πειράματα με τον αλγόριθμο ανάστροφης διάδοσης πραγματοποιήθηκαν σε Matlab (Version 6.5.0.180913a Release13), ενώ το χρησιμοποιούμενο hardware είναι Intel P4 2.53GHz με 256MB RAM. Τα σύνολα δεδομένων που χρησιμοποιούνται είναι κυρίως τα Data1, Data2 και Data3.

Η Matlab δίνει την δυνατότητα επιλογής μεταξύ τριών συναρτήσεων μεταφοράς για κάθε νευρώνα: τις tansig, logsig, και purelin. Σε όλα τα πειράματα χρησιμοποιείται η logsig, αφού μετά από δοκιμές θεωρήθηκε η καταλληλότερη. Αντίστοιχα, το μέσο τετραγωνικό σφάλμα (MSE) θεωρήθηκε η πιο κατάλληλη συνάρτηση επίδοσης και χρησιμοποιήθηκε σε όλα τα πειράματα έναντι του μέσου απόλυτου σφάλματος (MAE), του αθροίσματος του μέσου τετραγωνικού σφάλματος και της μέσης τετραγωνικής μεταβολής των weights και bias (MSEREG). Πρέπει να σημειωθεί ότι το πλήθος των εποχών επιλέγεται σε άμεση σχέση με τη συνάρτηση εκπαίδευσης που χρησιμοποιούμε σε κάθε περίπτωση, για αυτόν το λόγο ποικίλλει.

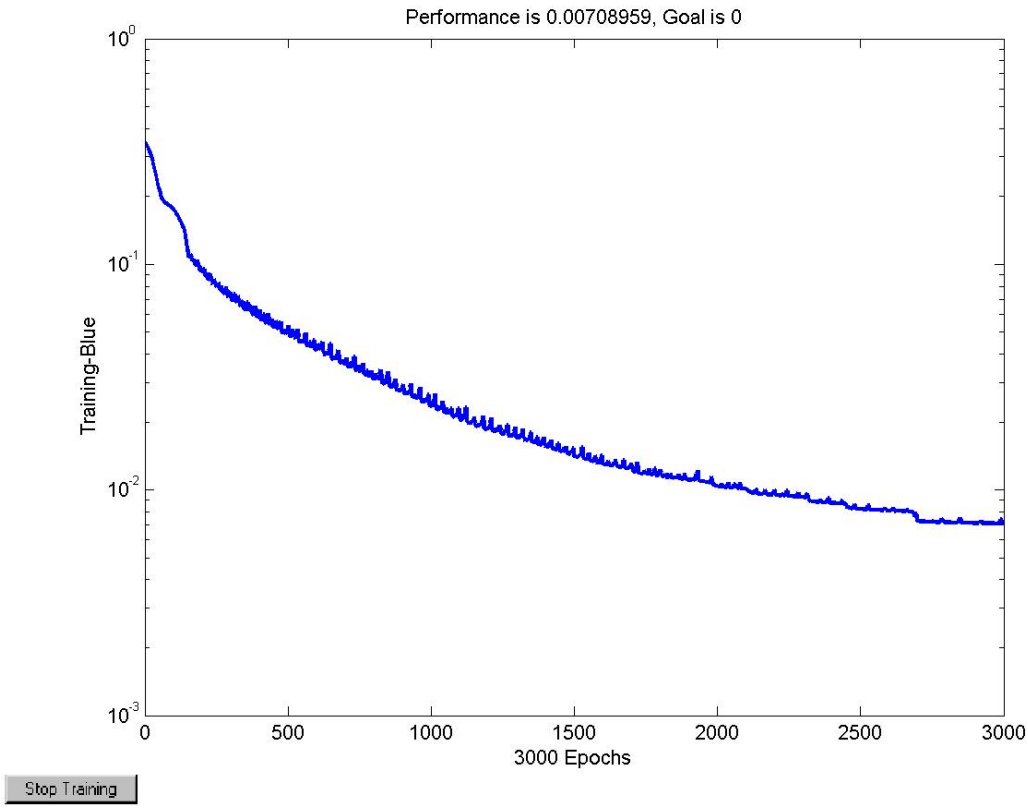
6.2.2 Πειραματικά αποτελέσματα

Για κάθε δυνατό συνδυασμό συνάρτησης, εποχών, νευρώνων και επιπέδων γίνονται 3 προσομοιώσεις (3 σύνολα δεδομένων). Κάθε μια φορά βρίσκεται ο αριθμός των προτύπων που δεν έχουν ταξινομηθεί σωστά πριν και μετά την εκπαίδευση. Τόσο το σύνολο δεδομένων που χρησιμοποιήθηκε κατά την εκπαίδευση, όσο και για τα σύνολα δεδομένων που δεν χρησιμοποιήθηκαν κατά τη φάση της εκπαίδευσης, υπολογίζεται το πλήθος των λανθασμένα ταξινομημένων προτύπων.

Ακολουθούν οι πίνακες αποτελεσμάτων και οι γραφικές παραστάσεις από την Matlab που απεικονίζουν το μέσο τετραγωνικό σφάλμα MSE.

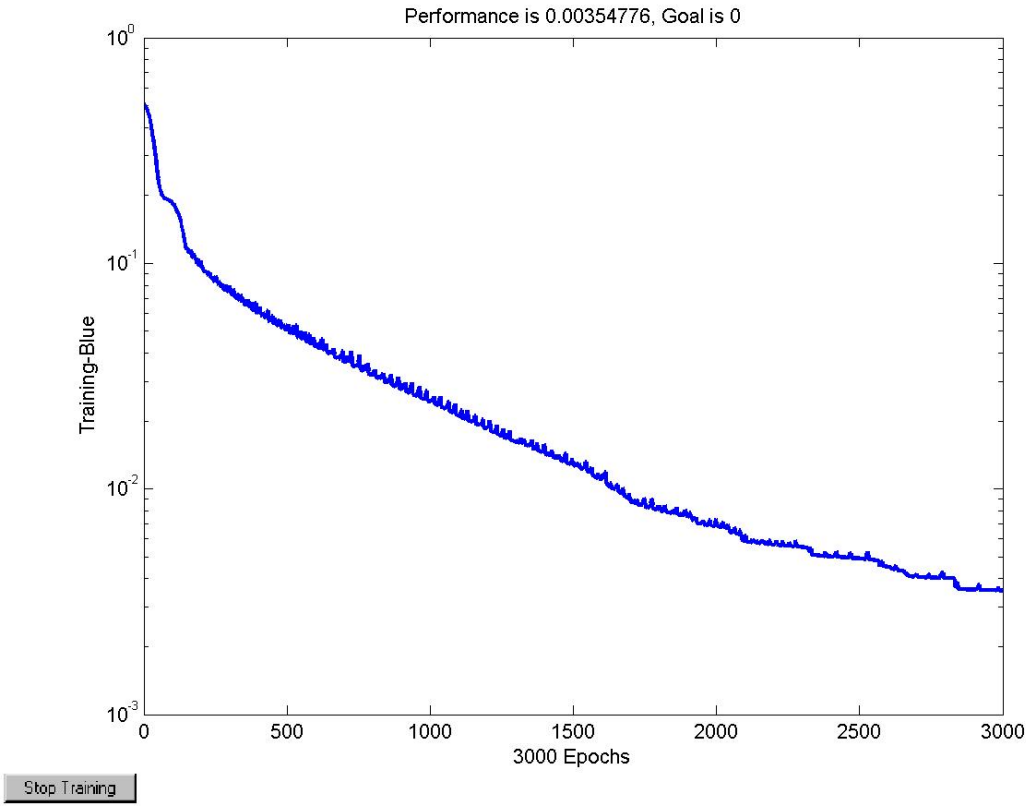
Πείραμα 1

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	6
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	0
Εποχές εκπαίδευσης (epochs)	3000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1163
Λάθη μετά την εκπαίδευση του Data1	346
Λάθη μετά την εκπαίδευση του Data2	111
Λάθη μετά την εκπαίδευση του Data3	16



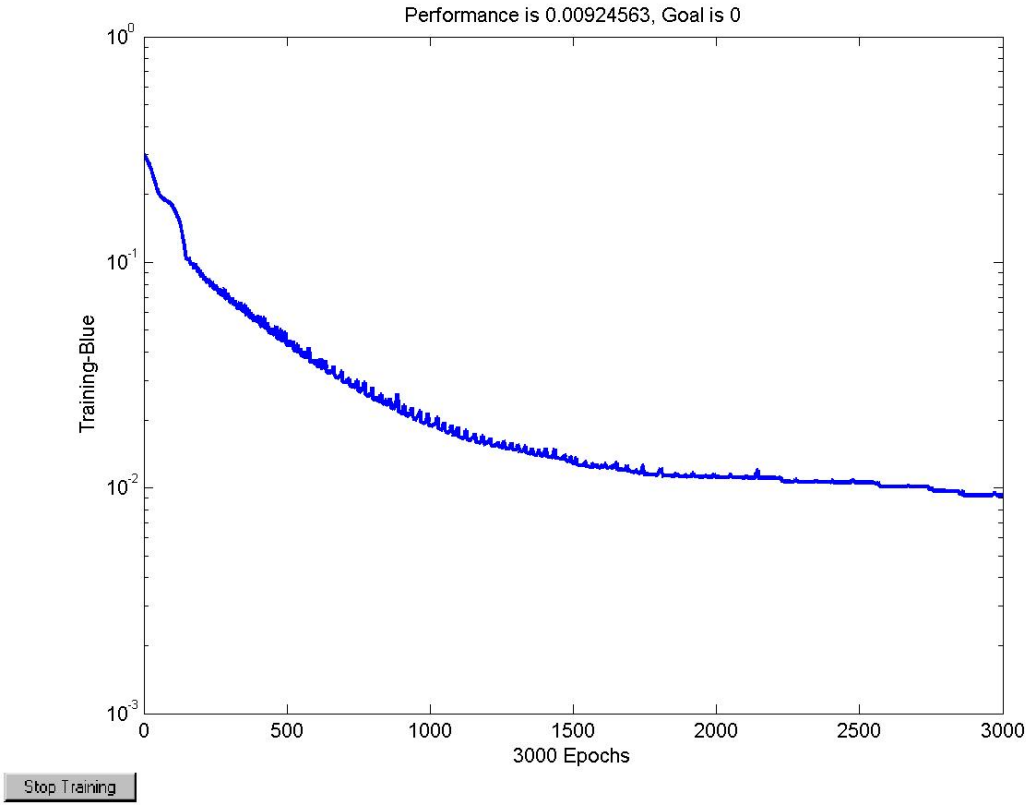
Πείραμα 4

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	8
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	0
Εποχές εκπαίδευσης (epochs)	3000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1566
Λάθη μετά την εκπαίδευση του Data1	333
Λάθη μετά την εκπαίδευση του Data2	73
Λάθη μετά την εκπαίδευση του Data3	8



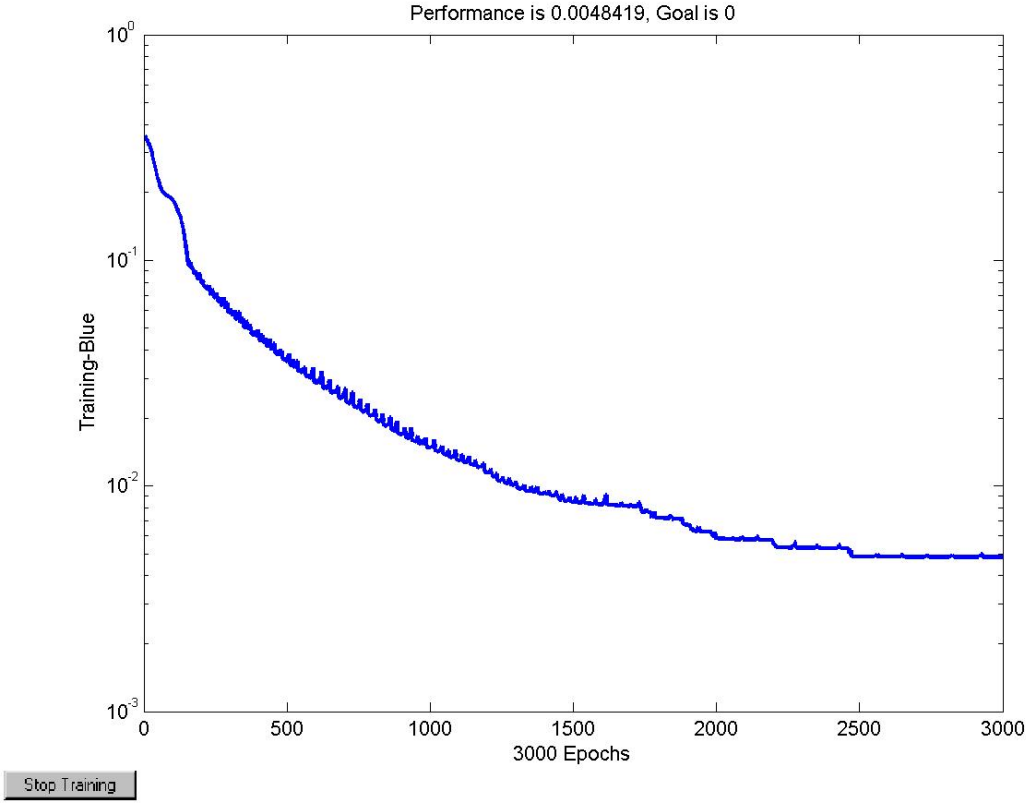
Πείραμα 5

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	10
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	0
Εποχές εκπαίδευσης (epochs)	3000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1010
Λάθη μετά την εκπαίδευση του Data1	308
Λάθη μετά την εκπαίδευση του Data2	95
Λάθη μετά την εκπαίδευση του Data3	21



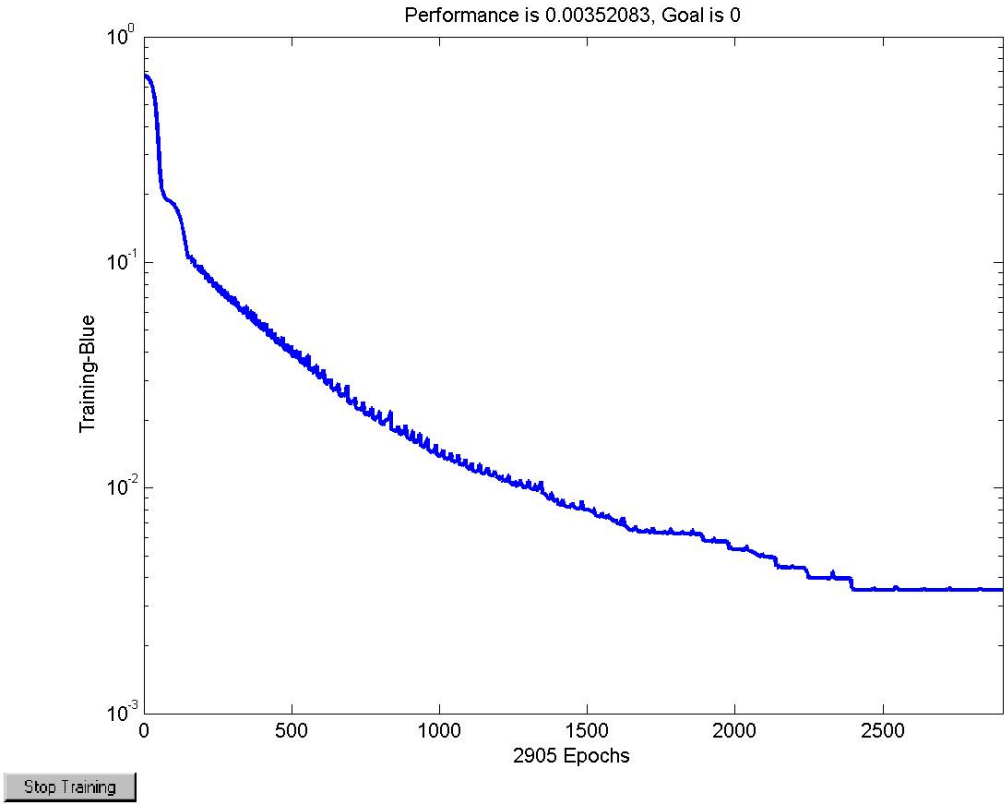
Πείραμα 6.

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	12
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	0
Εποχές εκπαίδευσης (epochs)	3000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1237
Λάθη μετά την εκπαίδευση του Data1	307
Λάθη μετά την εκπαίδευση του Data2	76
Λάθη μετά την εκπαίδευση του Data3	11



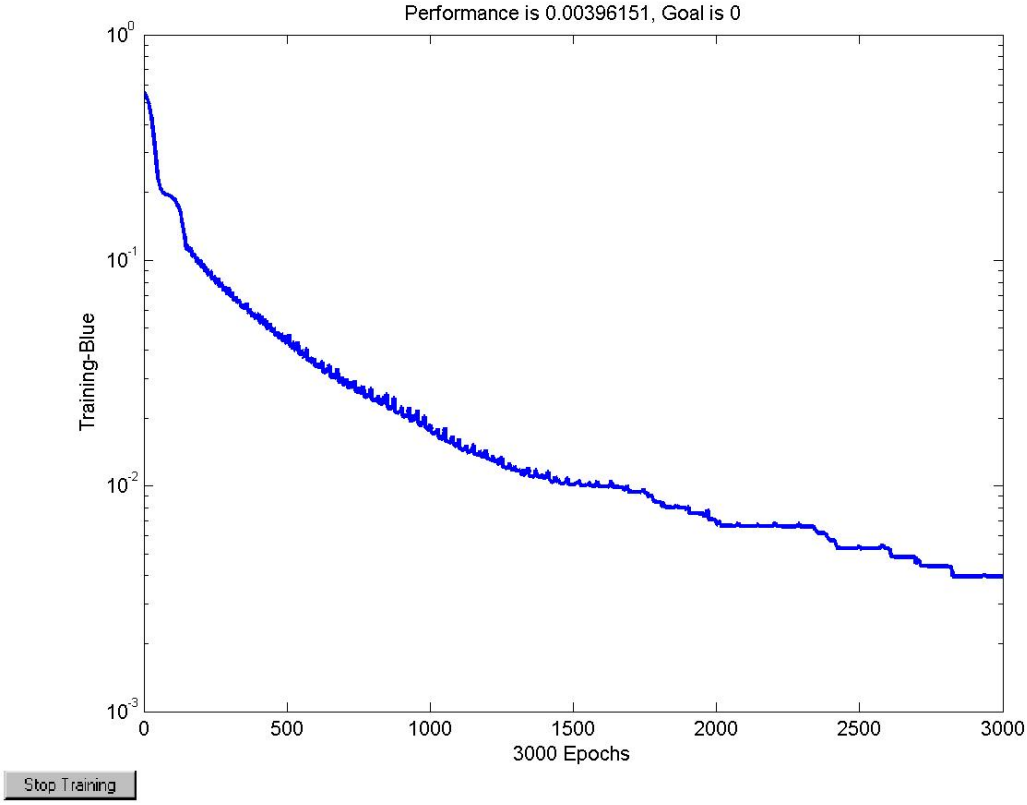
Πείραμα 7

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	0
Εποχές εκπαίδευσης (epochs)	3000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1760
Λάθη μετά την εκπαίδευση του Data1	265
Λάθη μετά την εκπαίδευση του Data2	104
Λάθη μετά την εκπαίδευση του Data3	8



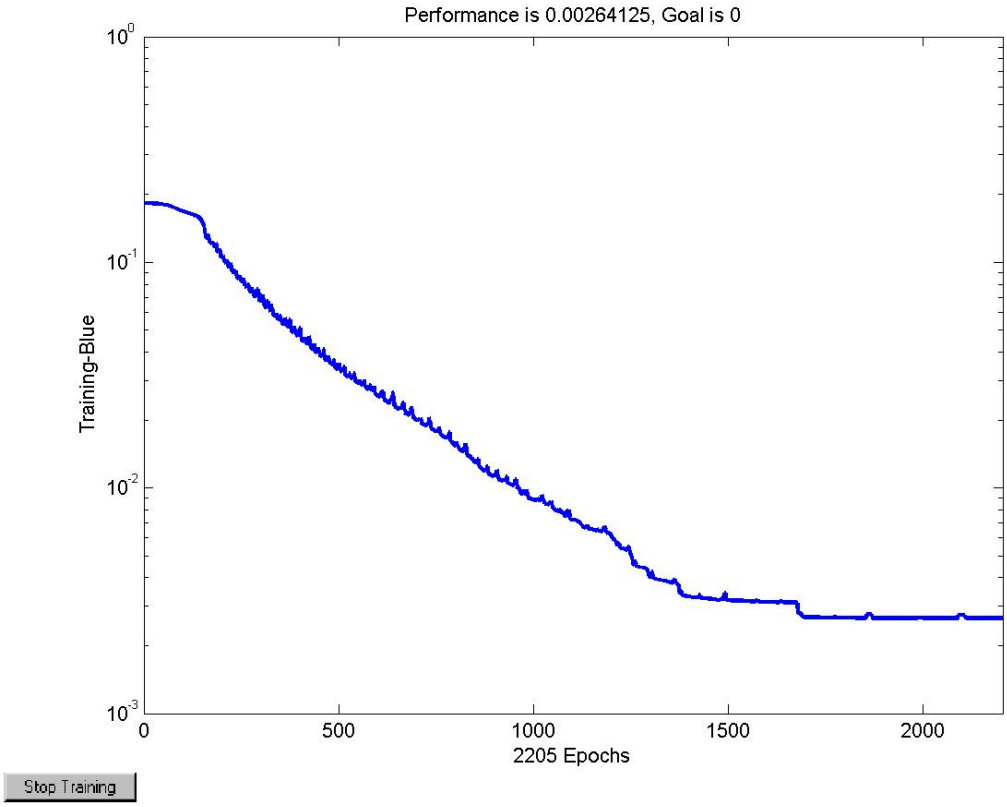
Πείραμα 8

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	18
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	0
Εποχές εκπαίδευσης (epochs)	3000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1635
Λάθη μετά την εκπαίδευση του Data1	298
Λάθη μετά την εκπαίδευση του Data2	46
Λάθη μετά την εκπαίδευση του Data3	9



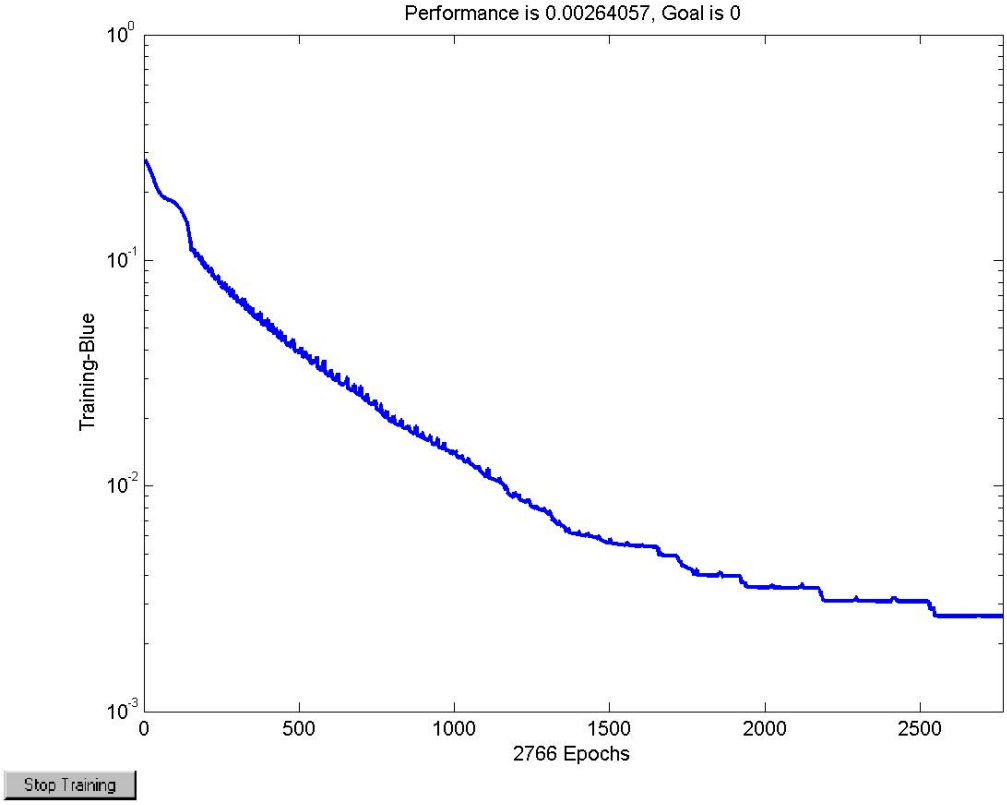
Πείραμα 9.

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	4
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	468
Λάθη μετά την εκπαίδευση του Data1	306
Λάθη μετά την εκπαίδευση του Data2	97
Λάθη μετά την εκπαίδευση του Data3	6



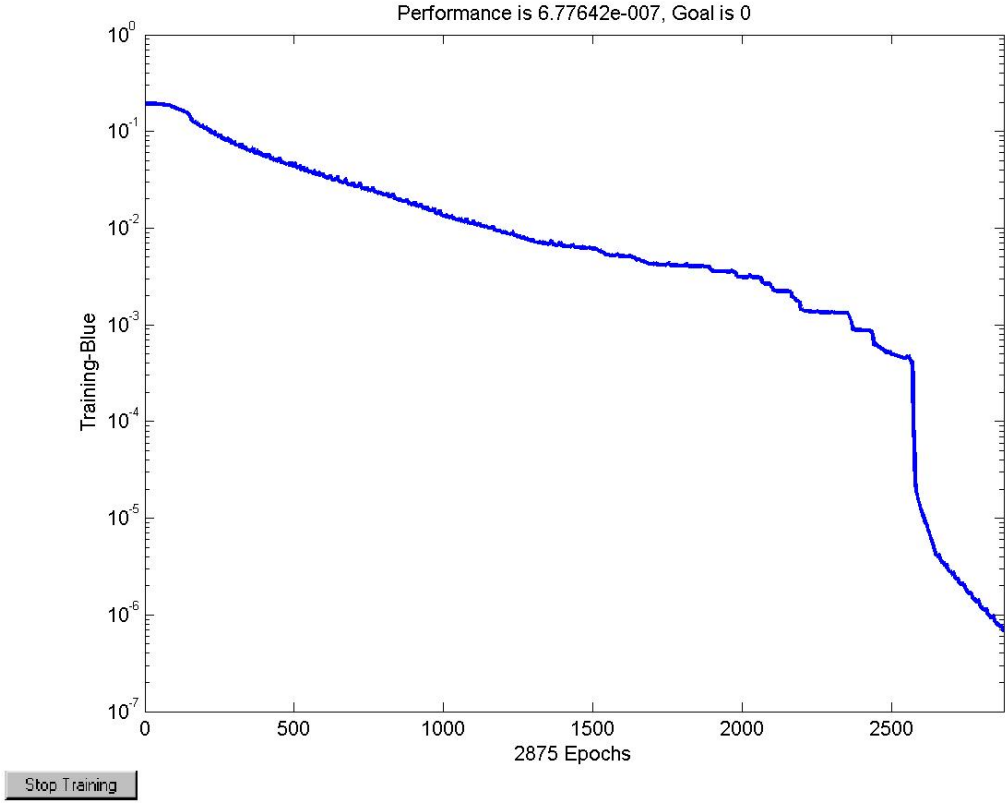
Πείραμα 10

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	5
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1063
Λάθη μετά την εκπαίδευση του Data1	290
Λάθη μετά την εκπαίδευση του Data2	34
Λάθη μετά την εκπαίδευση του Data3	6



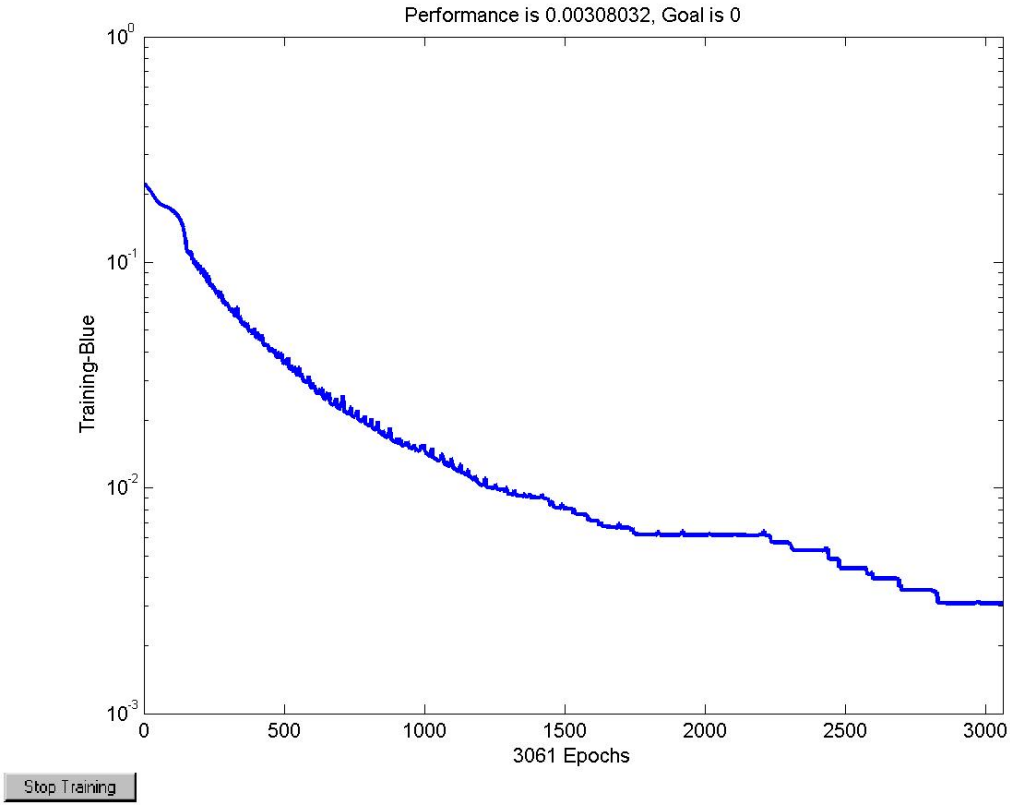
Πείραμα 11

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	549
Λάθη μετά την εκπαίδευση του Data1	264
Λάθη μετά την εκπαίδευση του Data2	44
Λάθη μετά την εκπαίδευση του Data3	0



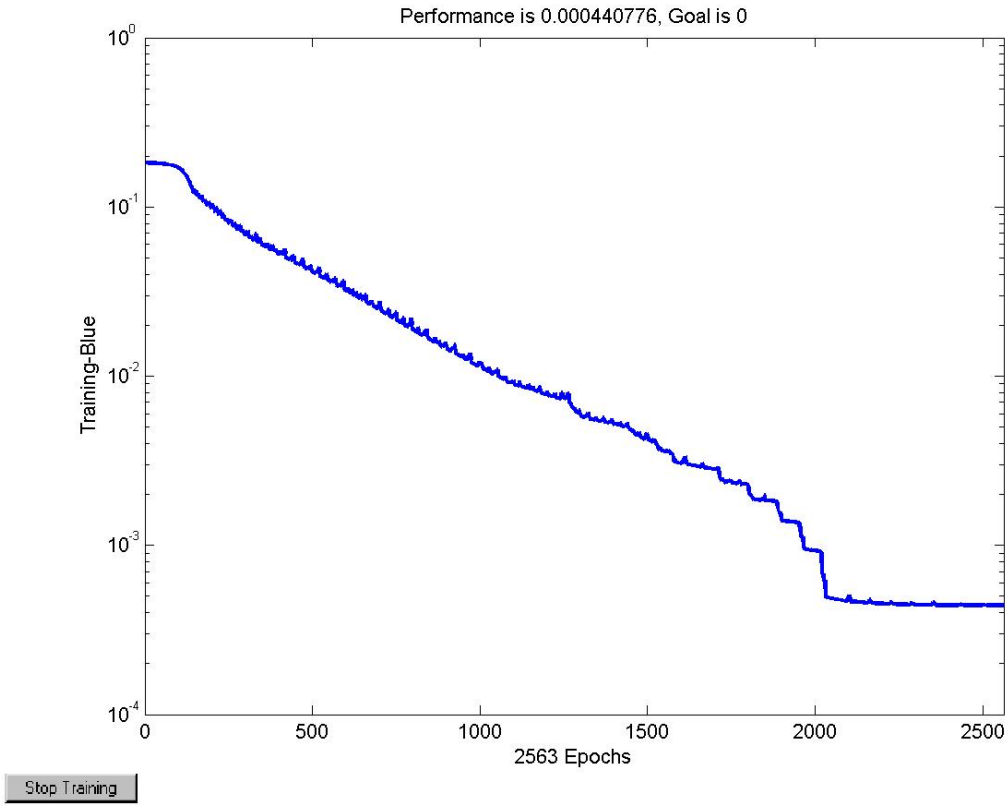
Πείραμα 12

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	7
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	764
Λάθη μετά την εκπαίδευση του Data1	294
Λάθη μετά την εκπαίδευση του Data2	54
Λάθη μετά την εκπαίδευση του Data3	7



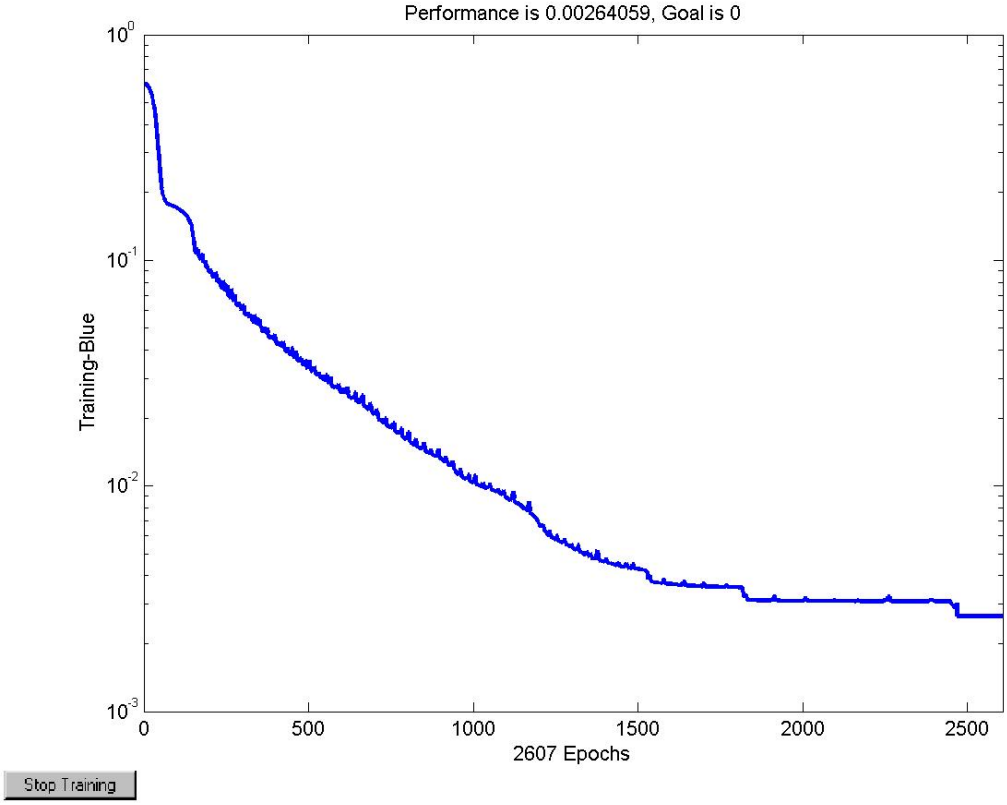
Πείραμα 13

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	8
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	523
Λάθη μετά την εκπαίδευση του Data1	350
Λάθη μετά την εκπαίδευση του Data2	57
Λάθη μετά την εκπαίδευση του Data3	1



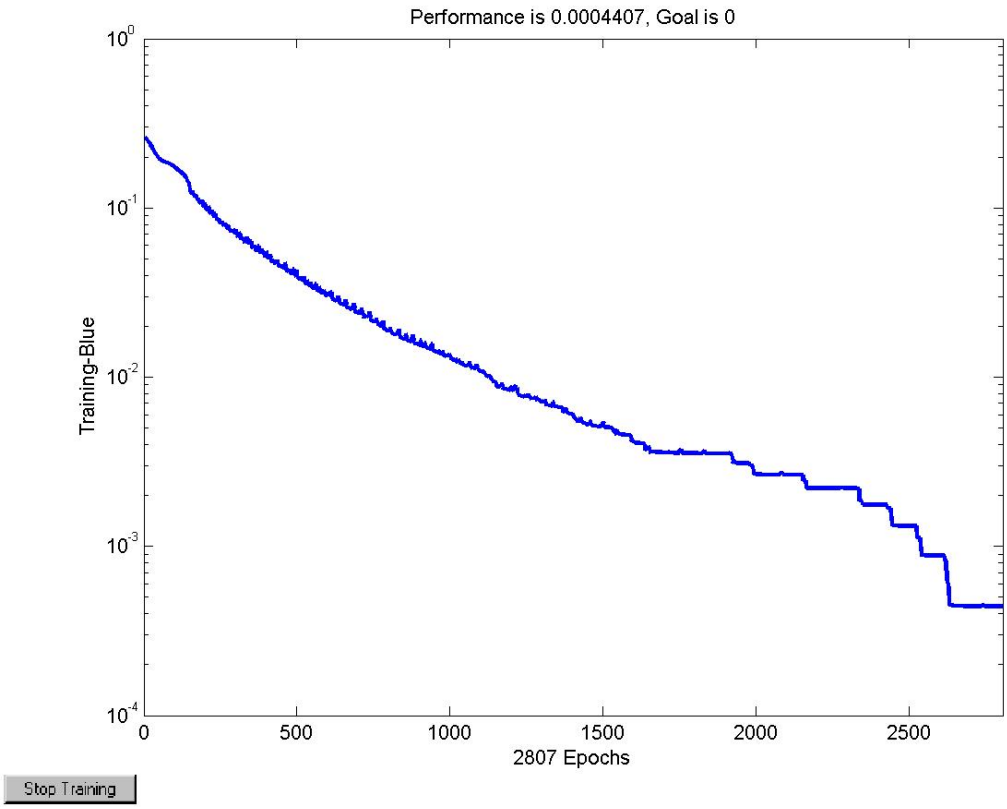
Πείραμα 14

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	9
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1781
Λάθη μετά την εκπαίδευση του Data1	294
Λάθη μετά την εκπαίδευση του Data2	81
Λάθη μετά την εκπαίδευση του Data3	6



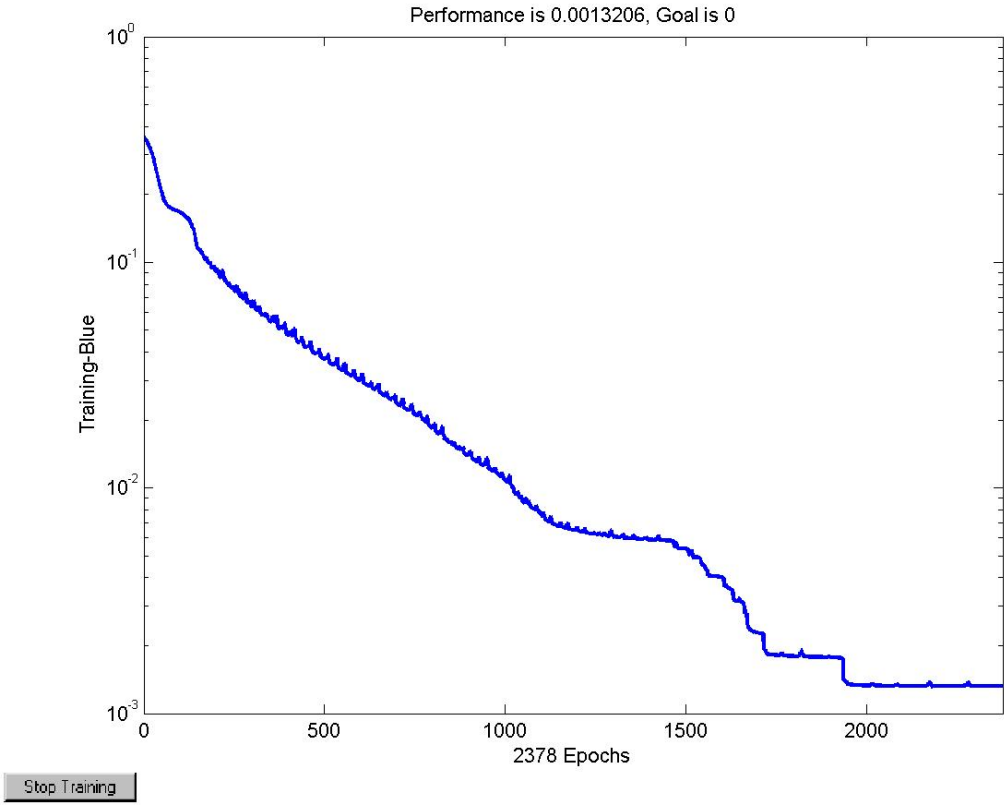
Πείραμα 15

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	10
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	949
Λάθη μετά την εκπαίδευση του Data1	349
Λάθη μετά την εκπαίδευση του Data2	48
Λάθη μετά την εκπαίδευση του Data3	1



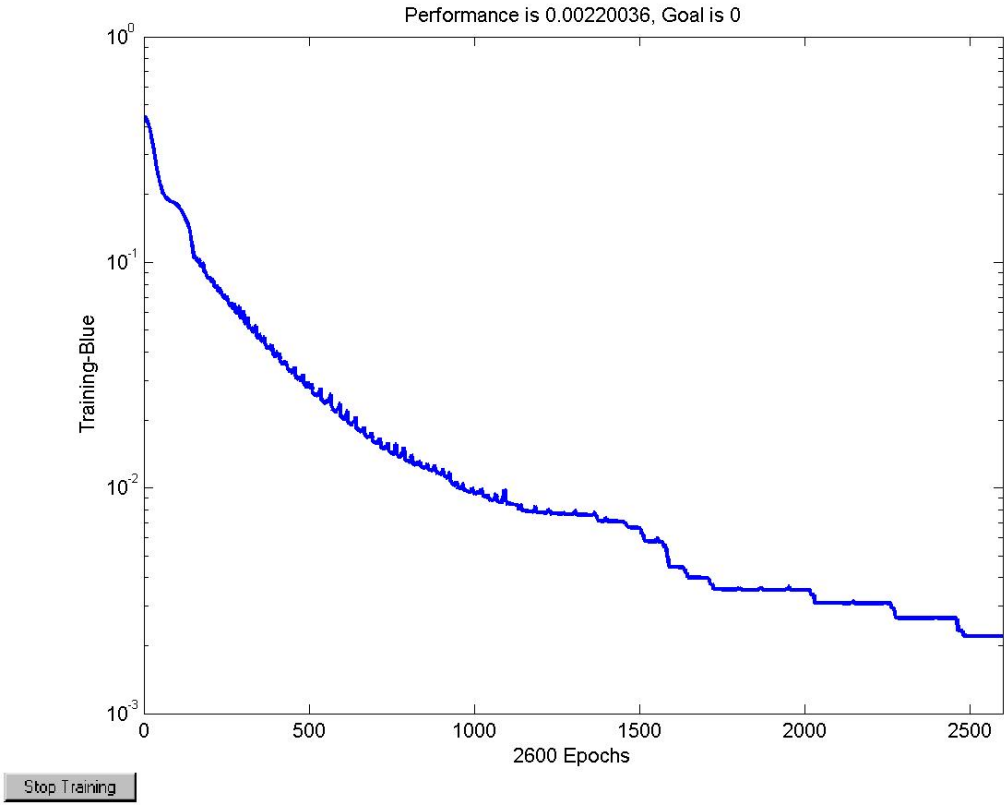
Πείραμα 16

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	11
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1330
Λάθη μετά την εκπαίδευση του Data1	326
Λάθη μετά την εκπαίδευση του Data2	127
Λάθη μετά την εκπαίδευση του Data3	3



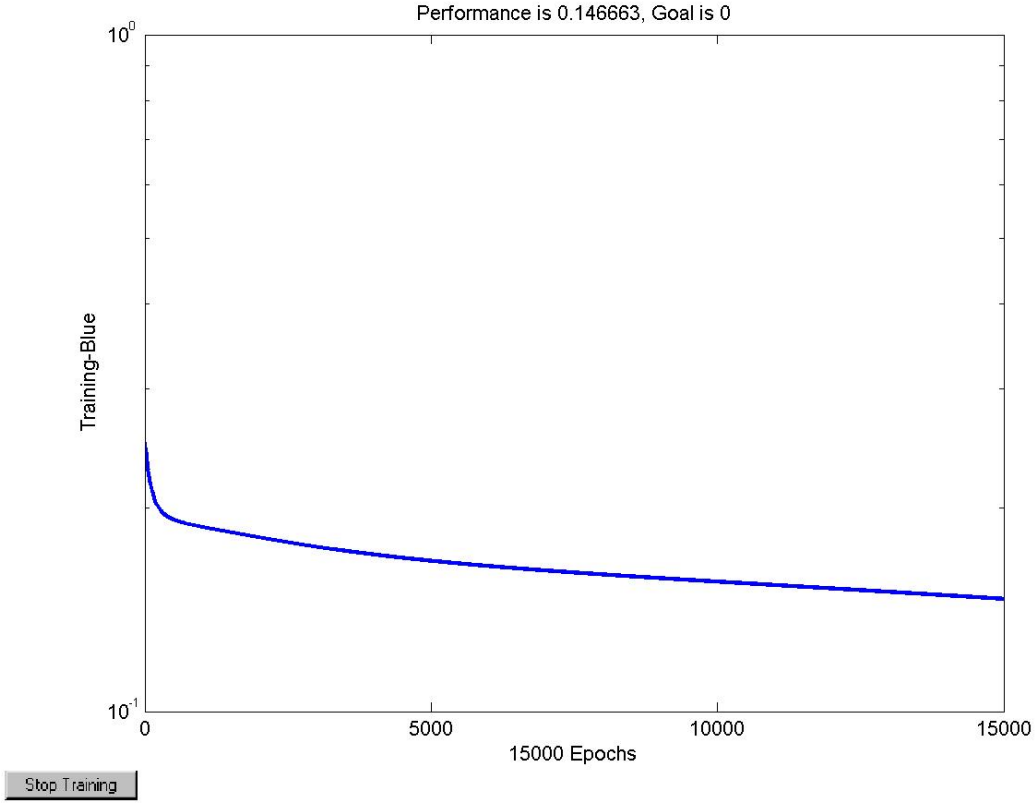
Πείραμα 17

Συνάρτηση εκπαίδευσης	traingda
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	12
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1330
Λάθη μετά την εκπαίδευση του Data1	316
Λάθη μετά την εκπαίδευση του Data2	173
Λάθη μετά την εκπαίδευση του Data3	5



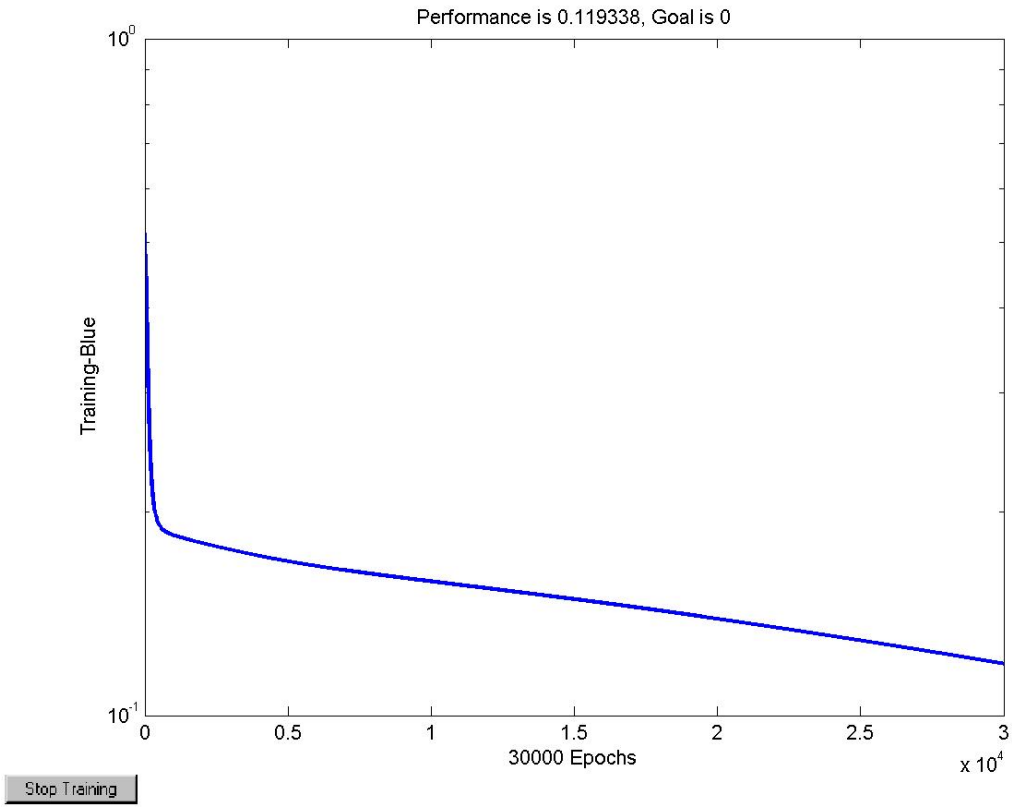
Πείραμα 20

Συνάρτηση εκπαίδευσης	traingd
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	15000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	889
Λάθη μετά την εκπαίδευση του Data1	501
Λάθη μετά την εκπαίδευση του Data2	435
Λάθη μετά την εκπαίδευση του Data3	444



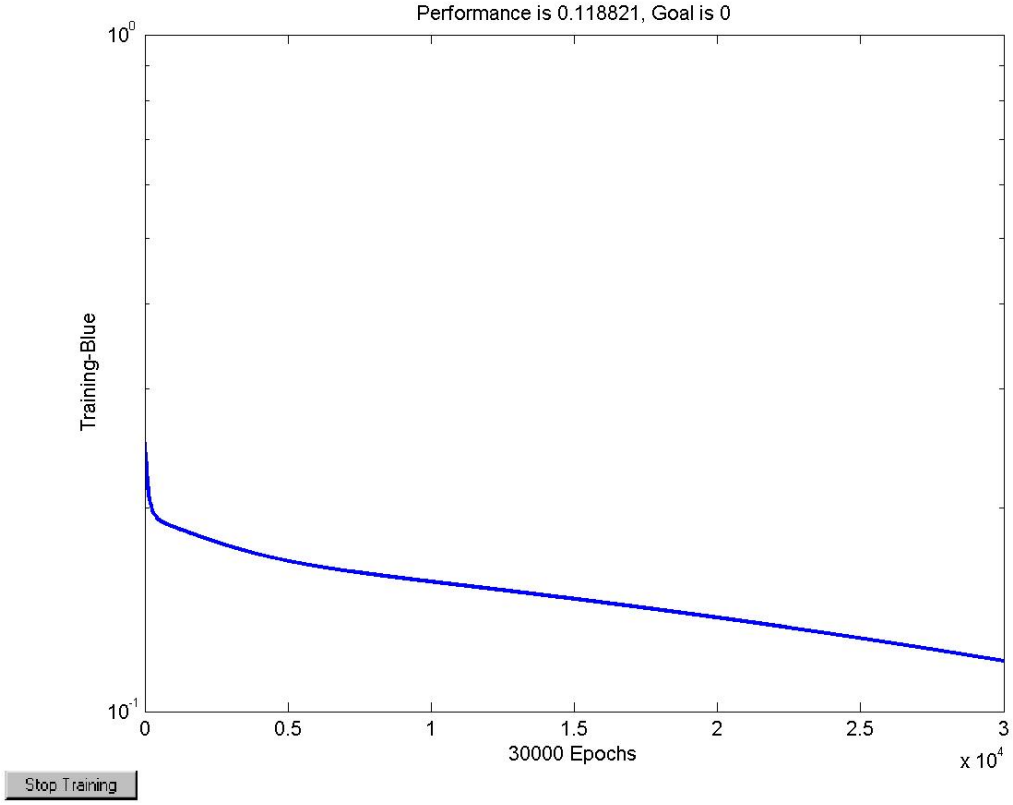
Πείραμα 21

Συνάρτηση εκπαίδευσης	traingd
Αριθμός νευρώνων στο 1ο κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2ο κρυμμένο επίπεδο (layer2)	12
Εποχές εκπαίδευσης (epochs)	30000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1668
Λάθη μετά την εκπαίδευση του Data1	457
Λάθη μετά την εκπαίδευση του Data2	398
Λάθη μετά την εκπαίδευση του Data3	330



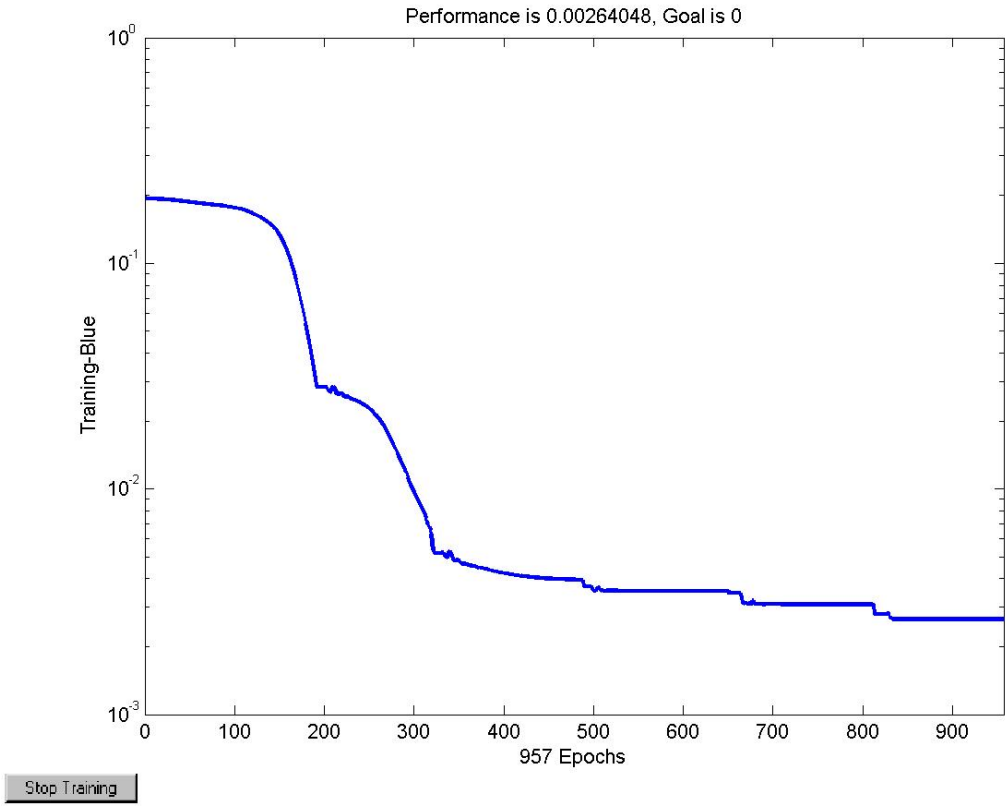
Πείραμα 24

Συνάρτηση εκπαίδευσης	traingdm
Αριθμός νευρώνων στο 1^{ο} κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2^{ο} κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	30000
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	889
Λάθη μετά την εκπαίδευση του Data1	496
Λάθη μετά την εκπαίδευση του Data2	329
Λάθη μετά την εκπαίδευση του Data3	349



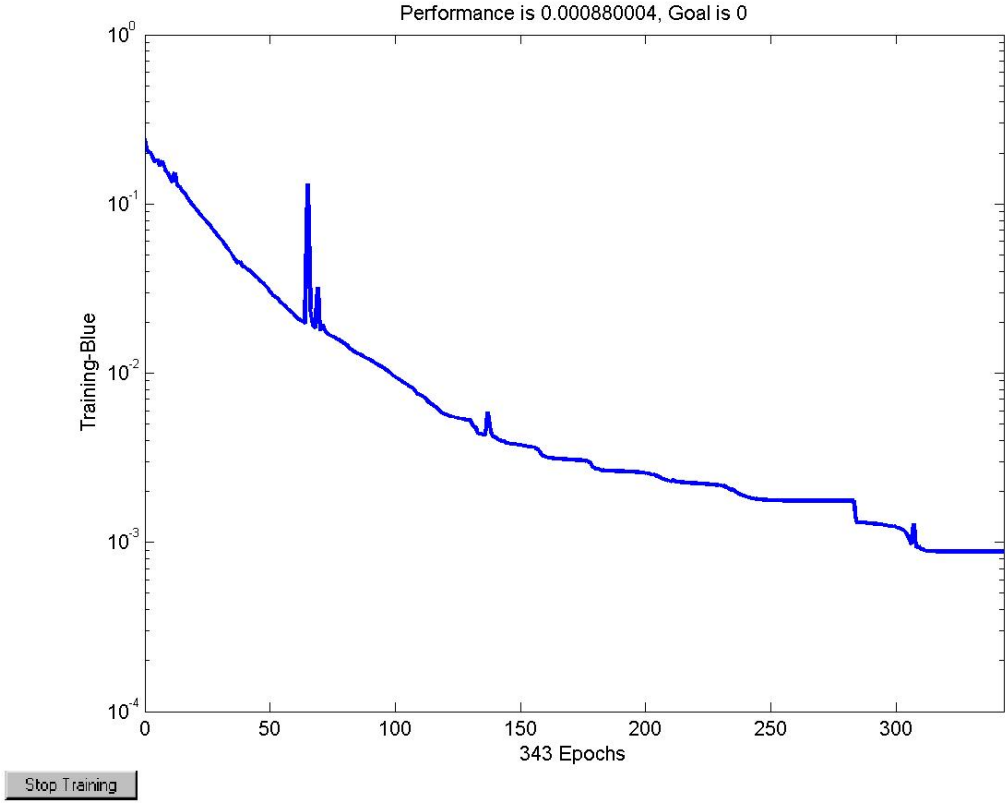
Πείραμα 28

Συνάρτηση εκπαίδευσης	traingdx
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	568
Λάθη μετά την εκπαίδευση του Data1	294
Λάθη μετά την εκπαίδευση του Data2	42
Λάθη μετά την εκπαίδευση του Data3	6



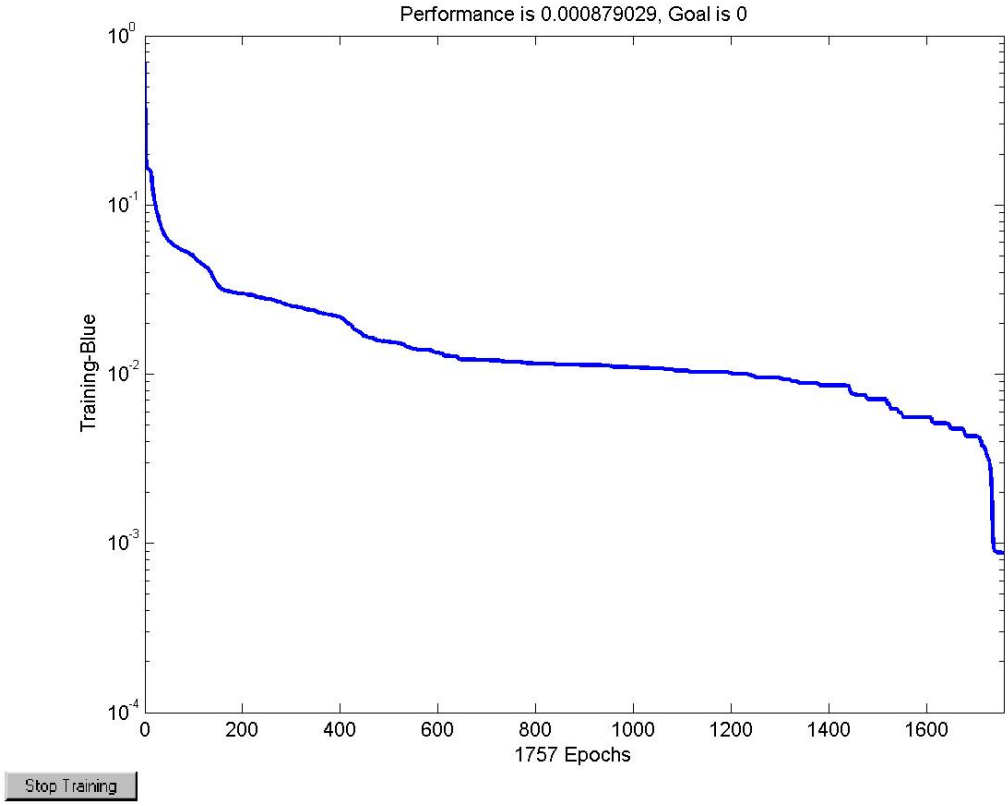
Πείραμα 31

Συνάρτηση εκπαίδευσης	trainrp
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	808
Λάθη μετά την εκπαίδευση του Data1	272
Λάθη μετά την εκπαίδευση του Data2	123
Λάθη μετά την εκπαίδευση του Data3	2



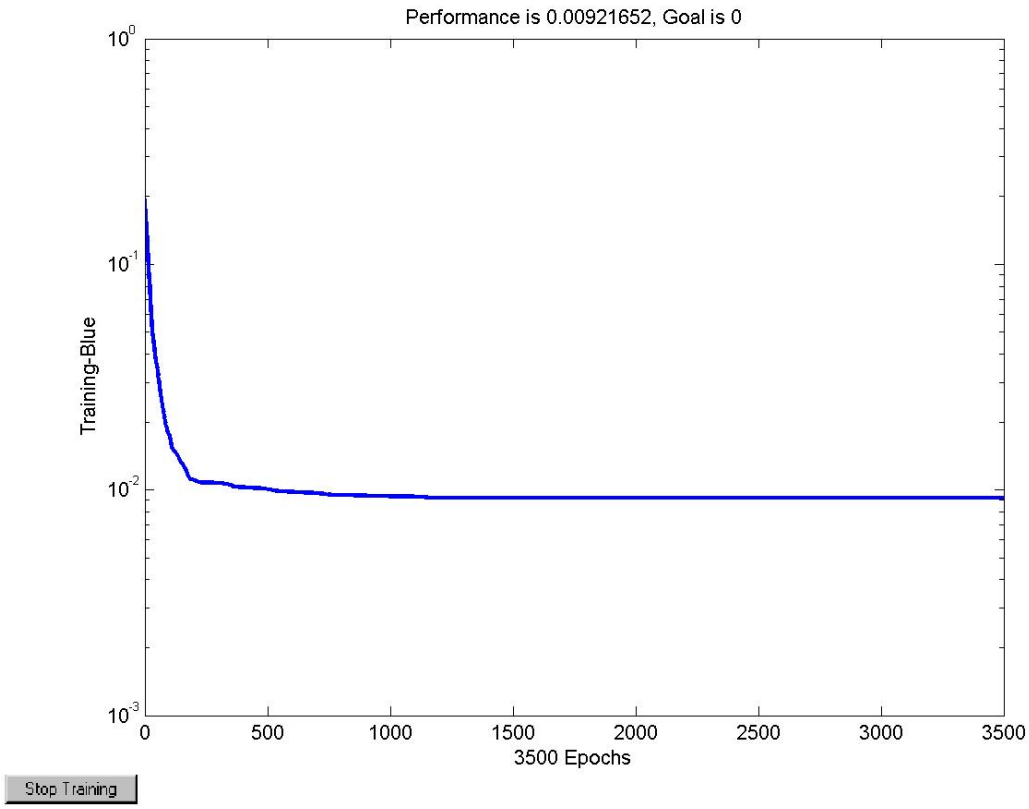
Πείραμα 34

Συνάρτηση εκπαίδευσης	traincgb
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1785
Λάθη μετά την εκπαίδευση του Data1	619
Λάθη μετά την εκπαίδευση του Data2	395
Λάθη μετά την εκπαίδευση του Data3	2



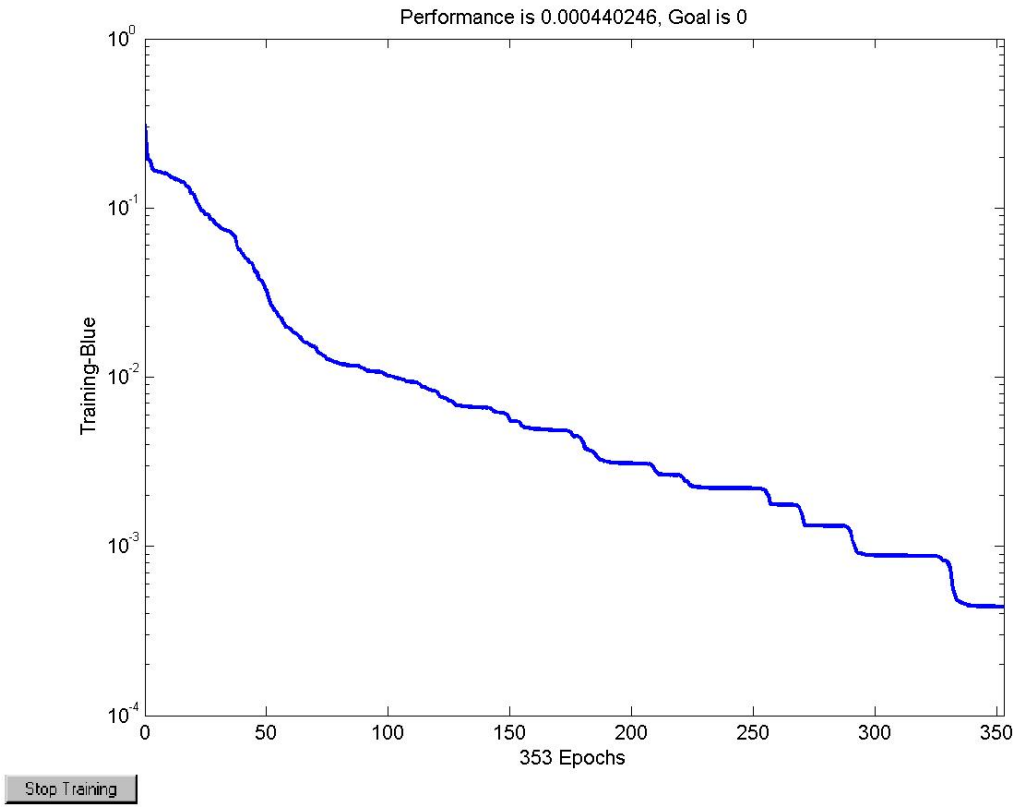
Πείραμα 37

Συνάρτηση εκπαίδευσης	traincgf
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	492
Λάθη μετά την εκπαίδευση του Data1	405
Λάθη μετά την εκπαίδευση του Data2	120
Λάθη μετά την εκπαίδευση του Data3	21



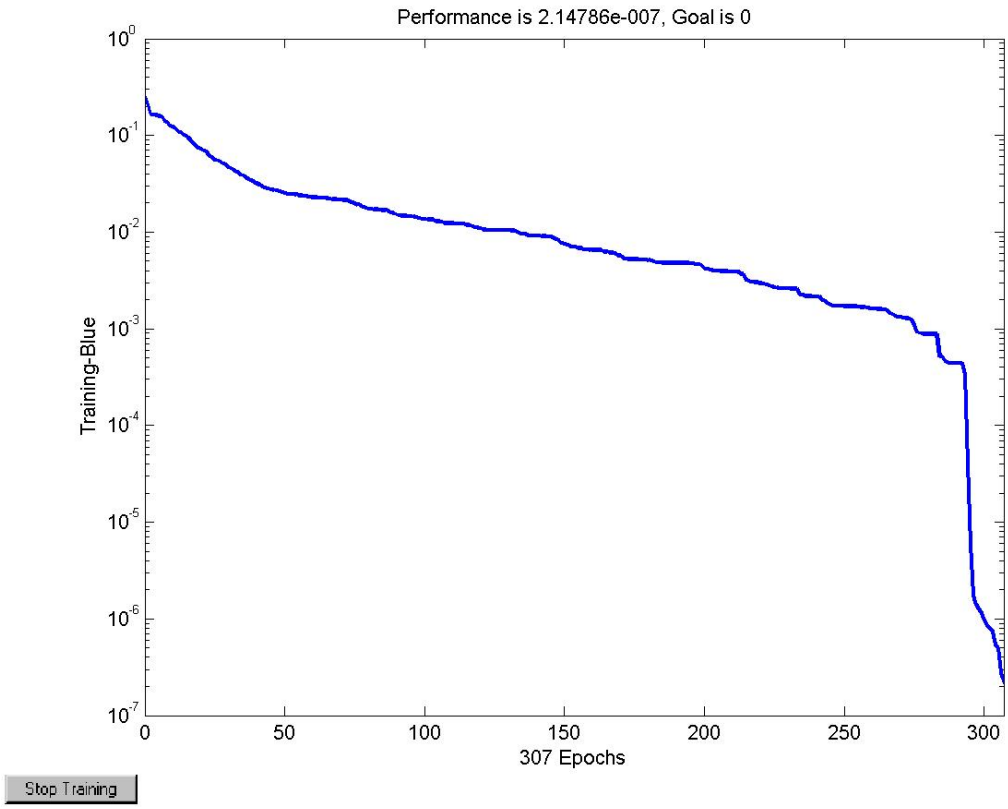
Πείραμα 40

Συνάρτηση εκπαίδευσης	trainscg
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1115
Λάθη μετά την εκπαίδευση του Data1	305
Λάθη μετά την εκπαίδευση του Data2	85
Λάθη μετά την εκπαίδευση του Data3	1



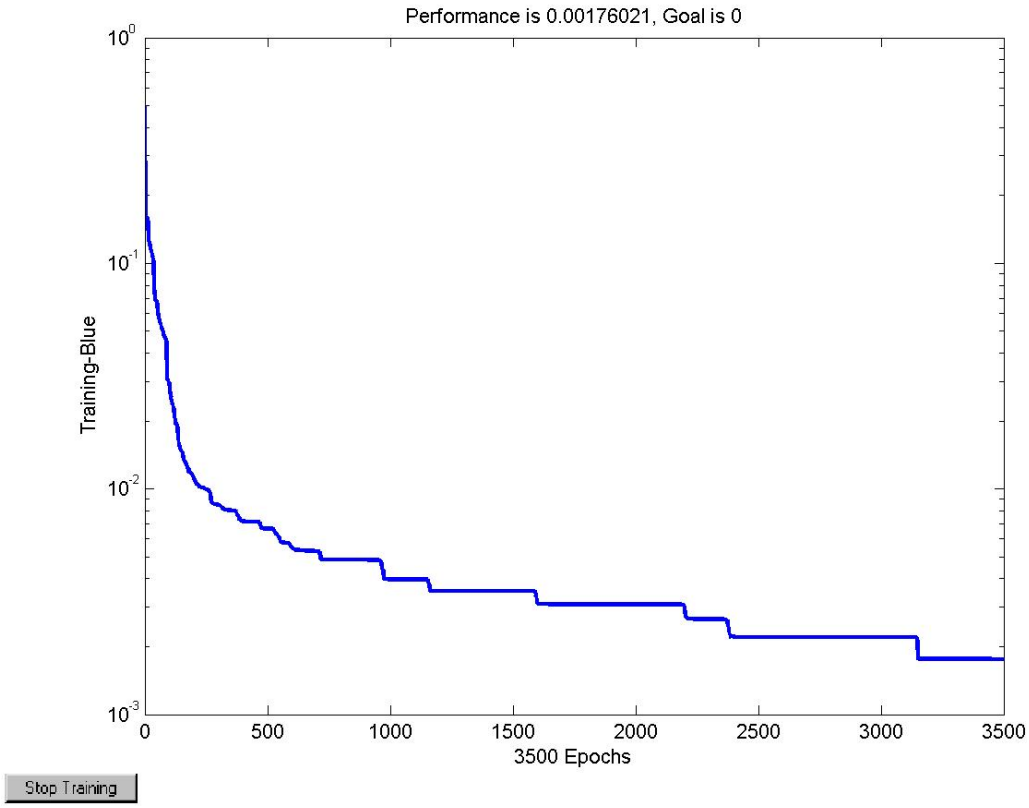
Πείραμα 43

Συνάρτηση εκπαίδευσης	traincgp
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	889
Λάθη μετά την εκπαίδευση του Data1	415
Λάθη μετά την εκπαίδευση του Data2	83
Λάθη μετά την εκπαίδευση του Data3	0



Πείραμα 46

Συνάρτηση εκπαίδευσης	trainoss
Αριθμός νευρώνων στο 1° κρυμμένο επίπεδο (layer1)	16
Αριθμός νευρώνων στο 2° κρυμμένο επίπεδο (layer2)	6
Εποχές εκπαίδευσης (epochs)	3500
Dataset που εκπαιδεύει το NN Train dataset	Data3
Λάθη πριν την εκπαίδευση του Train dataset	1543
Λάθη μετά την εκπαίδευση του Data1	285
Λάθη μετά την εκπαίδευση του Data2	58
Λάθη μετά την εκπαίδευση του Data3	4



Παραθέτουμε έναν πίνακα που συνοψίζει τις πειραματικές μετρήσεις

Πίνακας 6.1 Πειραματικές μετρήσεις

Αριθμός πειράματος	Συνάρτηση εκπαίδευσης	Αριθμός νευρώνων 1ου επιπέδου	Αριθμός νευρώνων 2ου επιπέδου	Dataset που χρησιμοποιήθηκε ως Trainset	Εποχές εκπαίδευσης
1	Traingda	8	0	Data1	3000
2	Traingda	8	0	Data2	3000
3	Traingda	6	0	Data3	3000
4	Traingda	8	0	Data3	3000
5	Traingda	10	0	Data3	3000
6	Traingda	12	0	Data3	3000
7	Traingda	16	0	Data3	3000
8	Traingda	18	0	Data3	3000
9	Traingda	16	4	Data3	3500
10	Traingda	16	5	Data3	3500
11	Traingda	16	6	Data3	3500
12	Traingda	16	7	Data3	3500
13	Traingda	16	8	Data3	3500
14	Traingda	16	9	Data3	3500
15	Traingda	16	10	Data3	3500
16	Traingda	16	11	Data3	3500
17	Traingda	16	12	Data3	3500
18	Traingd	16	6	Data1	15000
19	Traingd	16	6	Data2	15000
20	Traingd	16	6	Data3	15000
21	Traingd	16	12	Data3	30000
22	Traingdm	16	6	Data1	30000
23	Traingdm	16	6	Data2	30000
24	Traingdm	16	6	Data3	30000
25	Traingdm	16	6	Data1	50000
26	Traingdx	16	6	Data1	3500
27	Traingdx	16	6	Data2	3500
28	Traingdx	16	6	Data3	3500
29	Trainrp	16	6	Data1	3500
30	Trainrp	16	6	Data2	3500
31	Trainrp	16	6	Data3	3500
32	Traincgb	6	6	Data1	3500
33	Traincgb	16	6	Data2	3500
34	Traincgb	16	6	Data3	3500
35	Traincgf	16	6	Data1	3500
36	Traincgf	16	6	Data2	3500
37	Traincgf	16	6	Data3	3500
38	Trainscg	16	6	Data1	3500
39	Trainscg	16	6	Data2	3500
40	Trainscg	16	6	Data3	3500
41	Traincgp	16	6	Data1	3500
42	Traincgp	16	6	Data2	3500
43	Traincgp	16	6	Data3	3500
44	Trainoss	16	6	Data1	3500
45	Trainoss	16	6	Data2	3500
46	Trainoss	16	6	Data3	3500

Ο επόμενος πίνακας παρουσιάζει την λάθος ταξινόμηση των προτύπων πριν και μετά την εκπαίδευση του ΝΔ

Πίνακας 2: Αριθμητικά στοιχεία για το ΝΔ πριν και μετά την εκπαίδευση

Αριθμός πειράματος	Αριθμός προτύπων που ταξινομήθηκαν λάθος πριν την εκπαίδευσή	Αριθμός προτύπων που ταξινομήθηκαν λάθος μετά την εκπαίδευσή για το Data1	Αριθμός προτύπων που ταξινομήθηκαν λάθος μετά την εκπαίδευσή για το Data2	Αριθμός προτύπων που ταξινομήθηκαν λάθος μετά την εκπαίδευσή για το Data3
1	1163	346	111	16
2	1483	8	103	842
3	1705	1799	0	1800
4	1566	333	73	8
5	1010	308	95	21
6	1237	307	76	11
7	1760	265	104	8
8	1635	298	46	9
9	468	306	97	6
10	1063	290	34	6
11	549	264	44	0
12	764	294	54	7
13	523	350	57	1
14	1781	294	81	6
15	949	349	48	1
16	1330	326	127	3
17	1330	316	173	5
18	550	272	677	750
19	677	1750	2	1758
20	889	501	435	444
21	1668	457	398	330
22	566	204	694	907
23	1687	1770	0	1782
24	889	496	329	349
25	760	82	250	905
26	959	7	55	803
27	1373	1799	0	1801
28	568	294	42	6
29	1737	2	31	795
30	502	1802	0	1802
31	808	272	123	2
32	594	2	346	1068
33	562	1802	0	1802
34	1785	619	395	2
35	1435	3	187	835
36	609	1799	0	1801
37	492	405	120	21
38	1037	9	46	846
39	1772	1799	0	1801
40	1115	305	85	1
41	774	4	91	701
42	677	1801	0	1802
43	889	415	83	0
44	609	3	78	811
45	913	1799	0	1801
46	1543	285	58	4

6.3 *Σχολιασμός και συμπεράσματα*

Στα πρώτα 8 πειράματα έχοντας προαποφασίσει σαν συνάρτηση εκπαίδευσης την traingda, προσπαθούμε να βρούμε τον καταλληλότερο αριθμό νευρώνων στο πρώτο επίπεδο (ο βέλτιστος αριθμός νευρώνων καταλήγουμε ότι είναι 16) καθώς και έναν προσεγγιστικά ορθό αριθμό εποχών εκπαίδευσης (κατά μέσο όρο 3000) αναφορικά με μια συνάρτηση με χαμηλή υπολογιστική πολυπλοκότητα. Επίσης μελετάται η συμπεριφορά και οι ιδιότητες των τριών συνόλων δεδομένων όταν χρησιμοποιούνται ως σύνολα εκπαίδευσης και ως σύνολα ελέγχου (trainset – testset). Εκτός ενός τετριμμένου ελέγχου της γενικευτικής ικανότητας του υλοποιηθέντος δικτύου, εξετάζεται και το ζήτημα του κατά πόσο τα σύνολα δεδομένων που κατασκευάστηκαν περιγράφουν επαρκώς τον χώρο του προβλήματος (δηλαδή του κατά πόσο τα πρότυπα των συνόλων δεδομένων είναι αντιπροσωπευτικά του προβλήματος που καλούμαστε να επιλύσουμε). Παρατηρούμε, λοιπόν, ότι το Data3 έχει την πιο ομαλή / στατιστικά ευσταθή συμπεριφορά και τον μικρότερο αριθμό σφαλμάτων για τα τρία σύνολα δεδομένων. Από αυτήν τη σκοπιά, το νευρωνικό δίκτυο καταφέρνει μόνο όταν χρησιμοποιεί σαν σύνολο εκπαίδευσης και ελέγχου το Data2, να κατηγοριοποιεί σωστά όλα τα πρότυπα εισόδου με ένα κρυμμένο επίπεδο νευρώνων. Αυτό συμβαίνει λόγω του τρόπου κατασκευής των αρνητικών προτύπων εισόδου που είναι μια τυχαία ανάσυρση ακολουθιών από όλο το γονιδίωμα. Δηλαδή, οι ακολουθίες με τις οποίες δημιουργούμε το σύνολο δεδομένων Data2, εμπεριέχουν βιολογικές πληροφορίες που είναι πιο αξιόπιστες, σε σχέση με αυτές των άλλων συνόλων δεδομένων. Συνεπώς, το νευρωνικό δίκτυο έχει τη δυνατότητα να εξάγει αρτιότερα συμπεράσματα ως προς τις διαφορές που έχουν οι ακολουθίες υποκινητών με αυτές που βρίσκονται απλά στο γονιδίωμα του υπό εξέταση οργανισμού. Το Data1 βρίσκεται σε ένα μέσο επίπεδο από πλευράς αποτελεσμάτων αφού εμφανίζει λιγότερες αστοχίες ταξινόμησης προτύπων σε σχέση με το Data2. Στα πειράματα 7Γ-15Γ γίνονται δοκιμές με το Data3 για να βρεθεί το βέλτιστο πλήθος νευρώνων στο δεύτερο κρυμμένο επίπεδο. Ο βέλτιστος αριθμός κυρίως σύμφωνα με το πείραμα 9Γ είναι 6 νευρώνες. Οι εποχές εκπαίδευσης αυξάνονται στις 3500 κάτι που θεωρείται αναμενόμενο, αφού έχουμε συνολικά περισσότερους νευρώνες και άρα περισσότερες παραμέτρους που χρειάζεται να ρυθμιστούν. Σημειώνουμε το γεγονός ότι στο σύνολο των πειραμάτων υπήρξαν περιπτώσεις, που δεν έγινε χρήση του προεπιλεγμένου μέγιστου αριθμού εποχών εκπαίδευσης. Αυτό συνέβαινε γιατί κατά την εκπαίδευση ο αλγόριθμος ικανοποιούσε προηγουμένως κάποιο άλλο κριτήριο σύγκλισης, που στην πλειοψηφία των περιπτώσεων ήταν η ελάχιστη τιμή της κλίσης της μερικής παραγώγου του σφάλματος. Συνεπώς, οι πίνακες των πειραμάτων σε κάποιες περιπτώσεις είναι ανακριβείς όσον αφορά τον χρησιμοποιούμενο αριθμό των εποχών, ο οποίος απεικονίζεται στις γραφικές παραστάσεις.

Τα πειράματα 15Γ-17Δ πραγματοποιούνται με δύο συναρτήσεις αυξημένης πολυπλοκότητας, τις traingd και traingdm. Το ζητούμενο σε αυτή τη σειρά πειραμάτων ήταν ο καθορισμός ενός βέλτιστου αριθμού εποχών εκπαίδευσης. Όπως παρατηρείται σε αμφότερες περιπτώσεις υπάρχουν περιθώρια βελτίωσης των αποτελεσμάτων καθώς εξελίσσεται η φάση της εκπαίδευσης. Αυτό διαπιστώνεται από την μορφή των καμπυλών των γραφικών παραστάσεων που ακολουθούσαν φθίνουσα πορεία καθώς αυξάνονταν ο αριθμός των εποχών, χωρίς να μειώνεται λογαριθμικά η κλίση τους με την πάροδο των εποχών. Έγιναν δυο πειράματα προς αυτή την κατεύθυνση, το πρώτο με την traingd (πείραμα 15Γ) διπλασιάζοντας των αριθμό των εποχών από 15000 σε 30000, όπου μάλιστα είχαμε 12 νευρώνες στο 2 επίπεδο, δηλαδή όχι την βέλτιστη λύση, και το δεύτερο με την traingdm (πείραμα 17Δ)

ανεβάζοντας το πλήθος των εποχών από τις 30000 στις 50000, με αποτέλεσμα να επιβεβαιωθούν οι ενδείξεις για βελτιωμένα αποτελέσματα με περισσότερη εκπαίδευση. Αξιοσημείωτο είναι το γεγονός ότι σε αυτήν τη σειρά πειραμάτων οι γραφικές παραστάσεις είχαν την πιο ομαλή συμπεριφορά. Περαιτέρω διερεύνηση δεν πραγματοποιήθηκε επειδή με πιο εύρωστες υπολογιστικά συναρτήσεις εκπαίδευσης (π.χ. traingda) είχανε ήδη παρατηρηθεί καλά αποτελέσματα.

Τέλος, τα πειράματα 18Α-24Γ έγιναν με συναρτήσεις που περιλαμβάνουν μικρό αριθμό πράξεων (traingdx, trainrp, traincgb, traincgf, trainscg, traincgp, trainoss) και επομένως οι χρονικές απαιτήσεις τους είναι περιορισμένες. Για κάθε συνάρτηση χωριστά πραγματοποιήθηκαν τρία πειράματα. Τα τρία σύνολα δεδομένων εναλλάσσονταν σειριακά σαν σύνολα εκπαίδευσης. Οι συναρτήσεις αυτές παρουσιάζονται συγκεντρωτικά λόγω ομοιοτήτων στην απόδοσή τους. Οι βασικές τους ομοιότητες ήταν δύο: τα πανομοιότυπα αποτελέσματα αναφορικά με το Data2 και το γεγονός του ότι όταν γινόταν εκπαίδευση με τα Data1, Data3 οι λανθασμένες κατηγοριοποιήσεις κυμαίνονταν σε μια κοινή περιοχή αποτελεσμάτων της τάξης 2-9 και 0-21 λανθασμένων αποτελεσμάτων αντίστοιχα. Ποσοτικά αυτοί οι αριθμοί είναι πολύ κοντά, αν τους συγκρίνουμε με το πλήθος των προτύπων (συνολικά 2273 πρότυπα). Τα χειρότερα αποτελέσματα ελήφθησαν με την traincgf. Επισημαίνουμε ότι τα πειράματα (18Α-24Γ) επιβεβαίωσαν τα πρότερα συμπεράσματα για τις σχέσεις και τις επιδόσεις μεταξύ των τριών συνόλων δεδομένων.

Συνοψίζοντας τα συμπεράσματα με τον αλγόριθμο ανάστροφης διάδοσης σφάλματος, καταλήγουμε ότι τα αποτελέσματα έφτασαν σε επίπεδο επιτυχούς πρόβλεψης 100%, όταν τα σύνολα δεδομένων που χρησιμοποιήθηκαν είχαν ρόλο συνόλου εκπαίδευσης και ελέγχου ταυτόχρονα. Για την γενίκευση μεταξύ των συνόλων δεδομένων είναι δύσκολο να βγουν γενικά συμπεράσματα, γιατί, λόγω κατασκευής και προφανώς βιολογικών περιορισμών οι υποκινητές είναι και στα τρία σύνολα δεδομένων οι ίδιοι. Παρ' όλα αυτά, είναι ενθαρρυντικό ότι στο πείραμα 8Γ όπου εκπαιδεύσαμε το ΝΔ με το Data3 για το μεν Data2 έχουμε ένα ποσοστό αναγνώρισης των προτύπων της τάξης του 99% για το δε Data1 ένα ποσοστό της τάξης του 88%.

Κατά τη γνώμη μας υπάρχουν μια σειρά από δυνατές βελτιώσεις / τροποποιήσεις που θα μπορούσαν να γίνουν στην εφαρμογή μας, κυρίως σε επίπεδο υλοποίησης με την εισαγωγή ενός πιο πολύπλοκου υβριδικού μοντέλου που να έχει τη δυνατότητα συνδυασμού των αποτελεσμάτων του Backpropagation αλγορίθμου με την εισαγωγή ήδη υπαρχόντων κανόνων που περιγράφουν το πρόβλημα των υποκινητών[4]. Ένα από τα σημαντικότερα σημεία βελτίωσης του συστήματος θα ήταν η αναλυτική επεξεργασία των αποστάσεων μεταξύ των υποακολουθιών που μας ενδιαφέρουν (δηλαδή η απόσταση μεταξύ των υποακολουθιών -35 και -10). Βασική έννοια για να προσεγγίσουμε με αυτόν τον τρόπο το πρόβλημά μας είναι η απόσταση Levenshtein (γνωστή και ως edit distance)[5]. Επίσης, ιδιαίτερη αξία θα είχε στο ΝΔ να μπορούν να εισαχθούν ακολουθίες που να μην είναι ισομήκεις, αλλά το μήκος τους να είναι κυμαινόμενο. Επιπρόσθετα, με βάση τα πιο πρόσφατα άρθρα, για τη δημιουργία συνόλων δεδομένων με «ισχυρούς» υποκινητές, θα ήταν σωστό να ληφθούν υπόψη οι υπομονάδες σ (σ-factors). Τέλος, έχει ιδιαίτερο ενδιαφέρον να εξεταστεί ποια είναι καλύτερη επιλογή για το πλήθος των νευρώνων των δύο κρυμμένων επιπέδων. Ειδικότερα, αν κάθε κρυμμένο επίπεδο αντιπροσωπεύει και μια συγκεκριμένη κυρτή / σύνθετη περιοχή απόφασης (στην περίπτωσή μας τις περιοχές -35 και -10), θα μπορούσαμε να ρυθμίσουμε τις παραμέτρους των κρυμμένων επιπέδων βάσει αυτής της πληροφορίας και στη συνέχεια να ληφθεί ένα συνδυαστικό αποτέλεσμα από την έξοδο του νευρωνικού δικτύου.

Η ενασχόλησή μας για μεγάλο χρονικό διάστημα με τη διαμόρφωση των τελικών συνόλων δεδομένων που χρησιμοποιήσαμε δεν μας επέτρεψε να εξετάσουμε αναλυτικά τη συμπεριφορά πιο εξελιγμένων μοντέλων μηχανικής μάθησης. Παρ' όλα αυτά επιδιώξαμε να δούμε σε ένα πρωτόλειο επίπεδο τη συμπεριφορά στο συγκεκριμένο πρόβλημα του υβριδικού νευρο-ασαφούς μοντέλου SuPFuNIS[6]. Η προσπάθειά μας δεν είχε ικανοποιητικά αποτελέσματα, θεωρούμε όμως ότι είναι απαραίτητη μια περαιτέρω ενασχόληση με το συγκεκριμένο μοντέλο, πάνω στο συγκεκριμένο πρόβλημα, για να εξάγουμε ασφαλή συμπεράσματα σχετικά με τη χρηστικότητα ή μη του συγκεκριμένου νευρο-ασαφούς ταξινομητή για βιολογικά προβλήματα αντίστοιχα με αυτό που επιχειρήσαμε και εμείς να αντιμετωπίσουμε. Ιδιαίτερο ενδιαφέρον θα είχε τέλος να συγκριθούν οι διάφορες μέθοδοι μηχανικής μάθησης που παρέχονται μέσα από το εργαλείο WEKA[7], το οποίο προσφέρει μια μεγάλη ποικιλία μεθόδων και αλγόριθμων.

---

[1] Q.Ma, J.Wang, D.Shasha, C.Wu – DNA Sequence Classification via an Expectation Maximization Algorithm and Neural Networks: A Case Study – IEEE TRANSACTION ON SYSTEMS, MAN AND CYBERNETICS – PART C: APPLICATIONS AND REVIEWS – Vol.31, No.4, November 2001
Martin G. Reese – Computational prediction of gene structure and regulation in the genome of Drosophila Melanogaster – 2000
R.Ranawana – MultiNNProm: A Multi-Classifier System for Gene Finding

[2] Π.χ. στο: V.Cotik, R.Zalic, I.Zwir - A hybrid promoter analysis methodology for prokaryotic genomes – Fuzzy sets and systems, Elsevier

[3] http://www.ics.uci.edu/~mlearn/MLrepository.htm

[4] J.Ortega – On the Informativeness of the DNA Promoter Sequences Domain Theory – Journal of Artificial Intelligence Research 2 (1995) 361 – 367

[5] Για περισσότερες πληροφορίες όσον αφορά την απόσταση μετασχηματισμού: http://www.lisha.ufsc.br/~guto

[6] Sandeep Paul and Satish Kumar - Subsethood-Product Fuzzy Neural Inference System (SuPFuNIS) – IEEE Transactions On Neural Networks,Vol.13, No.3, May 2002
Εφαρμογή του μοντέλου SuPFuNIS: Κώστας Α. Τσιώλης & Χρήστος Ι. Φερλές - Ασύρματη Κατανεμημένη Υλοποίηση Νευρο-Ασαφούς Συστήματος Ταξινόμησης, 2003

[7] http://www.cs.waikato.ac.nz/ml/weka

ΕΠΙΛΟΓΟΣ

Η παρούσα διπλωματική εργασία είχε παράλληλα δύο στόχους: ο πρώτος ήταν να συστηματοποιήσει όσο το δυνατόν καλύτερα τις τάσεις που υπάρχουν σήμερα στο χώρο της βιοπληροφορικής, ιδιαίτερα δε αυτές που ασχολούνται με διάφορες μορφές μηχανικής μάθησης. Αυτό δεν θα μπορούσε να γίνει αν προηγουμένως δεν γινόταν μια συνοπτική παρουσίαση της σύγχρονης βιολογίας του κυττάρου και της γενετικής των οργανισμών. Ο δεύτερος στόχος ήταν να προσπαθήσουμε να δημιουργήσουμε μια εφαρμογή που να βασίζεται σε τεχνικές μηχανικής μάθησης και να δέχεται βιολογικές ακολουθίες σαν δεδομένα. Παρά το ότι η εμπειρία που είχαμε από τις προηγούμενα χρόνια των σπουδών μας δεν ήταν τέτοια που να μας επέτρεπε να είμαστε αισιόδοξοι για το τελικό αποτέλεσμα, θεωρούμε μετά την ολοκλήρωση της εργασίας μας ότι αυτό μπορεί να αποτελέσει τη βάση για μια ακόμα πιο πετυχημένη προσέγγιση του αντικειμένου της βιοπληροφορικής. Είναι γεγονός ότι η ταχύτατη ανάπτυξη του αντικειμένου δυσχεραίνει ακόμα περισσότερο την παρακολούθηση των εξελίξεων σε αυτό, δεδομένου όμως ότι τόσο η σχέση των περισσότερων συναδέλφων με την υπολογιστική επιστήμη, όσο και η όχι εκ του μηδενός απορρόφηση γνώσεων πάνω σε βιολογικά προβλήματα, μας δίνει το θάρρος να πούμε ότι και σε αυτόν τον τομέα υπάρχουν πολλές πιθανότητες επιτυχιών για όλους όσους διακρίνονται από όρεξη για δημιουργική έρευνα με κίνητρα ανθρωπιστικής χροιάς, κίνητρα τα οποία θα έπρεπε να ωθούν την έρευνα σε κάθε έκφανσή της.

Παράρτημα 1

Παραδείγματα μορφής FASTA και EMBL

Παράρτημα 1

Παραδείγματα μορφής FASTA και EMBL

Δύο από τις πιο διαδεδομένες μορφές με τις οποίες είναι αποθηκευμένα βιολογικά δεδομένα στις βάσεις δεδομένων είναι οι μορφές FASTA και EMBL. Και στις δυο περιπτώσεις, με μια σειρά καταχωρήσεων, η οποιαδήποτε ακολουθία ενός συγκεκριμένου οργανισμού συνοδεύεται από την πλήρη ονομασία του οργανισμού, αν και στην EMBL-μορφή υπάρχουν περισσότερες πληροφορίες όπως ο αριθμός καταχώρησης με τον οποίο μπορείς άμεσα να ανακτήσεις την ακολουθία αυτή κάποια αριθμητικά στοιχεία όπως το μήκος σε bp της ανακτηθείσας ακολουθίας.

Ειδικότερα, ένα αρχείο ακολουθιών FASTA-μορφής μπορεί να περιέχει περισσότερες της μιας ακολουθίας. Μια ακολουθία σε FASTA-μορφή ξεκινάει με μια περιγραφή για την ακολουθία στην πρώτη γραμμή και συνεχίζει με τη συγκεκριμένη ακολουθία. Η γραμμή περιγραφής πρέπει να ξεκινάει με το σύμβολο (“>”). Στο σχήμα Π 1.1 εμφανίζεται μια μορφή ακολουθίας σε FASTA-μορφή.

```
>U03518 Aspergillus awamori internal transcribed spacer 1 (ITS1)
AACCTGCGGAAGGATCATTACCGAGTGCGGGTCCTTTGGGCCCAACCTCCCATCCGTGTCTATTGTACCC
TGTTGCTTCGGCGGGCCCGCCGCTTGTCGGCCGCCGGGGGGGCGCCTCTGCCCCCCGGGCCCGTGCCCGC
CGGAGACCCCAACACGAACACTGTCTGAAAGCGTGCAGTCTGAGTTGATTGAATGCAATCAGTTAAAACT
TTCAACAATGGATCTCTTGGTTCCGGC
```

Σχήμα Π 1.1: Παράδειγμα ανάκτησης ακολουθίας σε FASTA-μορφή

Όπως και τα αρχεία ακολουθιών FASTA-μορφής, έτσι και τα αρχεία EMBL-μορφής μπορούν να περιέχουν περισσότερες από μια ακολουθίες. Κάθε είσοδος νέας ακολουθίας ξεκινάει με την αναγνωριστική γραμμή (identifier line – “ID”), η οποία ακολουθείται από άλλες γραμμές που περιγράφουν την συγκεκριμένη ακολουθία (annotation lines). Η έναρξη της ακολουθίας διακρίνεται από την γραμμή που έχει στην αρχή της το αναγνωριστικό (“SQ”), ενώ το τέλος της ακολουθίας από την γραμμή που στην αρχή της έχει το αναγνωριστικό (“//”). Ένα παράδειγμα αρχείου EMBL-μορφής είναι αυτό που εμφανίζεται στο σχήμα Π 1.2

```
ID   AA03518    standard; DNA; FUN; 237 BP.
XX
AC   U03518;
XX
DE   Aspergillus awamori internal transcribed spacer 1 (ITS1) and 18S
DE   rRNA and 5.8S rRNA genes, partial sequence.
XX
SQ   Sequence 237 BP; 41 A; 77 C; 67 G; 52 T; 0 other;
     aacctgcgga aggatcatta ccgagtgcgg gtcctttggg cccaacctcc catccgtgtc
60
     tattgtaccc tgttgcttcg gcgggcccgc cgcttgtcgg ccgccggggg ggcgcctctg
120
     ccccccgggc ccgtgcccgc cggagacccc aacacgaaca ctgtctgaaa gcgtgcagtc
180
     tgagttgatt gaatgcaatc agttaaaact ttcaacaatg gatctcttgg ttccggc
237
//
```

Σχήμα Π 1.2: Παράδειγμα ανάκτησης ακολουθίας σε EMBL-μορφή

Να σημειώσουμε ότι μια υπάρχουν μια σειρά άλλων μορφών κωδικοποίησης των ακολουθιών πέραν αυτών που παρουσιάζουμε παραπάνω. Επιλέξαμε να παρουσιάσουμε τις συγκεκριμένες, για 2 κυρίως λόγους: πρώτον γιατί είναι αυτές που χρησιμοποιούνται παραπάνω σε σχέση με τις υπόλοιπες και δεύτερον γιατί είναι αυτές που χρησιμοποιήσαμε και εμείς για να ανακτήσουμε τις απαραίτητες ακολουθίες ούτως ώστε να διαμορφώσουμε τα τελικά σύνολα δεδομένων. Άλλες μορφές που χρησιμοποιούνται είναι οι μορφές GCG, GenBANK, IG και η Plain Sequence.

Παράρτημα 2

Παράδειγμα χρήσης του Bioinformatics Toolbox της MATLAB 7 για στατιστική επεξεργασία ακολουθίας

Παράρτημα 2

Παράδειγμα χρήσης για του Bioinformatics Toolbox της MATLAB 7 για στατιστική επεξεργασία ακολουθίας

Το Bioinformatics Toolbox που υπάρχει στην έκδοση 7 της Matlab έχει μια σειρά από δυνατότητες που μέχρι σήμερα δεν ήταν συγκεντρωμένες σε ένα εργαλείο, με αποτέλεσμα να είναι απαραίτητη η ανάπτυξη κώδικα σε κάποια γλώσσα για την στατιστική επεξεργασία ακολουθιών και την μετατροπή τους σε μορφή που να είναι δυνατή η αναγνώρισή τους από νευρωνικά δίκτυα ή από κάποια άλλη μορφή μηχανικής μάθησης. Στο παρακάτω παράδειγμα έχουμε το γονιδίωμα με το όνομα mitochondria, το οποίο είναι σε EMBL-μορφή. Με την εντολή:

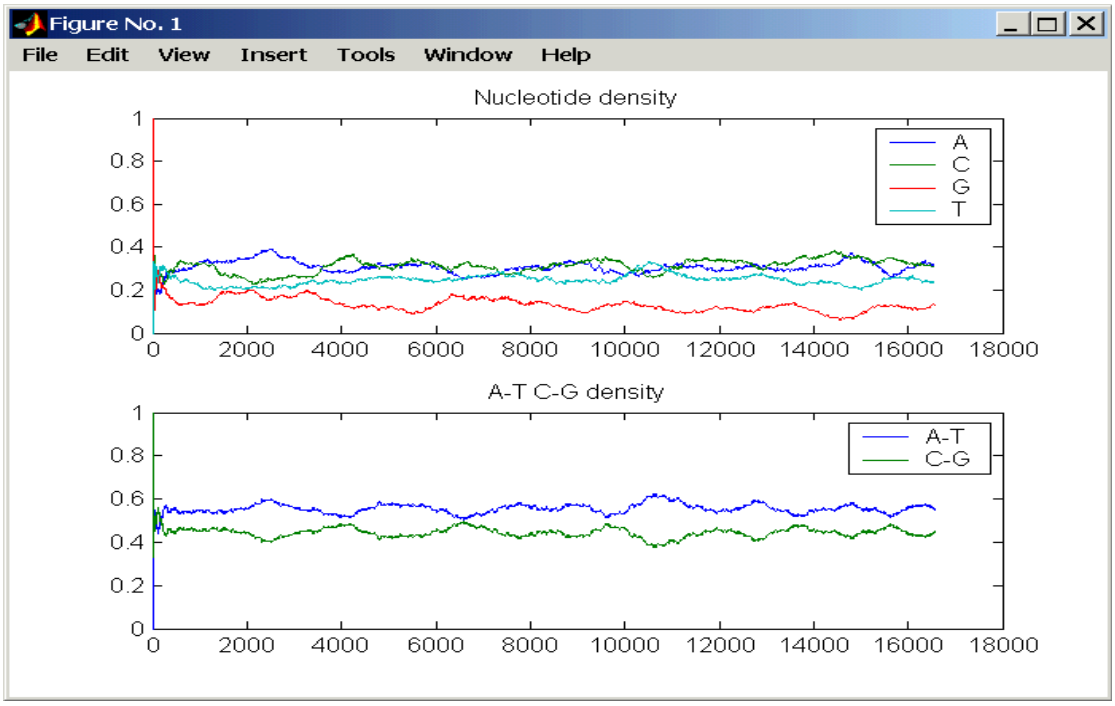
mitochondria = getembl('NC_001807','SequenceOnly',true)

αποθηκεύουμε την ακολουθία με αριθμό πρόσβασης NC_0018796 σε πίνακα με το όνομα mitochondria και μπορούμε στη συνέχεια να τον επεξεργαστούμε. Ο αριθμός πρόσβασης NC_001807 αντιστοιχεί στο ανθρώπινο γονιδίωμα των μιτοχονδρίων.

Στη συνέχεια με την εντολή

ntdensity(mitochondria)

παρατηρούμε ποια είναι η νουκλεοτιδική πυκνότητα στην ακολουθία που εξετάζουμε, δηλαδή στην συγκεκριμένη περίπτωση του ανθρώπινου γονιδιώματος των μιτοχονδρίων. Αποδεικνύεται από το γράφημα που εμφανίζεται ότι το γονιδίωμα έιναι AT-rich.



Σχήμα Π 2.1: Η εμφάνιση της νουκλεοτιδικής πυκνότητας για το υπο εξέταση γονιδίωμα.

Δίνεται επίσης η δυνατότητα μέσω του Bioinformatics Toolbox να υπολογιστεί το πλήθος των βάσεων που υπάρχουν στο γονιδίωμα μέσω της εντολής

basecount(mitochondria)

η οποία εμφανίζει το πλήθος εμφανίσεων για το υπο εξέταση γονιδίωμα.

Να σημειώσουμε ότι οι δυνατότητες που δίνει το Bioinformatics Toolbox είναι πολύ περισσότερες από αυτές που παρουσιάζουμε σε αυτό το παράρτημα, η σχετικά πρόσφατη όμως νέα έκδοση της MATLAB δεν μας επέτρεψε να εκμεταλλευτούμε και να παρουσιάσουμε εκτενέστερα τις δυνατότητές της. Για μια αναλυτική παρουσίαση των δυνατοτήτων του Bioinformatics Toolbox υπάρχει στον δικτυακό τόπο http://www.mathworks.com

Παράρτημα 3

Σύνολα Δεδομένων

Παράρτημα 3

Σύνολα δεδομένων

Το αρχικό σύνολο δεδομένων που χρησιμοποιήσαμε σε αρχικό στάδιο ανακτήθηκε από το διαδίκτυο και από τη βάση δεδομένων που βρίσκεται στο δικτυακό τόπο:

http://www.ics.uci.edu/~mlearn/MLRepository.htm

Το σύνολο δεδομένων που ανακτήθηκε από το παραπάνω δικτυακό τόπο είχε συνολικά 106 πρότυπα εισόδου, με 57 bp μήκος το καθένα. Τα μισά πρότυπα (53) ήταν θετικά πρότυπα εισόδου (δηλαδή υποκινητές) και τα υπόλοιπα (53) αρνητικά (δηλαδή μη-υποκινητές).

Για να δημιουργήσουμε ένα πιο αντιπροσωπευτικό σύνολο δεδομένων με βάση τις πιο πρόσφατες έρευνες που έχουν γίνει αλλά και βάσει των συνόλων δεδομένων που χρησιμοποιήθηκαν σε πιο πρόσφατες εργασίες που συνδύαζαν το πρόβλημα των υποκινητών στην Escherichia Coli με τη μηχανική μάθηση, ανακτήθηκαν τα θετικά πρότυπα εισόδου από τον δικτυακό τόπο:

http://bioinfo.md.huji.ac.il/marg/promec/

Στις ακολουθίες παριστάνεται με μικρό γράμμα το σημείο εκκίνησης της μεταγραφής (TSS). Όπου δεν έχει βρεθεί ακόμα η ακριβής θέση του TSS υπάρχουν περισσότερες από μια θέσεις με μικρά γράμματα. Οι ακολουθίες είναι ευθυγραμμισμένες. Τα αρνητικά πρότυπα εισόδου όπως αναφέραμε στο κεφάλαιο 6 δημιουργήθηκαν με τρεις διαφορετικούς τρόπους. Ο πρώτος τρόπος απαιτούσε μια συλλογή όλων των γονιδίων της E.Coli. Η συλλογή αυτή υπάρχει στον τόπο:

ftp://ftp.ebi.ac.uk/pub/databases/ecdc/

και το όνομα του αρχείου είναι gen.zip. Ο δεύτερος τρόπος απαιτούσε όλο το γονιδίωμα της E.Coli, που μπορεί να βρεθεί είτε στην παραπάνω ftp διεύθυνση είτε στη ftp διεύθυνση:

ftp://ftp.ncbi.nih.gov/genbank/genomes/Bacteria/Escherichia_coli_K12/

Στο παράρτημα 4 παρουσιάζεται ο κώδικας που χρησιμοποιήθηκε για τη μετατροπή των ακολουθιών αυτών στα τελικά σύνολα δεδομένων:

- ✓ Data1: το σύνολο δεδομένων με αρνητικά πρότυπα εισόδου τις ακολουθίες που προέρχονται από τα γονίδια της E.Coli, αφού

διαγράψουμε την αρχική περιοχή του γονιδίου όπου βρίσκονται οι υποκινητές.

- ✓ Data2: το σύνολο δεδομένων με αρνητικά πρότυπα εισόδου τις ακολουθίες που προέρχονται από ολόκληρο το γονιδίωμα της E.Coli, αφού εξασφαλίσουμε ότι δεν έχουμε ακολουθία υποκινητή.
- ✓ Data3: το σύνολο δεδομένων με αρνητικά πρότυπα εισόδου τις ακολουθίες που προέρχονται από τυχαίες ακολουθίες DNA, με πυκνότητα AT πάνω από 60%, ώστε να προσεγγίζουν τη νουκλεοτιδική πυκνότητα της κωδικοποιούμενης περιοχής ενός γονιδίου.

Παράρτημα 4

Πηγαίοι Κώδικες

Παράρτημα 4

Πηγαίοι κώδικες

Στο παράρτημα αυτό παρουσιάζουμε με σχόλια το μεγαλύτερο μέρος του κώδικα που χρησιμοποιήθηκε, είτε σε Matlab, είτε σε Java. Στο τέλος παρουσιάζουμε τους κανόνες που περάσαμε στη MATLAB και χρησιμοποιήθηκε στις δοκιμές που έγιναν πάνω στο νευρο-ασαφές μοντέλο SupFunis.

```
function matrix = gene_to_indicatorseq();
% Διαβάζει μια ακολουθία και τη μετατρέπει σε
%ακολουθία δυαδικών αριθμών με αναπαράσταση 4 bits
% για την κάθε βάση/γράμμη. Στο Input προστίθεται
%το όνομα του αρχείου που περιέχει τις ακολουθίες
% και στο Output το όνομα του αρχείου που θα
%εμφανιστεί το αποτέλεσμα

gene = textread('Input','%s','delimiter','');
N = length(gene);

for i = 1:N,
   temp = [];
   for j = 1:length(gene{i})
         if gene{i}(j) == 'A'
            temp = [temp,bitget(8,4:-1:1)];
         elseif gene{i}(j) == 'C'
           temp = [temp,bitget(4,4:-1:1)];
         elseif gene{i}(j) == 'G'
            temp = [temp,bitget(2,4:-1:1)];
         elseif gene{i}(j) == 'T'
            temp = [temp,bitget(1,4:-1:1)];
         elseif gene{i}(j) == 'a'
            temp = [temp,bitget(8,4:-1:1)];
         elseif gene{i}(j) == 'c'
            temp = [temp,bitget(4,4:-1:1)];
         elseif gene{i}(j) == 'g'
            temp = [temp,bitget(2,4:-1:1)];
         elseif gene{i}(j) == 't'
            temp = [temp,bitget(1,4:-1:1)];
         end
         if j == length(gene{i})
         temp = [temp,2];
         end
   end
    value(i,:) = temp;

end
dlmwrite('Output.txt',value,' ');
```

```
import java.io.*;
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;   //διαβάζει ένα αρχείο με EMBL-μορφή και
                                   //το μετατρέπει σε μορφή απλής ακολουθίας
                                   // DNA σε άλλο αρχείο. Filename το όνομα
                                   //του αρχείου εισόδου και out το όνομα της
                                   // εξόδου (με κατάληξη .txt)

public class ReadEcoliGenes {

public static void main(String[] args)
        throws Exception
        {

BufferedReader br = new BufferedReader(
                              new FileReader("Filename.txt");
        SequenceIterator stream = SeqIOTools.readEmbl(br);


BufferedWriter out = new BufferedWriter(
                            new FileWriter("Out.txt"));


while (stream.hasNext()) {
                Sequence seq = stream.nextSequence();
                out.write(seq.seqString(),0,seq.length());
                out.newLine();

                 }

        br.close();
        out.close();

        }
}
```

```
function value = nonPromoterFromGenes(sourceName,destinName,length)
% Δημιουργεί αρνητικά πρότυπα εισόδου.
% Το sourceName δηλώνει το αρχείο από το
% οποίο ιδαβάζει την είσοδο και το
% destinName το αρχείο που θα αποθηκευτούν οι
% παραγόμενες ακολουθίες. Η παράμετρος length
% δηλώνει το μήκος της παραγόμενης ακολουθίας.
```

```
cellArray = textread(sourceName,'%s','bufsize',100000);
[m,n] = size(cellArray);
result = [];
for i = 1:m,
   array = cell2mat(cellArray(i));
   [dimx,dimy] = size(array);
   if(dimy>=200+length)
      index = round(rand*(dimy-length-200))+201;
      result = [result;array(1,index:(index+length-1))];
   end
end
dlmwrite(destinName,result,'');
```

```
function value = nonPromoterRandomATRich(fileName,number,length)
%Δημιουργούνται ακολουθίες μη-υποκινητών,
%οι οποίες έχουν κατά μέσο όρο 60%
%πυκνότητα σε A και T κατά μέσον όρο.
%Δηλώνονται το όνομα του παραγόμενου αρχείου (fileName)
%το πλήθος των παραγόμενων ακολουθιών (number)
%και το μήκος των παραγόμενων ακολουθιών(length)

pin = [];
for i = 1:number,
   sequence = [];
   for j = 1:length,
      if(rand<0.6)
         if(rand<0.5)
            sequence = [sequence,'A'];
         else
            sequence = [sequence,'T'];
         end
      else
         if(rand<0.5)
            sequence = [sequence,'C'];
         else
            sequence = [sequence,'G'];
         end
      end
   end
   pin = [pin;sequence];
end
dlmwrite(fileName,pin,'');
```

```
function value =
nonPromoterRandomSubsequence(filename,number,length,segments)
%Δημιουργούνται τυχαίες ακολουθίες DNA,
%οι οποίες είναι μη-υποκινητές και
%λαμβάνονται από το γονιδίωμα της E.Coli.
%Με το input.txt δηλώνεται το αρχείο το
%οποίο θα "διαβαστεί", το filename δείχνει
%την έξοδο, ενώ οι παράμετροι number,
%length, segments αντιπροσωπεύουν το
%πλήθος των παραγόμενων ακολουθιών,
%το μήκος τους και το πλήθος των
%υποακολουθιών από το άθροισμα των
%οποίων θα δημιουργηθούν οι τελικές ακολουθίες αντίστοιχα.
cellArray = textread('input.txt','%s','bufsize',6000000);
sequence = cell2mat(cellArray);
[dummy,n] = size(sequence);
pin = [];
step = floor(n/(segments+1));
sublength = floor(length/segments);
for i = 1:number,
   index = 1;
   result = [];
   for loop = 1:segments,
      subsequence = sequence(1,index:(index+step-1));
      point = round(rand*(sublength-1))+1;
      result = [result,subsequence(1,point:(point+sublength-1))];
      index = index+step;
   end
      subsequence = sequence(1,index:(index+step-1));
      point = round(rand*(sublength-1))+1;
      [dummy,m] = size(result);
      result = [result,subsequence(1,point:(point+length-m-1))];
   pin = [pin;result];
end
dlmwrite(output,pin,'');
```

```
function many = removeDuplicate(source,destination);
%Απομακρύνονται τυχόν διπλές ακολουθίες.
%Πρέπει να δηλωθεί σαν source το όνομα
%του αρχείου που ελέγχεται και τελικά
%με απομακρυσμένες τις ακολουθίες που
%έχουν εμφανισθεί παραπάνω από μια φορές
%εμφανίζεται το αποτέλεσμα στο αρχείο
%με όνομα destination
cellArray = textread(source,'%s');
pin = cell2mat(cellArray);
```

```
[m,dummy] = size(pin);
duplicate = zeros(m,1);
for i = 1:m,
   buffer = pin(i,:);
   for j = (i+1):m,
      duplicate(j) = duplicate(j)+strcmp(buffer,pin(j,:));
   end
end
many = 0;
final = [];
for i =1:m,
   if(duplicate(i)==0)
      many = many+1;
      final = [final;pin(i,:)];
   end
end
dlmwrite(destination,final,'');
```

```
%Για να δημιουργήσουμε το τελικό
%σύνολο δεδομένων πρέπει να το
%"ανακατέψουμε", έτσι ώστε να έχει
%τόσο θετικά, όσο και αρνητικά
%πρότυπα εισόδου με ανακατεμένη
%σειρά. Σαν είσοδος το Data(i) και
%η έξοδος εμφανίζεται στο αρχείο
%με όνομα Answer.txt
 pin = dlmread('Data(i).txt',' ');
 [r c] = size(pin);
 for i = 1:20000,
    m = floor(rand*(r-1.0))+1;
    n = floor(rand*(r-1.0))+1;
    swap = pin(m,:);
    pin(m,:) = pin(n,:);
    pin(n,:) = swap;
 end
 dlmwrite('Answer.txt',pin,' ');
```

```
function value = verificationATRich(filename)
%Έλεγχος στις ακολουθίες που
%υπάρχουν στο αρχείο με όνομα
%filename αν έχουμε πυκνότητα AT rich

cellArray = textread(filename,'%s');
pin = cell2mat(cellArray);
[m,n] = size(pin);
for i = 1:m,
   count = 0;
   for j = 1:n,
      if(pin(i,j)=='A'|pin(i,j)=='T')
         count = count+1;
      end
   end
   value(i) = count/n;
end
value = value';
```

```
function void = generate(n,q,p,fileid)
%Το παρόν πρόγραμμα, πέραν του συνόλου
%των κανόνων που περιέχει, και οι οποίοι
%ισχύουν για DNA ακολουθίες υποκινητών,
%δημιουργεί τα αρχεία, στα οποία στη
%συνέχεια θα βασιστεί το νευρο-ασαφές
%μοντέλο Supfunis για να ορίσει
%αυτούς τους κανόνες σαν αρχικούς
%και να τους ανανεώσει, δίνοντας τελικά
%νέους κανόνες με βάση την εκπαίδευση
%που θα έχει δεχτεί από τα πρότυπα εισόδου

%n=είσοδοι, q=κανόνες, p=έξοδοι, fileid=προοίμιο του ονόματος αρχείου


%GIA TA XS
pin = [];
for i = 1:n,
   pin(i,1) = 0.2;
end
dlmwrite(strcat(fileid,'Xs.txt'),pin,' ');
%GIA TA WC && WS
pin = [];
for i = 1:n,
   for j = 1:q,
      pin(i,j) = 0.5;
```

```
    end
end
%-------------------------------------------------
%-------------------- KANONES ---------------------
%-------------------------------------------------

%-------KANONAS1-----1-1-1-----
pin = bitDefine(1,-37,'c',pin);
pin = bitDefine(1,-36,'t',pin);
pin = bitDefine(1,-35,'t',pin);
pin = bitDefine(1,-34,'g',pin);
pin = bitDefine(1,-33,'a',pin);
pin = bitDefine(1,-32,'c',pin);
%------------------------------
pin = bitDefine(1,-14,'t',pin);
pin = bitDefine(1,-13,'a',pin);
pin = bitDefine(1,-12,'t',pin);
pin = bitDefine(1,-11,'a',pin);
pin = bitDefine(1,-10,'a',pin);
pin = bitDefine(1,-9,'t',pin);
%------------------------------
pin = bitDefine(1,-47,'c',pin);
pin = bitDefine(1,-46,'a',pin);
pin = bitDefine(1,-45,'a',pin);
pin = bitDefine(1,-43,'t',pin);
pin = bitDefine(1,-42,'t',pin);
pin = bitDefine(1,-40,'a',pin);
pin = bitDefine(1,-39,'c',pin);
pin = bitDefine(1,-22,'g',pin);
pin = bitDefine(1,-18,'t',pin);
pin = bitDefine(1,-16,'c',pin);
pin = bitDefine(1,-8,'g',pin);
pin = bitDefine(1,-7,'c',pin);
pin = bitDefine(1,-6,'g',pin);
pin = bitDefine(1,-5,'c',pin);
pin = bitDefine(1,-4,'c',pin);
pin = bitDefine(1,-2,'c',pin);
pin = bitDefine(1,-1,'c',pin);
%------------------------------
%-----------KANONAS2--1-1-2----
pin = bitDefine(2,-37,'c',pin);
pin = bitDefine(2,-36,'t',pin);
pin = bitDefine(2,-35,'t',pin);
pin = bitDefine(2,-34,'g',pin);
pin = bitDefine(2,-33,'a',pin);
pin = bitDefine(2,-32,'c',pin);
% %-----------------------------
pin = bitDefine(2,-14,'t',pin);
pin = bitDefine(2,-13,'a',pin);
pin = bitDefine(2,-12,'t',pin);
```

```
pin = bitDefine(2,-11,'a',pin);
pin = bitDefine(2,-10,'a',pin);
pin = bitDefine(2,-9,'t',pin);
%------------------------------
pin = bitDefine(2,-45,'a',pin);
pin = bitDefine(2,-44,'a',pin);
pin = bitDefine(2,-41,'a',pin);
%------------------------------
%------------KANONAS3--1-1-3---
pin = bitDefine(3,-37,'c',pin);
pin = bitDefine(3,-36,'t',pin);
pin = bitDefine(3,-35,'t',pin);
pin = bitDefine(3,-34,'g',pin);
pin = bitDefine(3,-33,'a',pin);
pin = bitDefine(3,-32,'c',pin);
% %------------------------------
pin = bitDefine(3,-14,'t',pin);
pin = bitDefine(3,-13,'a',pin);
pin = bitDefine(3,-12,'t',pin);
pin = bitDefine(3,-11,'a',pin);
pin = bitDefine(3,-10,'a',pin);
pin = bitDefine(3,-9,'t',pin);
%------------------------------
pin = bitDefine(3,-49,'a',pin);
pin = bitDefine(3,-44,'t',pin);
pin = bitDefine(3,-27,'t',pin);
pin = bitDefine(3,-22,'a',pin);
pin = bitDefine(3,-18,'t',pin);
pin = bitDefine(3,-16,'t',pin);
pin = bitDefine(3,-15,'g',pin);
pin = bitDefine(3,-1,'a',pin);
%-----------------------------
%----------KANONAS4---1-1-4---
pin = bitDefine(4,-37,'c',pin);
pin = bitDefine(4,-36,'t',pin);
pin = bitDefine(4,-35,'t',pin);
pin = bitDefine(4,-34,'g',pin);
pin = bitDefine(4,-33,'a',pin);
pin = bitDefine(4,-32,'c',pin);
% %------------------------------
pin = bitDefine(4,-14,'t',pin);
pin = bitDefine(4,-13,'a',pin);
pin = bitDefine(4,-12,'t',pin);
pin = bitDefine(4,-11,'a',pin);
pin = bitDefine(4,-10,'a',pin);
pin = bitDefine(4,-9,'t',pin);
%------------------------------
pin = bitDefine(4,-45,'a',pin);
pin = bitDefine(4,-41,'a',pin);
pin = bitDefine(4,-28,'t',pin);
```

```
pin = bitDefine(4,-27,'t',pin);
pin = bitDefine(4,-23,'t',pin);
pin = bitDefine(4,-21,'a',pin);
pin = bitDefine(4,-20,'a',pin);
pin = bitDefine(4,-17,'t',pin);
pin = bitDefine(4,-15,'t',pin);
pin = bitDefine(4,-4,'a',pin);
%-----------------------------
%----------KANONAS5--1-2-1-----
pin = bitDefine(2,-37,'c',pin);
pin = bitDefine(2,-36,'t',pin);
pin = bitDefine(2,-35,'t',pin);
pin = bitDefine(2,-34,'g',pin);
pin = bitDefine(2,-33,'a',pin);
pin = bitDefine(2,-32,'c',pin);
%-----------------------------
pin = bitDefine(2,-13,'t',pin);
pin = bitDefine(2,-12,'a',pin);
pin = bitDefine(2,-10,'a',pin);
pin = bitDefine(2,-8,'t',pin);
%-----------------------------
pin = bitDefine(5,-47,'c',pin);
pin = bitDefine(5,-46,'a',pin);
pin = bitDefine(5,-45,'a',pin);
pin = bitDefine(5,-43,'t',pin);
pin = bitDefine(5,-42,'t',pin);
pin = bitDefine(5,-40,'a',pin);
pin = bitDefine(5,-39,'c',pin);
pin = bitDefine(5,-22,'g',pin);
pin = bitDefine(5,-18,'t',pin);
pin = bitDefine(5,-16,'c',pin);
pin = bitDefine(5,-8,'g',pin);
pin = bitDefine(5,-7,'c',pin);
pin = bitDefine(5,-6,'g',pin);
pin = bitDefine(5,-5,'c',pin);
pin = bitDefine(5,-4,'c',pin);
pin = bitDefine(5,-2,'c',pin);
pin = bitDefine(5,-1,'c',pin);
%-----------------------------
%---------KANONAS6--1-2-2-----
pin = bitDefine(6,-37,'c',pin);
pin = bitDefine(6,-36,'t',pin);
pin = bitDefine(6,-35,'t',pin);
pin = bitDefine(6,-34,'g',pin);
pin = bitDefine(6,-33,'a',pin);
pin = bitDefine(6,-32,'c',pin);
% %-----------------------------
pin = bitDefine(6,-13,'t',pin);
pin = bitDefine(6,-12,'a',pin);
pin = bitDefine(6,-10,'a',pin);
```

```
pin = bitDefine(6,-8,'t',pin);
%-----------------------------
pin = bitDefine(6,-45,'a',pin);
pin = bitDefine(6,-44,'a',pin);
pin = bitDefine(6,-41,'a',pin);
%-----------------------------
%---------KANONAS7--1-2-3------
pin = bitDefine(7,-37,'c',pin);
pin = bitDefine(7,-36,'t',pin);
pin = bitDefine(7,-35,'t',pin);
pin = bitDefine(7,-34,'g',pin);
pin = bitDefine(7,-33,'a',pin);
pin = bitDefine(7,-32,'c',pin);
% %-----------------------------
pin = bitDefine(7,-13,'t',pin);
pin = bitDefine(7,-12,'a',pin);
pin = bitDefine(7,-10,'a',pin);
pin = bitDefine(7,-8,'t',pin);
%-----------------------------
pin = bitDefine(7,-49,'a',pin);
pin = bitDefine(7,-44,'t',pin);
pin = bitDefine(7,-27,'t',pin);
pin = bitDefine(7,-22,'a',pin);
pin = bitDefine(7,-18,'t',pin);
pin = bitDefine(7,-16,'t',pin);
pin = bitDefine(7,-15,'g',pin);
pin = bitDefine(7,-1,'a',pin);
%-----------------------------
%----------KANONAS8--1-2-4-----
pin = bitDefine(8,-37,'c',pin);
pin = bitDefine(8,-36,'t',pin);
pin = bitDefine(8,-35,'t',pin);
pin = bitDefine(8,-34,'g',pin);
pin = bitDefine(8,-33,'a',pin);
pin = bitDefine(8,-32,'c',pin);
% %-----------------------------
pin = bitDefine(8,-13,'t',pin);
pin = bitDefine(8,-12,'a',pin);
pin = bitDefine(8,-10,'a',pin);
pin = bitDefine(8,-8,'t',pin);
%-----------------------------
pin = bitDefine(8,-45,'a',pin);
pin = bitDefine(8,-41,'a',pin);
pin = bitDefine(8,-28,'t',pin);
pin = bitDefine(8,-27,'t',pin);
pin = bitDefine(8,-23,'t',pin);
pin = bitDefine(8,-21,'a',pin);
pin = bitDefine(8,-20,'a',pin);
pin = bitDefine(8,-17,'t',pin);
pin = bitDefine(8,-15,'t',pin);
```

```
pin = bitDefine(8,-4,'a',pin);
%-----------------------------
%----------KANONAS9--1-3-1-----
pin = bitDefine(3,-37,'c',pin);
pin = bitDefine(3,-36,'t',pin);
pin = bitDefine(3,-35,'t',pin);
pin = bitDefine(3,-34,'g',pin);
pin = bitDefine(3,-33,'a',pin);
pin = bitDefine(3,-32,'c',pin);
%-----------------------------
pin = bitDefine(3,-13,'t',pin);
pin = bitDefine(3,-12,'a',pin);
pin = bitDefine(3,-11,'t',pin);
pin = bitDefine(3,-10,'a',pin);
pin = bitDefine(3,-9,'a',pin);
pin = bitDefine(3,-8,'t',pin);
%-----------------------------
pin = bitDefine(9,-47,'c',pin);
pin = bitDefine(9,-46,'a',pin);
pin = bitDefine(9,-45,'a',pin);
pin = bitDefine(9,-43,'t',pin);
pin = bitDefine(9,-42,'t',pin);
pin = bitDefine(9,-40,'a',pin);
pin = bitDefine(9,-39,'c',pin);
pin = bitDefine(9,-22,'g',pin);
pin = bitDefine(9,-18,'t',pin);
pin = bitDefine(9,-16,'c',pin);
pin = bitDefine(9,-8,'g',pin);
pin = bitDefine(9,-7,'c',pin);
pin = bitDefine(9,-6,'g',pin);
pin = bitDefine(9,-5,'c',pin);
pin = bitDefine(9,-4,'c',pin);
pin = bitDefine(9,-2,'c',pin);
pin = bitDefine(9,-1,'c',pin);
%-----------------------------
%--------KANONAS10--1-3-2-----
pin = bitDefine(10,-37,'c',pin);
pin = bitDefine(10,-36,'t',pin);
pin = bitDefine(10,-35,'t',pin);
pin = bitDefine(10,-34,'g',pin);
pin = bitDefine(10,-33,'a',pin);
pin = bitDefine(10,-32,'c',pin);
% %-----------------------------
pin = bitDefine(10,-13,'t',pin);
pin = bitDefine(10,-12,'a',pin);
pin = bitDefine(10,-11,'t',pin);
pin = bitDefine(10,-10,'a',pin);
pin = bitDefine(10,-9,'a',pin);
pin = bitDefine(10,-8,'t',pin);
%-----------------------------
```

```
pin = bitDefine(10,-45,'a',pin);
pin = bitDefine(10,-44,'a',pin);
pin = bitDefine(10,-41,'a',pin);
%-----------------------------
%---------KANONAS11---1-3-3----
pin = bitDefine(11,-37,'c',pin);
pin = bitDefine(11,-36,'t',pin);
pin = bitDefine(11,-35,'t',pin);
pin = bitDefine(11,-34,'g',pin);
pin = bitDefine(11,-33,'a',pin);
pin = bitDefine(11,-32,'c',pin);
% %-----------------------------
pin = bitDefine(11,-13,'t',pin);
pin = bitDefine(11,-12,'a',pin);
pin = bitDefine(11,-11,'t',pin);
pin = bitDefine(11,-10,'a',pin);
pin = bitDefine(11,-9,'a',pin);
pin = bitDefine(11,-8,'t',pin);
%-----------------------------
pin = bitDefine(11,-49,'a',pin);
pin = bitDefine(11,-44,'t',pin);
pin = bitDefine(11,-27,'t',pin);
pin = bitDefine(11,-22,'a',pin);
pin = bitDefine(11,-18,'t',pin);
pin = bitDefine(11,-16,'t',pin);
pin = bitDefine(11,-15,'g',pin);
pin = bitDefine(11,-1,'a',pin);
%-----------------------------
%----------KANONAS12--1-3-4----
pin = bitDefine(12,-37,'c',pin);
pin = bitDefine(12,-36,'t',pin);
pin = bitDefine(12,-35,'t',pin);
pin = bitDefine(12,-34,'g',pin);
pin = bitDefine(12,-33,'a',pin);
pin = bitDefine(12,-32,'c',pin);
% %-----------------------------
pin = bitDefine(12,-13,'t',pin);
pin = bitDefine(12,-12,'a',pin);
pin = bitDefine(12,-11,'t',pin);
pin = bitDefine(12,-10,'a',pin);
pin = bitDefine(12,-9,'a',pin);
pin = bitDefine(12,-8,'t',pin);
%-----------------------------
pin = bitDefine(12,-45,'a',pin);
pin = bitDefine(12,-41,'a',pin);
pin = bitDefine(12,-28,'t',pin);
pin = bitDefine(12,-27,'t',pin);
pin = bitDefine(12,-23,'t',pin);
pin = bitDefine(12,-21,'a',pin);
pin = bitDefine(12,-20,'a',pin);
```

```
pin = bitDefine(12,-17,'t',pin);
pin = bitDefine(12,-15,'t',pin);
pin = bitDefine(12,-4,'a',pin);
%-----------------------------
%----------KANONAS13---1-4-1---
pin = bitDefine(4,-37,'c',pin);
pin = bitDefine(4,-36,'t',pin);
pin = bitDefine(4,-35,'t',pin);
pin = bitDefine(4,-34,'g',pin);
pin = bitDefine(4,-33,'a',pin);
pin = bitDefine(4,-32,'c',pin);
%-----------------------------
pin = bitDefine(4,-12,'t',pin);
pin = bitDefine(4,-11,'a',pin);
pin = bitDefine(4,-7,'t',pin);
%-----------------------------
pin = bitDefine(13,-47,'c',pin);
pin = bitDefine(13,-46,'a',pin);
pin = bitDefine(13,-45,'a',pin);
pin = bitDefine(13,-43,'t',pin);
pin = bitDefine(13,-42,'t',pin);
pin = bitDefine(13,-40,'a',pin);
pin = bitDefine(13,-39,'c',pin);
pin = bitDefine(13,-22,'g',pin);
pin = bitDefine(13,-18,'t',pin);
pin = bitDefine(13,-16,'c',pin);
pin = bitDefine(13,-8,'g',pin);
pin = bitDefine(13,-7,'c',pin);
pin = bitDefine(13,-6,'g',pin);
pin = bitDefine(13,-5,'c',pin);
pin = bitDefine(13,-4,'c',pin);
pin = bitDefine(13,-2,'c',pin);
pin = bitDefine(13,-1,'c',pin);
%-----------------------------
%--------KANONAS14--1-4-2------
pin = bitDefine(14,-37,'c',pin);
pin = bitDefine(14,-36,'t',pin);
pin = bitDefine(14,-35,'t',pin);
pin = bitDefine(14,-34,'g',pin);
pin = bitDefine(14,-33,'a',pin);
pin = bitDefine(14,-32,'c',pin);
% %----------------------------
pin = bitDefine(14,-12,'t',pin);
pin = bitDefine(14,-11,'a',pin);
pin = bitDefine(14,-7,'t',pin);
%-----------------------------
pin = bitDefine(14,-45,'a',pin);
pin = bitDefine(14,-44,'a',pin);
pin = bitDefine(14,-41,'a',pin);
%-----------------------------
```

```
%--------KANONAS15--1-4-3-----
pin = bitDefine(15,-37,'c',pin);
pin = bitDefine(15,-36,'t',pin);
pin = bitDefine(15,-35,'t',pin);
pin = bitDefine(15,-34,'g',pin);
pin = bitDefine(15,-33,'a',pin);
pin = bitDefine(15,-32,'c',pin);
% %-----------------------------
pin = bitDefine(15,-12,'t',pin);
pin = bitDefine(15,-11,'a',pin);
pin = bitDefine(15,-7,'t',pin);
%-----------------------------
pin = bitDefine(15,-49,'a',pin);
pin = bitDefine(15,-44,'t',pin);
pin = bitDefine(15,-27,'t',pin);
pin = bitDefine(15,-22,'a',pin);
pin = bitDefine(15,-18,'t',pin);
pin = bitDefine(15,-16,'t',pin);
pin = bitDefine(15,-15,'g',pin);
pin = bitDefine(15,-1,'a',pin);
%-----------------------------
%----------KANONAS16--1-4-4------
pin = bitDefine(16,-37,'c',pin);
pin = bitDefine(16,-36,'t',pin);
pin = bitDefine(16,-35,'t',pin);
pin = bitDefine(16,-34,'g',pin);
pin = bitDefine(16,-33,'a',pin);
pin = bitDefine(16,-32,'c',pin);
% %-----------------------------
pin = bitDefine(16,-12,'t',pin);
pin = bitDefine(16,-11,'a',pin);
pin = bitDefine(16,-7,'t',pin);
%-----------------------------
pin = bitDefine(16,-45,'a',pin);
pin = bitDefine(16,-41,'a',pin);
pin = bitDefine(16,-28,'t',pin);
pin = bitDefine(16,-27,'t',pin);
pin = bitDefine(16,-23,'t',pin);
pin = bitDefine(16,-21,'a',pin);
pin = bitDefine(16,-20,'a',pin);
pin = bitDefine(16,-17,'t',pin);
pin = bitDefine(16,-15,'t',pin);
pin = bitDefine(16,-4,'a',pin);
%-----------------------------
%----------KANONAS17---2-1-1---
pin = bitDefine(5,-36,'t',pin);
pin = bitDefine(5,-35,'t',pin);
pin = bitDefine(5,-34,'g',pin);
pin = bitDefine(5,-32,'c',pin);
pin = bitDefine(5,-31,'a',pin);
```

```
%-----------------------------
pin = bitDefine(5,-14,'t',pin);
pin = bitDefine(5,-13,'a',pin);
pin = bitDefine(5,-12,'t',pin);
pin = bitDefine(5,-11,'a',pin);
pin = bitDefine(5,-10,'a',pin);
pin = bitDefine(5,-9,'t',pin);
%-----------------------------
pin = bitDefine(17,-47,'c',pin);
pin = bitDefine(17,-46,'a',pin);
pin = bitDefine(17,-45,'a',pin);
pin = bitDefine(17,-43,'t',pin);
pin = bitDefine(17,-42,'t',pin);
pin = bitDefine(17,-40,'a',pin);
pin = bitDefine(17,-39,'c',pin);
pin = bitDefine(17,-22,'g',pin);
pin = bitDefine(17,-18,'t',pin);
pin = bitDefine(17,-16,'c',pin);
pin = bitDefine(17,-8,'g',pin);
pin = bitDefine(17,-7,'c',pin);
pin = bitDefine(17,-6,'g',pin);
pin = bitDefine(17,-5,'c',pin);
pin = bitDefine(17,-4,'c',pin);
pin = bitDefine(17,-2,'c',pin);
pin = bitDefine(17,-1,'c',pin);
%-----------------------------
%-----------KANONAS18---2-1-2--
pin = bitDefine(18,-36,'t',pin);
pin = bitDefine(18,-35,'t',pin);
pin = bitDefine(18,-34,'g',pin);
pin = bitDefine(18,-32,'c',pin);
pin = bitDefine(18,-31,'a',pin);
% %----------------------------
pin = bitDefine(18,-14,'t',pin);
pin = bitDefine(18,-13,'a',pin);
pin = bitDefine(18,-12,'t',pin);
pin = bitDefine(18,-11,'a',pin);
pin = bitDefine(18,-10,'a',pin);
pin = bitDefine(18,-9,'t',pin);
%-----------------------------
pin = bitDefine(18,-45,'a',pin);
pin = bitDefine(18,-44,'a',pin);
pin = bitDefine(18,-41,'a',pin);
%-----------------------------
%------------KANONAS19--2-1-3--
pin = bitDefine(19,-36,'t',pin);
pin = bitDefine(19,-35,'t',pin);
pin = bitDefine(19,-34,'g',pin);
pin = bitDefine(19,-32,'c',pin);
pin = bitDefine(19,-31,'a',pin);
```

```
% %-----------------------------
pin = bitDefine(19,-14,'t',pin);
pin = bitDefine(19,-13,'a',pin);
pin = bitDefine(19,-12,'t',pin);
pin = bitDefine(19,-11,'a',pin);
pin = bitDefine(19,-10,'a',pin);
pin = bitDefine(19,-9,'t',pin);
%-----------------------------
pin = bitDefine(19,-49,'a',pin);
pin = bitDefine(19,-44,'t',pin);
pin = bitDefine(19,-27,'t',pin);
pin = bitDefine(19,-22,'a',pin);
pin = bitDefine(19,-18,'t',pin);
pin = bitDefine(19,-16,'t',pin);
pin = bitDefine(19,-15,'g',pin);
pin = bitDefine(19,-1,'a',pin);
%-----------------------------
%----------KANONAS20--2-1-4----
pin = bitDefine(20,-36,'t',pin);
pin = bitDefine(20,-35,'t',pin);
pin = bitDefine(20,-34,'g',pin);
pin = bitDefine(20,-32,'c',pin);
pin = bitDefine(20,-31,'a',pin);
% %-----------------------------
pin = bitDefine(20,-14,'t',pin);
pin = bitDefine(20,-13,'a',pin);
pin = bitDefine(20,-12,'t',pin);
pin = bitDefine(20,-11,'a',pin);
pin = bitDefine(20,-10,'a',pin);
pin = bitDefine(20,-9,'t',pin);
%-----------------------------
pin = bitDefine(20,-45,'a',pin);
pin = bitDefine(20,-41,'a',pin);
pin = bitDefine(20,-28,'t',pin);
pin = bitDefine(20,-27,'t',pin);
pin = bitDefine(20,-23,'t',pin);
pin = bitDefine(20,-21,'a',pin);
pin = bitDefine(20,-20,'a',pin);
pin = bitDefine(20,-17,'t',pin);
pin = bitDefine(20,-15,'t',pin);
pin = bitDefine(20,-4,'a',pin);
%-----------------------------
%----------KANONAS21--2-2-1----
pin = bitDefine(6,-36,'t',pin);
pin = bitDefine(6,-35,'t',pin);
pin = bitDefine(6,-34,'g',pin);
pin = bitDefine(6,-32,'c',pin);
pin = bitDefine(6,-31,'a',pin);
%-----------------------------
pin = bitDefine(6,-13,'t',pin);
```

```
pin = bitDefine(6,-12,'a',pin);
pin = bitDefine(6,-10,'a',pin);
pin = bitDefine(6,-8,'t',pin);
%------------------------------
pin = bitDefine(21,-47,'c',pin);
pin = bitDefine(21,-46,'a',pin);
pin = bitDefine(21,-45,'a',pin);
pin = bitDefine(21,-43,'t',pin);
pin = bitDefine(21,-42,'t',pin);
pin = bitDefine(21,-40,'a',pin);
pin = bitDefine(21,-39,'c',pin);
pin = bitDefine(21,-22,'g',pin);
pin = bitDefine(21,-18,'t',pin);
pin = bitDefine(21,-16,'c',pin);
pin = bitDefine(21,-8,'g',pin);
pin = bitDefine(21,-7,'c',pin);
pin = bitDefine(21,-6,'g',pin);
pin = bitDefine(21,-5,'c',pin);
pin = bitDefine(21,-4,'c',pin);
pin = bitDefine(21,-2,'c',pin);
pin = bitDefine(21,-1,'c',pin);
%------------------------------
%-----------KANONAS22---2-2-2--
pin = bitDefine(22,-36,'t',pin);
pin = bitDefine(22,-35,'t',pin);
pin = bitDefine(22,-34,'g',pin);
pin = bitDefine(22,-32,'c',pin);
pin = bitDefine(22,-31,'a',pin);
% %-----------------------------
pin = bitDefine(22,-13,'t',pin);
pin = bitDefine(22,-12,'a',pin);
pin = bitDefine(22,-10,'a',pin);
pin = bitDefine(22,-8,'t',pin);
%------------------------------
pin = bitDefine(22,-45,'a',pin);
pin = bitDefine(22,-44,'a',pin);
pin = bitDefine(22,-41,'a',pin);
%------------------------------
%------------KANONAS23--2-2-3--
pin = bitDefine(23,-36,'t',pin);
pin = bitDefine(23,-35,'t',pin);
pin = bitDefine(23,-34,'g',pin);
pin = bitDefine(23,-32,'c',pin);
pin = bitDefine(23,-31,'a',pin);
% %-----------------------------
pin = bitDefine(23,-13,'t',pin);
pin = bitDefine(23,-12,'a',pin);
pin = bitDefine(23,-10,'a',pin);
pin = bitDefine(23,-8,'t',pin);
%------------------------------
```

```
pin = bitDefine(23,-49,'a',pin);
pin = bitDefine(23,-44,'t',pin);
pin = bitDefine(23,-27,'t',pin);
pin = bitDefine(23,-22,'a',pin);
pin = bitDefine(23,-18,'t',pin);
pin = bitDefine(23,-16,'t',pin);
pin = bitDefine(23,-15,'g',pin);
pin = bitDefine(23,-1,'a',pin);
%------------------------------
%----------KANONAS24---2-2-4---
pin = bitDefine(24,-36,'t',pin);
pin = bitDefine(24,-35,'t',pin);
pin = bitDefine(24,-34,'g',pin);
pin = bitDefine(24,-32,'c',pin);
pin = bitDefine(24,-31,'a',pin);
%------------------------------
pin = bitDefine(24,-13,'t',pin);
pin = bitDefine(24,-12,'a',pin);
pin = bitDefine(24,-10,'a',pin);
pin = bitDefine(24,-8,'t',pin);
------------------------------
pin = bitDefine(24,-45,'a',pin);
pin = bitDefine(24,-41,'a',pin);
pin = bitDefine(24,-28,'t',pin);
pin = bitDefine(24,-27,'t',pin);
pin = bitDefine(24,-23,'t',pin);
pin = bitDefine(24,-21,'a',pin);
pin = bitDefine(24,-20,'a',pin);
pin = bitDefine(24,-17,'t',pin);
pin = bitDefine(24,-15,'t',pin);
pin = bitDefine(24,-4,'a',pin);
------------------------------
%----------KANONAS25--2-3-1----
pin = bitDefine(7,-36,'t',pin);
pin = bitDefine(7,-35,'t',pin);
pin = bitDefine(7,-34,'g',pin);
pin = bitDefine(7,-32,'c',pin);
pin = bitDefine(7,-31,'a',pin);
%------------------------------
pin = bitDefine(7,-13,'t',pin);
pin = bitDefine(7,-12,'a',pin);
pin = bitDefine(7,-11,'t',pin);
pin = bitDefine(7,-10,'a',pin);
pin = bitDefine(7,-9,'a',pin);
pin = bitDefine(7,-8,'t',pin);
%------------------------------
pin = bitDefine(25,-47,'c',pin);
pin = bitDefine(25,-46,'a',pin);
pin = bitDefine(25,-45,'a',pin);
pin = bitDefine(25,-43,'t',pin);
```

```
pin = bitDefine(25,-42,'t',pin);
pin = bitDefine(25,-40,'a',pin);
pin = bitDefine(25,-39,'c',pin);
pin = bitDefine(25,-22,'g',pin);
pin = bitDefine(25,-18,'t',pin);
pin = bitDefine(25,-16,'c',pin);
pin = bitDefine(25,-8,'g',pin);
pin = bitDefine(25,-7,'c',pin);
pin = bitDefine(25,-6,'g',pin);
pin = bitDefine(25,-5,'c',pin);
pin = bitDefine(25,-4,'c',pin);
pin = bitDefine(25,-2,'c',pin);
pin = bitDefine(25,-1,'c',pin);
%-----------------------------
%--KANONAS26--2-3-2------------
pin = bitDefine(26,-36,'t',pin);
pin = bitDefine(26,-35,'t',pin);
pin = bitDefine(26,-34,'g',pin);
pin = bitDefine(26,-32,'c',pin);
pin = bitDefine(26,-31,'a',pin);
% %-----------------------------
pin = bitDefine(26,-13,'t',pin);
pin = bitDefine(26,-12,'a',pin);
pin = bitDefine(26,-11,'t',pin);
pin = bitDefine(26,-10,'a',pin);
pin = bitDefine(26,-9,'a',pin);
pin = bitDefine(26,-8,'t',pin);
%-----------------------------
pin = bitDefine(26,-45,'a',pin);
pin = bitDefine(26,-44,'a',pin);
pin = bitDefine(26,-41,'a',pin);
%-----------------------------
%--KANONAS27----2-3-3----------
pin = bitDefine(27,-36,'t',pin);
pin = bitDefine(27,-35,'t',pin);
pin = bitDefine(27,-34,'g',pin);
pin = bitDefine(27,-32,'c',pin);
pin = bitDefine(27,-31,'a',pin);
%-----------------------------
pin = bitDefine(27,-13,'t',pin);
pin = bitDefine(27,-12,'a',pin);
pin = bitDefine(27,-11,'t',pin);
pin = bitDefine(27,-10,'a',pin);
pin = bitDefine(27,-9,'a',pin);
pin = bitDefine(27,-8,'t',pin);
%-----------------------------
pin = bitDefine(27,-49,'a',pin);
pin = bitDefine(27,-44,'t',pin);
pin = bitDefine(27,-27,'t',pin);
pin = bitDefine(27,-22,'a',pin);
```

```
pin = bitDefine(27,-18,'t',pin);
pin = bitDefine(27,-16,'t',pin);
pin = bitDefine(27,-15,'g',pin);
pin = bitDefine(27,-1,'a',pin);
%-----------------------------
%--KANONAS28-----2-3-4---------
pin = bitDefine(28,-36,'t',pin);
pin = bitDefine(28,-35,'t',pin);
pin = bitDefine(28,-34,'g',pin);
pin = bitDefine(28,-32,'c',pin);
pin = bitDefine(28,-31,'a',pin);
%-----------------------------
pin = bitDefine(28,-13,'t',pin);
pin = bitDefine(28,-12,'a',pin);
pin = bitDefine(28,-11,'t',pin);
pin = bitDefine(28,-10,'a',pin);
pin = bitDefine(28,-9,'a',pin);
pin = bitDefine(28,-8,'t',pin);
%----------------------------
pin = bitDefine(28,-45,'a',pin);
pin = bitDefine(28,-41,'a',pin);
pin = bitDefine(28,-28,'t',pin);
pin = bitDefine(28,-27,'t',pin);
pin = bitDefine(28,-23,'t',pin);
pin = bitDefine(28,-21,'a',pin);
pin = bitDefine(28,-20,'a',pin);
pin = bitDefine(28,-17,'t',pin);
pin = bitDefine(28,-15,'t',pin);
pin = bitDefine(28,-4,'a',pin);
%-----------------------------
%--KANONAS29---2-4-1-----------
pin = bitDefine(8,-36,'t',pin);
pin = bitDefine(8,-35,'t',pin);
pin = bitDefine(8,-34,'g',pin);
pin = bitDefine(8,-32,'c',pin);
pin = bitDefine(8,-31,'a',pin);
%-----------------------------
pin = bitDefine(8,-12,'t',pin);
pin = bitDefine(8,-11,'a',pin);
pin = bitDefine(8,-7,'t',pin);
%----------------------------
pin = bitDefine(29,-47,'c',pin);
pin = bitDefine(29,-46,'a',pin);
pin = bitDefine(29,-45,'a',pin);
pin = bitDefine(29,-43,'t',pin);
pin = bitDefine(29,-42,'t',pin);
pin = bitDefine(29,-40,'a',pin);
pin = bitDefine(29,-39,'c',pin);
pin = bitDefine(29,-22,'g',pin);
pin = bitDefine(29,-18,'t',pin);
```

```
pin = bitDefine(29,-16,'c',pin);
pin = bitDefine(29,-8,'g',pin);
pin = bitDefine(29,-7,'c',pin);
pin = bitDefine(29,-6,'g',pin);
pin = bitDefine(29,-5,'c',pin);
pin = bitDefine(29,-4,'c',pin);
pin = bitDefine(29,-2,'c',pin);
pin = bitDefine(29,-1,'c',pin);
%-----------------------------
%--KANONAS30--2-4-2------------
pin = bitDefine(30,-36,'t',pin);
pin = bitDefine(30,-35,'t',pin);
pin = bitDefine(30,-34,'g',pin);
pin = bitDefine(30,-32,'c',pin);
pin = bitDefine(30,-31,'a',pin);
%-----------------------------
pin = bitDefine(30,-12,'t',pin);
pin = bitDefine(30,-11,'a',pin);
pin = bitDefine(30,-7,'t',pin);
%-----------------------------
pin = bitDefine(30,-45,'a',pin);
pin = bitDefine(30,-44,'a',pin);
pin = bitDefine(30,-41,'a',pin);
%-----------------------------
%--KANONAS31----2-4-3----------
pin = bitDefine(31,-36,'t',pin);
pin = bitDefine(31,-35,'t',pin);
pin = bitDefine(31,-34,'g',pin);
pin = bitDefine(31,-32,'c',pin);
pin = bitDefine(31,-31,'a',pin);
%-----------------------------
pin = bitDefine(31,-12,'t',pin);
pin = bitDefine(31,-11,'a',pin);
pin = bitDefine(31,-7,'t',pin);
%-----------------------------
pin = bitDefine(31,-49,'a',pin);
pin = bitDefine(31,-44,'t',pin);
pin = bitDefine(31,-27,'t',pin);
pin = bitDefine(31,-22,'a',pin);
pin = bitDefine(31,-18,'t',pin);
pin = bitDefine(31,-16,'t',pin);
pin = bitDefine(31,-15,'g',pin);
pin = bitDefine(31,-1,'a',pin);
%-----------------------------
--KANONAS32-----2-4-4---------
pin = bitDefine(32,-36,'t',pin);
pin = bitDefine(32,-35,'t',pin);
pin = bitDefine(32,-34,'g',pin);
pin = bitDefine(32,-32,'c',pin);
pin = bitDefine(32,-31,'a',pin);
```

```
%-------------------------------
pin = bitDefine(32,-12,'t',pin);
pin = bitDefine(32,-11,'a',pin);
pin = bitDefine(32,-7,'t',pin);
%-----------------------------
pin = bitDefine(32,-45,'a',pin);
pin = bitDefine(32,-41,'a',pin);
pin = bitDefine(32,-28,'t',pin);
pin = bitDefine(32,-27,'t',pin);
pin = bitDefine(32,-23,'t',pin);
pin = bitDefine(32,-21,'a',pin);
pin = bitDefine(32,-20,'a',pin);
pin = bitDefine(32,-17,'t',pin);
pin = bitDefine(32,-15,'t',pin);
pin = bitDefine(32,-4,'a',pin);


%-----KANONAS 33 3 1 1----------
%-------------3-----------------
pin = bitDefine(9,-36,'t',pin);
pin = bitDefine(9,-35,'t',pin);
pin = bitDefine(9,-34,'g',pin);
pin = bitDefine(9,-33,'a',pin);
pin = bitDefine(9,-32,'c',pin);
pin = bitDefine(9,-31,'a',pin);
%-------------1-----------------
pin = bitDefine(9,-14,'t',pin);
pin = bitDefine(9,-13,'a',pin);
pin = bitDefine(9,-12,'t',pin);
pin = bitDefine(9,-11,'a',pin);
pin = bitDefine(9,-10,'a',pin);
pin = bitDefine(9,-9,'t',pin);
%-------------1-----------------
pin = bitDefine(33,-47,'c',pin);
pin = bitDefine(33,-46,'a',pin);
pin = bitDefine(33,-45,'a',pin);
pin = bitDefine(33,-43,'t',pin);
pin = bitDefine(33,-42,'t',pin);
pin = bitDefine(33,-40,'a',pin);
pin = bitDefine(33,-39,'c',pin);
pin = bitDefine(33,-22,'g',pin);
pin = bitDefine(33,-18,'t',pin);
pin = bitDefine(33,-16,'c',pin);
pin = bitDefine(33,-8,'g',pin);
pin = bitDefine(33,-7,'c',pin);
pin = bitDefine(33,-6,'g',pin);
pin = bitDefine(33,-5,'c',pin);
pin = bitDefine(33,-4,'c',pin);
pin = bitDefine(33,-2,'c',pin);
pin = bitDefine(33,-1,'c',pin);
```

```
%-------------------------------

%-----KANONAS 34 3 1 2----------
%-------------3-----------------
pin = bitDefine(34,-36,'t',pin);
pin = bitDefine(34,-35,'t',pin);
pin = bitDefine(34,-34,'g',pin);
pin = bitDefine(34,-33,'a',pin);
pin = bitDefine(34,-32,'c',pin);
pin = bitDefine(34,-31,'a',pin);
%-------------1-----------------
pin = bitDefine(34,-14,'t',pin);
pin = bitDefine(34,-13,'a',pin);
pin = bitDefine(34,-12,'t',pin);
pin = bitDefine(34,-11,'a',pin);
pin = bitDefine(34,-10,'a',pin);
pin = bitDefine(34,-9,'t',pin);
%-------------2-----------------
pin = bitDefine(34,-45,'a',pin);
pin = bitDefine(34,-44,'a',pin);
pin = bitDefine(34,-41,'a',pin);
%-------------------------------

%-----KANONAS 35 3 1 3----------
%-------------3-----------------
pin = bitDefine(35,-36,'t',pin);
pin = bitDefine(35,-35,'t',pin);
pin = bitDefine(35,-34,'g',pin);
pin = bitDefine(35,-33,'a',pin);
pin = bitDefine(35,-32,'c',pin);
pin = bitDefine(35,-31,'a',pin);
%-------------1-----------------
pin = bitDefine(35,-14,'t',pin);
pin = bitDefine(35,-13,'a',pin);
pin = bitDefine(35,-12,'t',pin);
pin = bitDefine(35,-11,'a',pin);
pin = bitDefine(35,-10,'a',pin);
pin = bitDefine(35,-9,'t',pin);
%-------------3-----------------
pin = bitDefine(35,-49,'a',pin);
pin = bitDefine(35,-44,'t',pin);
pin = bitDefine(35,-27,'t',pin);
pin = bitDefine(35,-22,'a',pin);
pin = bitDefine(35,-18,'t',pin);
pin = bitDefine(35,-16,'t',pin);
pin = bitDefine(35,-15,'g',pin);
pin = bitDefine(35,-1,'a',pin);
%-------------------------------

%-----KANONAS 36 3 1 4----------
```

```
%-------------3-----------------
pin = bitDefine(36,-36,'t',pin);
pin = bitDefine(36,-35,'t',pin);
pin = bitDefine(36,-34,'g',pin);
pin = bitDefine(36,-33,'a',pin);
pin = bitDefine(36,-32,'c',pin);
pin = bitDefine(36,-31,'a',pin);
%-------------1-----------------
pin = bitDefine(36,-14,'t',pin);
pin = bitDefine(36,-13,'a',pin);
pin = bitDefine(36,-12,'t',pin);
pin = bitDefine(36,-11,'a',pin);
pin = bitDefine(36,-10,'a',pin);
pin = bitDefine(36,-9,'t',pin);
%-------------4-----------------
pin = bitDefine(36,-45,'a',pin);
pin = bitDefine(36,-41,'a',pin);
pin = bitDefine(36,-28,'t',pin);
pin = bitDefine(36,-27,'t',pin);
pin = bitDefine(36,-23,'t',pin);
pin = bitDefine(36,-21,'a',pin);
pin = bitDefine(36,-20,'a',pin);
pin = bitDefine(36,-17,'t',pin);
pin = bitDefine(36,-15,'t',pin);
pin = bitDefine(36,-4,'a',pin);
%------------------------------

%-----KANONAS 37 3 2 1----------
%-------------3-----------------
pin = bitDefine(10,-36,'t',pin);
pin = bitDefine(10,-35,'t',pin);
pin = bitDefine(10,-34,'g',pin);
pin = bitDefine(10,-33,'a',pin);
pin = bitDefine(10,-32,'c',pin);
pin = bitDefine(10,-31,'a',pin);
%-------------2-----------------
pin = bitDefine(10,-13,'t',pin);
pin = bitDefine(10,-12,'a',pin);
pin = bitDefine(10,-10,'a',pin);
pin = bitDefine(10,-8,'t',pin);
%-------------1-----------------
pin = bitDefine(37,-47,'c',pin);
pin = bitDefine(37,-46,'a',pin);
pin = bitDefine(37,-45,'a',pin);
pin = bitDefine(37,-43,'t',pin);
pin = bitDefine(37,-42,'t',pin);
pin = bitDefine(37,-40,'a',pin);
pin = bitDefine(37,-39,'c',pin);
pin = bitDefine(37,-22,'g',pin);
pin = bitDefine(37,-18,'t',pin);
```

```
pin = bitDefine(37,-16,'c',pin);
pin = bitDefine(37,-8,'g',pin);
pin = bitDefine(37,-7,'c',pin);
pin = bitDefine(37,-6,'g',pin);
pin = bitDefine(37,-5,'c',pin);
pin = bitDefine(37,-4,'c',pin);
pin = bitDefine(37,-2,'c',pin);
pin = bitDefine(37,-1,'c',pin);
%------------------------------

%-----KANONAS 38 3 2 2----------
%-------------3-----------------
pin = bitDefine(38,-36,'t',pin);
pin = bitDefine(38,-35,'t',pin);
pin = bitDefine(38,-34,'g',pin);
pin = bitDefine(38,-33,'a',pin);
pin = bitDefine(38,-32,'c',pin);
pin = bitDefine(38,-31,'a',pin);
%-------------2-----------------
pin = bitDefine(38,-13,'t',pin);
pin = bitDefine(38,-12,'a',pin);
pin = bitDefine(38,-10,'a',pin);
pin = bitDefine(38,-8,'t',pin);
%-------------2-----------------
pin = bitDefine(34,-45,'a',pin);
pin = bitDefine(34,-44,'a',pin);
pin = bitDefine(34,-41,'a',pin);
%------------------------------

%-----KANONAS 39 3 2 3----------
%-------------3-----------------
pin = bitDefine(39,-36,'t',pin);
pin = bitDefine(39,-35,'t',pin);
pin = bitDefine(39,-34,'g',pin);
pin = bitDefine(39,-33,'a',pin);
pin = bitDefine(39,-32,'c',pin);
pin = bitDefine(39,-31,'a',pin);
%-------------2-----------------
pin = bitDefine(39,-13,'t',pin);
pin = bitDefine(39,-12,'a',pin);
pin = bitDefine(39,-10,'a',pin);
pin = bitDefine(39,-8,'t',pin);
%-------------3-----------------
pin = bitDefine(39,-49,'a',pin);
pin = bitDefine(39,-44,'t',pin);
pin = bitDefine(39,-27,'t',pin);
pin = bitDefine(39,-22,'a',pin);
pin = bitDefine(39,-18,'t',pin);
pin = bitDefine(39,-16,'t',pin);
pin = bitDefine(39,-15,'g',pin);
```

```
pin = bitDefine(39,-1,'a',pin);
%------------------------------

%-----KANONAS 40 3 2 4----------
%-------------3-----------------
pin = bitDefine(40,-36,'t',pin);
pin = bitDefine(40,-35,'t',pin);
pin = bitDefine(40,-34,'g',pin);
pin = bitDefine(40,-33,'a',pin);
pin = bitDefine(40,-32,'c',pin);
pin = bitDefine(40,-31,'a',pin);
%-------------2-----------------
pin = bitDefine(40,-13,'t',pin);
pin = bitDefine(40,-12,'a',pin);
pin = bitDefine(40,-10,'a',pin);
pin = bitDefine(40,-8,'t',pin);
%-------------4-----------------
pin = bitDefine(40,-45,'a',pin);
pin = bitDefine(40,-41,'a',pin);
pin = bitDefine(40,-28,'t',pin);
pin = bitDefine(40,-27,'t',pin);
pin = bitDefine(40,-23,'t',pin);
pin = bitDefine(40,-21,'a',pin);
pin = bitDefine(40,-20,'a',pin);
pin = bitDefine(40,-17,'t',pin);
pin = bitDefine(40,-15,'t',pin);
pin = bitDefine(40,-4,'a',pin);
%------------------------------

%-----KANONAS 41 3 3 1----------
%-------------3-----------------
pin = bitDefine(11,-36,'t',pin);
pin = bitDefine(11,-35,'t',pin);
pin = bitDefine(11,-34,'g',pin);
pin = bitDefine(11,-33,'a',pin);
pin = bitDefine(11,-32,'c',pin);
pin = bitDefine(11,-31,'a',pin);
%-------------3-----------------
pin = bitDefine(11,-13,'t',pin);
pin = bitDefine(11,-12,'a',pin);
pin = bitDefine(11,-11,'t',pin);
pin = bitDefine(11,-10,'a',pin);
pin = bitDefine(11,-9,'a',pin);
pin = bitDefine(11,-8,'t',pin);
%-------------1-----------------
pin = bitDefine(41,-47,'c',pin);
pin = bitDefine(41,-46,'a',pin);
pin = bitDefine(41,-45,'a',pin);
pin = bitDefine(41,-43,'t',pin);
pin = bitDefine(41,-42,'t',pin);
```

```
pin = bitDefine(41,-40,'a',pin);
pin = bitDefine(41,-39,'c',pin);
pin = bitDefine(41,-22,'g',pin);
pin = bitDefine(41,-18,'t',pin);
pin = bitDefine(41,-16,'c',pin);
pin = bitDefine(41,-8,'g',pin);
pin = bitDefine(41,-7,'c',pin);
pin = bitDefine(41,-6,'g',pin);
pin = bitDefine(41,-5,'c',pin);
pin = bitDefine(41,-4,'c',pin);
pin = bitDefine(41,-2,'c',pin);
pin = bitDefine(41,-1,'c',pin);
%------------------------------

%-----KANONAS 42 3 3 2----------
%-------------3-----------------
pin = bitDefine(42,-36,'t',pin);
pin = bitDefine(42,-35,'t',pin);
pin = bitDefine(42,-34,'g',pin);
pin = bitDefine(42,-33,'a',pin);
pin = bitDefine(42,-32,'c',pin);
pin = bitDefine(42,-31,'a',pin);
%-------------3-----------------
pin = bitDefine(42,-13,'t',pin);
pin = bitDefine(42,-12,'a',pin);
pin = bitDefine(42,-11,'t',pin);
pin = bitDefine(42,-10,'a',pin);
pin = bitDefine(42,-9,'a',pin);
pin = bitDefine(42,-8,'t',pin);
%-------------2-----------------
pin = bitDefine(42,-45,'a',pin);
pin = bitDefine(42,-44,'a',pin);
pin = bitDefine(42,-41,'a',pin);
%------------------------------

%-----KANONAS 43 3 3 3----------
%-------------3-----------------
pin = bitDefine(43,-36,'t',pin);
pin = bitDefine(43,-35,'t',pin);
pin = bitDefine(43,-34,'g',pin);
pin = bitDefine(43,-33,'a',pin);
pin = bitDefine(43,-32,'c',pin);
pin = bitDefine(43,-31,'a',pin);
%-------------3-----------------
pin = bitDefine(43,-13,'t',pin);
pin = bitDefine(43,-12,'a',pin);
pin = bitDefine(43,-11,'t',pin);
pin = bitDefine(43,-10,'a',pin);
pin = bitDefine(43,-9,'a',pin);
pin = bitDefine(43,-8,'t',pin);
```

```
%-------------3-----------------
pin = bitDefine(43,-49,'a',pin);
pin = bitDefine(43,-44,'t',pin);
pin = bitDefine(43,-27,'t',pin);
pin = bitDefine(43,-22,'a',pin);
pin = bitDefine(43,-18,'t',pin);
pin = bitDefine(43,-16,'t',pin);
pin = bitDefine(43,-15,'g',pin);
pin = bitDefine(43,-1,'a',pin);
%------------------------------

%-----KANONAS 44 3 3 4----------
%-------------3-----------------
pin = bitDefine(44,-36,'t',pin);
pin = bitDefine(44,-35,'t',pin);
pin = bitDefine(44,-34,'g',pin);
pin = bitDefine(44,-33,'a',pin);
pin = bitDefine(44,-32,'c',pin);
pin = bitDefine(44,-31,'a',pin);
%-------------3-----------------
pin = bitDefine(44,-13,'t',pin);
pin = bitDefine(44,-12,'a',pin);
pin = bitDefine(44,-11,'t',pin);
pin = bitDefine(44,-10,'a',pin);
pin = bitDefine(44,-9,'a',pin);
pin = bitDefine(44,-8,'t',pin);
%-------------4-----------------
pin = bitDefine(44,-45,'a',pin);
pin = bitDefine(44,-41,'a',pin);
pin = bitDefine(44,-28,'t',pin);
pin = bitDefine(44,-27,'t',pin);
pin = bitDefine(44,-23,'t',pin);
pin = bitDefine(44,-21,'a',pin);
pin = bitDefine(44,-20,'a',pin);
pin = bitDefine(44,-17,'t',pin);
pin = bitDefine(44,-15,'t',pin);
pin = bitDefine(44,-4,'a',pin);
%------------------------------

%-----KANONAS 45 3 4 1----------
%-------------3-----------------
pin = bitDefine(12,-36,'t',pin);
pin = bitDefine(12,-35,'t',pin);
pin = bitDefine(12,-34,'g',pin);
pin = bitDefine(12,-33,'a',pin);
pin = bitDefine(12,-32,'c',pin);
pin = bitDefine(12,-31,'a',pin);
%-------------4-----------------
pin = bitDefine(12,-12,'t',pin);
pin = bitDefine(12,-11,'a',pin);
```

```
pin = bitDefine(12,-7,'t',pin);
%-------------1-----------------
pin = bitDefine(45,-47,'c',pin);
pin = bitDefine(45,-46,'a',pin);
pin = bitDefine(45,-45,'a',pin);
pin = bitDefine(45,-43,'t',pin);
pin = bitDefine(45,-42,'t',pin);
pin = bitDefine(45,-40,'a',pin);
pin = bitDefine(45,-39,'c',pin);
pin = bitDefine(45,-22,'g',pin);
pin = bitDefine(45,-18,'t',pin);
pin = bitDefine(45,-16,'c',pin);
pin = bitDefine(45,-8,'g',pin);
pin = bitDefine(45,-7,'c',pin);
pin = bitDefine(45,-6,'g',pin);
pin = bitDefine(45,-5,'c',pin);
pin = bitDefine(45,-4,'c',pin);
pin = bitDefine(45,-2,'c',pin);
pin = bitDefine(45,-1,'c',pin);
%------------------------------

%-----KANONAS 46 3 4 2----------
%-------------3-----------------
pin = bitDefine(46,-36,'t',pin);
pin = bitDefine(46,-35,'t',pin);
pin = bitDefine(46,-34,'g',pin);
pin = bitDefine(46,-33,'a',pin);
pin = bitDefine(46,-32,'c',pin);
pin = bitDefine(46,-31,'a',pin);
%-------------4------------------
pin = bitDefine(46,-12,'t',pin);
pin = bitDefine(46,-11,'a',pin);
pin = bitDefine(46,-7,'t',pin);
%-------------2-----------------
pin = bitDefine(46,-45,'a',pin);
pin = bitDefine(46,-44,'a',pin);
pin = bitDefine(46,-41,'a',pin);
%------------------------------

%-----KANONAS 47 3 4 3----------
%-------------3-----------------
pin = bitDefine(47,-36,'t',pin);
pin = bitDefine(47,-35,'t',pin);
pin = bitDefine(47,-34,'g',pin);
pin = bitDefine(47,-33,'a',pin);
pin = bitDefine(47,-32,'c',pin);
pin = bitDefine(47,-31,'a',pin);
%-------------4------------------
pin = bitDefine(47,-12,'t',pin);
pin = bitDefine(47,-11,'a',pin);
```

```
pin = bitDefine(47,-7,'t',pin);
%-------------3-----------------
pin = bitDefine(47,-49,'a',pin);
pin = bitDefine(47,-44,'t',pin);
pin = bitDefine(47,-27,'t',pin);
pin = bitDefine(47,-22,'a',pin);
pin = bitDefine(47,-18,'t',pin);
pin = bitDefine(47,-16,'t',pin);
pin = bitDefine(47,-15,'g',pin);
pin = bitDefine(47,-1,'a',pin);
%------------------------------

%-----KANONAS 48 3 4 4----------
%-------------3-----------------
pin = bitDefine(48,-36,'t',pin);
pin = bitDefine(48,-35,'t',pin);
pin = bitDefine(48,-34,'g',pin);
pin = bitDefine(48,-33,'a',pin);
pin = bitDefine(48,-32,'c',pin);
pin = bitDefine(48,-31,'a',pin);
%-------------4------------------
pin = bitDefine(48,-12,'t',pin);
pin = bitDefine(48,-11,'a',pin);
pin = bitDefine(48,-7,'t',pin);
%-------------4-----------------
pin = bitDefine(48,-45,'a',pin);
pin = bitDefine(48,-41,'a',pin);
pin = bitDefine(48,-28,'t',pin);
pin = bitDefine(48,-27,'t',pin);
pin = bitDefine(48,-23,'t',pin);
pin = bitDefine(48,-21,'a',pin);
pin = bitDefine(48,-20,'a',pin);
pin = bitDefine(48,-17,'t',pin);
pin = bitDefine(48,-15,'t',pin);
pin = bitDefine(48,-4,'a',pin);
%------------------------------

%-----KANONAS 49 4 1 1----------
%-------------4-----------------
pin = bitDefine(13,-36,'t',pin);
pin = bitDefine(13,-35,'t',pin);
pin = bitDefine(13,-34,'g',pin);
pin = bitDefine(13,-33,'a',pin);
pin = bitDefine(13,-32,'c',pin);
%-------------1-----------------
pin = bitDefine(13,-14,'t',pin);
pin = bitDefine(13,-13,'a',pin);
pin = bitDefine(13,-12,'t',pin);
pin = bitDefine(13,-11,'a',pin);
pin = bitDefine(13,-10,'a',pin);
```

```
pin = bitDefine(13,-9,'t',pin);
%-------------1-----------------
pin = bitDefine(49,-47,'c',pin);
pin = bitDefine(49,-46,'a',pin);
pin = bitDefine(49,-45,'a',pin);
pin = bitDefine(49,-43,'t',pin);
pin = bitDefine(49,-42,'t',pin);
pin = bitDefine(49,-40,'a',pin);
pin = bitDefine(49,-39,'c',pin);
pin = bitDefine(49,-22,'g',pin);
pin = bitDefine(49,-18,'t',pin);
pin = bitDefine(49,-16,'c',pin);
pin = bitDefine(49,-8,'g',pin);
pin = bitDefine(49,-7,'c',pin);
pin = bitDefine(49,-6,'g',pin);
pin = bitDefine(49,-5,'c',pin);
pin = bitDefine(49,-4,'c',pin);
pin = bitDefine(49,-2,'c',pin);
pin = bitDefine(49,-1,'c',pin);
%------------------------------

%-----KANONAS 50 4 1 2----------
%-------------4-----------------
pin = bitDefine(50,-36,'t',pin);
pin = bitDefine(50,-35,'t',pin);
pin = bitDefine(50,-34,'g',pin);
pin = bitDefine(50,-33,'a',pin);
pin = bitDefine(50,-32,'c',pin);
%-------------1-----------------
pin = bitDefine(50,-14,'t',pin);
pin = bitDefine(50,-13,'a',pin);
pin = bitDefine(50,-12,'t',pin);
pin = bitDefine(50,-11,'a',pin);
pin = bitDefine(50,-10,'a',pin);
pin = bitDefine(50,-9,'t',pin);
%-------------2-----------------
pin = bitDefine(50,-45,'a',pin);
pin = bitDefine(50,-44,'a',pin);
pin = bitDefine(50,-41,'a',pin);
%------------------------------

%-----KANONAS 51 4 1 3----------
%-------------4-----------------
pin = bitDefine(51,-36,'t',pin);
pin = bitDefine(51,-35,'t',pin);
pin = bitDefine(51,-34,'g',pin);
pin = bitDefine(51,-33,'a',pin);
pin = bitDefine(51,-32,'c',pin);
%-------------1-----------------
pin = bitDefine(51,-14,'t',pin);
```

```
pin = bitDefine(51,-13,'a',pin);
pin = bitDefine(51,-12,'t',pin);
pin = bitDefine(51,-11,'a',pin);
pin = bitDefine(51,-10,'a',pin);
pin = bitDefine(51,-9,'t',pin);
%-------------3-----------------
pin = bitDefine(51,-49,'a',pin);
pin = bitDefine(51,-44,'t',pin);
pin = bitDefine(51,-27,'t',pin);
pin = bitDefine(51,-22,'a',pin);
pin = bitDefine(51,-18,'t',pin);
pin = bitDefine(51,-16,'t',pin);
pin = bitDefine(51,-15,'g',pin);
pin = bitDefine(51,-1,'a',pin);
%------------------------------

%-----KANONAS 52 4 1 4----------
%-------------4-----------------
pin = bitDefine(52,-36,'t',pin);
pin = bitDefine(52,-35,'t',pin);
pin = bitDefine(52,-34,'g',pin);
pin = bitDefine(52,-33,'a',pin);
pin = bitDefine(52,-32,'c',pin);
%-------------1-----------------
pin = bitDefine(52,-14,'t',pin);
pin = bitDefine(52,-13,'a',pin);
pin = bitDefine(52,-12,'t',pin);
pin = bitDefine(52,-11,'a',pin);
pin = bitDefine(52,-10,'a',pin);
pin = bitDefine(52,-9,'t',pin);
%-------------4-----------------
pin = bitDefine(52,-45,'a',pin);
pin = bitDefine(52,-41,'a',pin);
pin = bitDefine(52,-28,'t',pin);
pin = bitDefine(52,-27,'t',pin);
pin = bitDefine(52,-23,'t',pin);
pin = bitDefine(52,-21,'a',pin);
pin = bitDefine(52,-20,'a',pin);
pin = bitDefine(52,-17,'t',pin);
pin = bitDefine(52,-15,'t',pin);
pin = bitDefine(52,-4,'a',pin);
%------------------------------

%-----KANONAS 53 4 2 1----------
%-------------4-----------------
pin = bitDefine(14,-36,'t',pin);
pin = bitDefine(14,-35,'t',pin);
pin = bitDefine(14,-34,'g',pin);
pin = bitDefine(14,-33,'a',pin);
pin = bitDefine(14,-32,'c',pin);
```

```
%-------------2-----------------
pin = bitDefine(14,-13,'t',pin);
pin = bitDefine(14,-12,'a',pin);
pin = bitDefine(14,-10,'a',pin);
pin = bitDefine(14,-8,'t',pin);
%-------------1-----------------
pin = bitDefine(53,-47,'c',pin);
pin = bitDefine(53,-46,'a',pin);
pin = bitDefine(53,-45,'a',pin);
pin = bitDefine(53,-43,'t',pin);
pin = bitDefine(53,-42,'t',pin);
pin = bitDefine(53,-40,'a',pin);
pin = bitDefine(53,-39,'c',pin);
pin = bitDefine(53,-22,'g',pin);
pin = bitDefine(53,-18,'t',pin);
pin = bitDefine(53,-16,'c',pin);
pin = bitDefine(53,-8,'g',pin);
pin = bitDefine(53,-7,'c',pin);
pin = bitDefine(53,-6,'g',pin);
pin = bitDefine(53,-5,'c',pin);
pin = bitDefine(53,-4,'c',pin);
pin = bitDefine(53,-2,'c',pin);
pin = bitDefine(53,-1,'c',pin);
%------------------------------

%-----KANONAS 54 4 2 2----------
%-------------4-----------------
pin = bitDefine(54,-36,'t',pin);
pin = bitDefine(54,-35,'t',pin);
pin = bitDefine(54,-34,'g',pin);
pin = bitDefine(54,-33,'a',pin);
pin = bitDefine(54,-32,'c',pin);
%-------------2-----------------
pin = bitDefine(54,-13,'t',pin);
pin = bitDefine(54,-12,'a',pin);
pin = bitDefine(54,-10,'a',pin);
pin = bitDefine(54,-8,'t',pin);
%-------------2-----------------
pin = bitDefine(54,-45,'a',pin);
pin = bitDefine(54,-44,'a',pin);
pin = bitDefine(54,-41,'a',pin);
%------------------------------

%-----KANONAS 55 4 2 3----------
%-------------4-----------------
pin = bitDefine(55,-36,'t',pin);
pin = bitDefine(55,-35,'t',pin);
pin = bitDefine(55,-34,'g',pin);
pin = bitDefine(55,-33,'a',pin);
pin = bitDefine(55,-32,'c',pin);
```

```
%-------------2-----------------
pin = bitDefine(55,-13,'t',pin);
pin = bitDefine(55,-12,'a',pin);
pin = bitDefine(55,-10,'a',pin);
pin = bitDefine(55,-8,'t',pin);
%-------------3-----------------
pin = bitDefine(55,-49,'a',pin);
pin = bitDefine(55,-44,'t',pin);
pin = bitDefine(55,-27,'t',pin);
pin = bitDefine(55,-22,'a',pin);
pin = bitDefine(55,-18,'t',pin);
pin = bitDefine(55,-16,'t',pin);
pin = bitDefine(55,-15,'g',pin);
pin = bitDefine(55,-1,'a',pin);
%------------------------------

%-----KANONAS 56 4 2 4----------
%-------------4-----------------
pin = bitDefine(56,-36,'t',pin);
pin = bitDefine(56,-35,'t',pin);
pin = bitDefine(56,-34,'g',pin);
pin = bitDefine(56,-33,'a',pin);
pin = bitDefine(56,-32,'c',pin);
%-------------2-----------------
pin = bitDefine(56,-13,'t',pin);
pin = bitDefine(56,-12,'a',pin);
pin = bitDefine(56,-10,'a',pin);
pin = bitDefine(56,-8,'t',pin);
%-------------4-----------------
pin = bitDefine(56,-45,'a',pin);
pin = bitDefine(56,-41,'a',pin);
pin = bitDefine(56,-28,'t',pin);
pin = bitDefine(56,-27,'t',pin);
pin = bitDefine(56,-23,'t',pin);
pin = bitDefine(56,-21,'a',pin);
pin = bitDefine(56,-20,'a',pin);
pin = bitDefine(56,-17,'t',pin);
pin = bitDefine(56,-15,'t',pin);
pin = bitDefine(56,-4,'a',pin);
%------------------------------

%-----KANONAS 57 4 3 1----------
%-------------4-----------------
pin = bitDefine(15,-36,'t',pin);
pin = bitDefine(15,-35,'t',pin);
pin = bitDefine(15,-34,'g',pin);
pin = bitDefine(15,-33,'a',pin);
pin = bitDefine(15,-32,'c',pin);
%-------------3-----------------
pin = bitDefine(15,-13,'t',pin);
```

```
pin = bitDefine(15,-12,'a',pin);
pin = bitDefine(15,-11,'t',pin);
pin = bitDefine(15,-10,'a',pin);
pin = bitDefine(15,-9,'a',pin);
pin = bitDefine(15,-8,'t',pin);
%-------------1-----------------
pin = bitDefine(57,-47,'c',pin);
pin = bitDefine(57,-46,'a',pin);
pin = bitDefine(57,-45,'a',pin);
pin = bitDefine(57,-43,'t',pin);
pin = bitDefine(57,-42,'t',pin);
pin = bitDefine(57,-40,'a',pin);
pin = bitDefine(57,-39,'c',pin);
pin = bitDefine(57,-22,'g',pin);
pin = bitDefine(57,-18,'t',pin);
pin = bitDefine(57,-16,'c',pin);
pin = bitDefine(57,-8,'g',pin);
pin = bitDefine(57,-7,'c',pin);
pin = bitDefine(57,-6,'g',pin);
pin = bitDefine(57,-5,'c',pin);
pin = bitDefine(57,-4,'c',pin);
pin = bitDefine(57,-2,'c',pin);
pin = bitDefine(57,-1,'c',pin);
%------------------------------

%-----KANONAS 58 4 3 2----------
%-------------4-----------------
pin = bitDefine(58,-36,'t',pin);
pin = bitDefine(58,-35,'t',pin);
pin = bitDefine(58,-34,'g',pin);
pin = bitDefine(58,-33,'a',pin);
pin = bitDefine(58,-32,'c',pin);
%-------------3-----------------
pin = bitDefine(58,-13,'t',pin);
pin = bitDefine(58,-12,'a',pin);
pin = bitDefine(58,-11,'t',pin);
pin = bitDefine(58,-10,'a',pin);
pin = bitDefine(58,-9,'a',pin);
pin = bitDefine(58,-8,'t',pin);
%-------------2-----------------
pin = bitDefine(58,-45,'a',pin);
pin = bitDefine(58,-44,'a',pin);
pin = bitDefine(58,-41,'a',pin);
%------------------------------

%-----KANONAS 59 4 3 3----------
%-------------4-----------------
pin = bitDefine(58,-36,'t',pin);
pin = bitDefine(58,-35,'t',pin);
pin = bitDefine(58,-34,'g',pin);
```

```
pin = bitDefine(58,-33,'a',pin);
pin = bitDefine(58,-32,'c',pin);
%-------------3-----------------
pin = bitDefine(58,-13,'t',pin);
pin = bitDefine(58,-12,'a',pin);
pin = bitDefine(58,-11,'t',pin);
pin = bitDefine(58,-10,'a',pin);
pin = bitDefine(58,-9,'a',pin);
pin = bitDefine(58,-8,'t',pin);
%-------------3-----------------
pin = bitDefine(58,-49,'a',pin);
pin = bitDefine(58,-44,'t',pin);
pin = bitDefine(58,-27,'t',pin);
pin = bitDefine(58,-22,'a',pin);
pin = bitDefine(58,-18,'t',pin);
pin = bitDefine(58,-16,'t',pin);
pin = bitDefine(58,-15,'g',pin);
pin = bitDefine(58,-1,'a',pin);
%------------------------------

%-----KANONAS 60 4 3 4----------
%-------------4-----------------
pin = bitDefine(60,-36,'t',pin);
pin = bitDefine(60,-35,'t',pin);
pin = bitDefine(60,-34,'g',pin);
pin = bitDefine(60,-33,'a',pin);
pin = bitDefine(60,-32,'c',pin);
%-------------3-----------------
pin = bitDefine(60,-13,'t',pin);
pin = bitDefine(60,-12,'a',pin);
pin = bitDefine(60,-11,'t',pin);
pin = bitDefine(60,-10,'a',pin);
pin = bitDefine(60,-9,'a',pin);
pin = bitDefine(60,-8,'t',pin);
%-------------4-----------------
pin = bitDefine(60,-45,'a',pin);
pin = bitDefine(60,-41,'a',pin);
pin = bitDefine(60,-28,'t',pin);
pin = bitDefine(60,-27,'t',pin);
pin = bitDefine(60,-23,'t',pin);
pin = bitDefine(60,-21,'a',pin);
pin = bitDefine(60,-20,'a',pin);
pin = bitDefine(60,-17,'t',pin);
pin = bitDefine(60,-15,'t',pin);
pin = bitDefine(60,-4,'a',pin);
%------------------------------

%-----KANONAS 61 4 4 1----------
%-------------4-----------------
pin = bitDefine(16,-36,'t',pin);
```

```
pin = bitDefine(16,-35,'t',pin);
pin = bitDefine(16,-34,'g',pin);
pin = bitDefine(16,-33,'a',pin);
pin = bitDefine(16,-32,'c',pin);
%-------------4------------------
pin = bitDefine(16,-12,'t',pin);
pin = bitDefine(16,-11,'a',pin);
pin = bitDefine(16,-7,'t',pin);
%-------------1-----------------
pin = bitDefine(61,-47,'c',pin);
pin = bitDefine(61,-46,'a',pin);
pin = bitDefine(61,-45,'a',pin);
pin = bitDefine(61,-43,'t',pin);
pin = bitDefine(61,-42,'t',pin);
pin = bitDefine(61,-40,'a',pin);
pin = bitDefine(61,-39,'c',pin);
pin = bitDefine(61,-22,'g',pin);
pin = bitDefine(61,-18,'t',pin);
pin = bitDefine(61,-16,'c',pin);
pin = bitDefine(61,-8,'g',pin);
pin = bitDefine(61,-7,'c',pin);
pin = bitDefine(61,-6,'g',pin);
pin = bitDefine(61,-5,'c',pin);
pin = bitDefine(61,-4,'c',pin);
pin = bitDefine(61,-2,'c',pin);
pin = bitDefine(61,-1,'c',pin);
%------------------------------

%-----KANONAS 62 4 4 2----------
%-------------4-----------------
pin = bitDefine(62,-36,'t',pin);
pin = bitDefine(62,-35,'t',pin);
pin = bitDefine(62,-34,'g',pin);
pin = bitDefine(62,-33,'a',pin);
pin = bitDefine(62,-32,'c',pin);
%-------------4------------------
pin = bitDefine(62,-12,'t',pin);
pin = bitDefine(62,-11,'a',pin);
pin = bitDefine(62,-7,'t',pin);
%-------------2-----------------
pin = bitDefine(58,-45,'a',pin);
pin = bitDefine(58,-44,'a',pin);
pin = bitDefine(58,-41,'a',pin);
%------------------------------

%-----KANONAS 63 4 4 3----------
%-------------4-----------------
pin = bitDefine(63,-36,'t',pin);
pin = bitDefine(63,-35,'t',pin);
pin = bitDefine(63,-34,'g',pin);
```

```
pin = bitDefine(63,-33,'a',pin);
pin = bitDefine(63,-32,'c',pin);
%-------------4------------------
pin = bitDefine(63,-12,'t',pin);
pin = bitDefine(63,-11,'a',pin);
pin = bitDefine(63,-7,'t',pin);
%-------------3-----------------
pin = bitDefine(63,-49,'a',pin);
pin = bitDefine(63,-44,'t',pin);
pin = bitDefine(63,-27,'t',pin);
pin = bitDefine(63,-22,'a',pin);
pin = bitDefine(63,-18,'t',pin);
pin = bitDefine(63,-16,'t',pin);
pin = bitDefine(63,-15,'g',pin);
pin = bitDefine(63,-1,'a',pin);
%------------------------------

%-----KANONAS 64 4 4 4----------
%-------------4-----------------
pin = bitDefine(64,-36,'t',pin);
pin = bitDefine(64,-35,'t',pin);
pin = bitDefine(64,-34,'g',pin);
pin = bitDefine(64,-33,'a',pin);
pin = bitDefine(64,-32,'c',pin);
%-------------4------------------
pin = bitDefine(64,-12,'t',pin);
pin = bitDefine(64,-11,'a',pin);
pin = bitDefine(64,-7,'t',pin);
%-------------4-----------------
pin = bitDefine(64,-45,'a',pin);
pin = bitDefine(64,-41,'a',pin);
pin = bitDefine(64,-28,'t',pin);
pin = bitDefine(64,-27,'t',pin);
pin = bitDefine(64,-23,'t',pin);
pin = bitDefine(64,-21,'a',pin);
pin = bitDefine(64,-20,'a',pin);
pin = bitDefine(64,-17,'t',pin);
pin = bitDefine(64,-15,'t',pin);
pin = bitDefine(64,-4,'a',pin);
%
%-------------------------------------------------
%-------------------------------------------------
%-------------------------------------------------
dlmwrite(strcat(fileid,'Wc.txt'),pin,' ');
pin = [];
for i = 1:n,
   for j = 1:q,
      pin(i,j) = 0.3;
   end
end
```

```
dlmwrite(strcat(fileid,'Ws.txt'),pin,' ');
%GIA TA VC && VS
pin = [];
for j = 1:q,
   for k = 1:p,
      pin(j,k) = 0.5;
      if(k==1)
         pin(j,k) = 1.0;
      elseif(k==2)
         pin(j,k) = 0.0;
      end
   end
end
dlmwrite(strcat(fileid,'Vc.txt'),pin,' ');
pin = [];
for j = 1:q,
   for k = 1:p,
      pin(j,k) = 0.4;
   end
end
dlmwrite(strcat(fileid,'Vs.txt'),pin,' ');

%*************************************************
function array = bitDefine(kanonas, bitBegin, gramma, pinakas)

array = pinakas;
bitBegin = (bitBegin+50)*4+1;
if(strcmp(gramma,'a'))
   array(bitBegin:bitBegin+3,kanonas) = [1;0;0;0];
elseif(strcmp(gramma,'c'))
   array(bitBegin:bitBegin+3,kanonas) = [0;1;0;0];
elseif(strcmp(gramma,'g'))
   array(bitBegin:bitBegin+3,kanonas) = [0;0;1;0];
elseif(strcmp(gramma,'t'))
   array(bitBegin:bitBegin+3,kanonas) = [0;0;0;1];
end
```