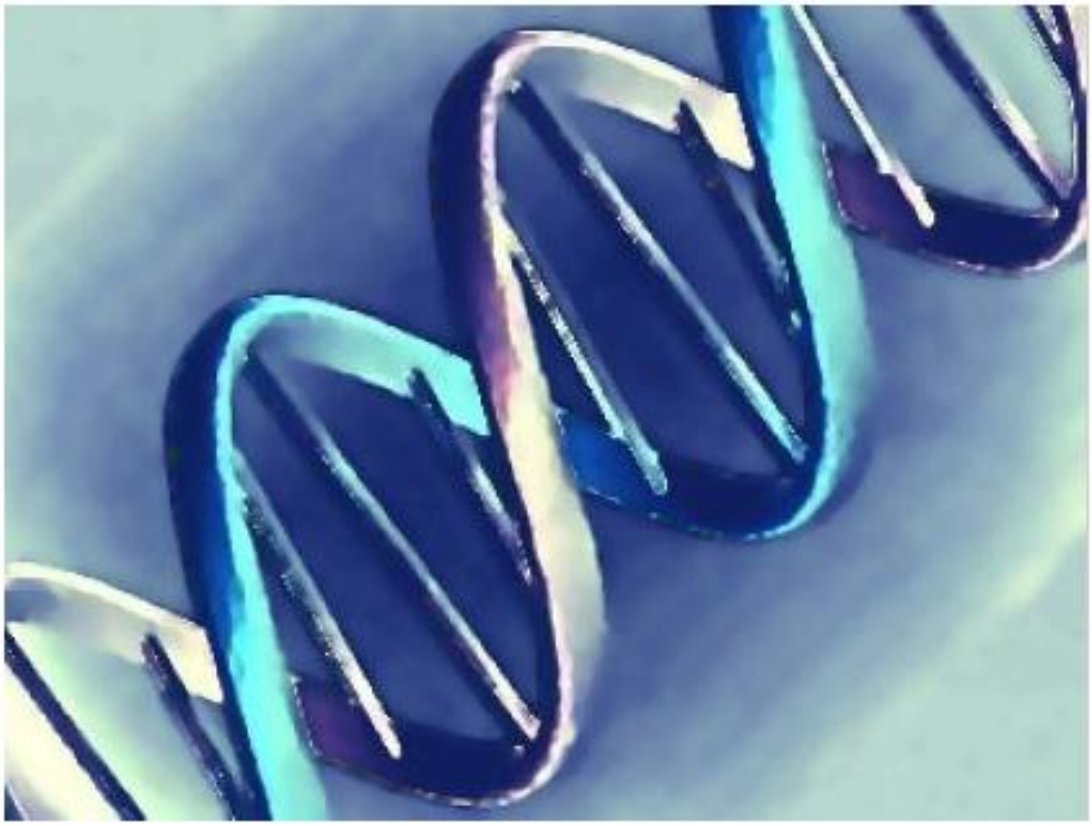
Τ.Ε.Ι. ΗΠΕΙΡΟΥ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία : Εκπαίδευση Τεχνητού Νευρωνικού Δικτύου με χρήση γενετικού αλγορίθμου



Μπαλατσός Νικόλαος ΑΜ : 10109

Επιβλέπων Καθηγητής : Ιωάννης Τσούλος

ΑΡΤΑ , 2014

Περιεχόμενα Εργασίας

Πίνακας Εικόνων

1. Εισαγωγή

1.1 Σκοπός της εργασίας

Σκοπός της εργασίας αυτής είναι η μελέτη,η ανάλυση και η αξιολόγηση των τεχνητών νευρωνικών δικτύων καθώς και η ανάλυση και επεξήγηση των γενετικών αλγορίθμων. Αυτό πραγματοποιείται μέσω μιας ιστορικής αναδρομής καθώς και η επεξήγηση των διαφόρων παραλλαγών και ιδιοτήτων τόσο των τεχνητών νευρωνικών δικτύων τόσο και των γενετικών αλγορίθμων. Επίσης παρουσιάζονται τα βήματα που χρειάζονται για την κατασκευή τους ,όπως επίσης τα διάφορα πλεονεκτήματα και μειονεκτήματα τους και οι διάφορες εφαρμογές στον επιστημονικό τομέα και των δύο. Τέλος δημιουργήθηκε μια εφαρμογή για την απόδειξη του πώς οι δύο αυτές τεχνικές μπορούν να συνδυαστούν μεταξύ τους και πρακτική αναφορά της θεωρίας.

1.2 Προβλήματα που επιλύει

Τα τεχνητά νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν σχεδόν σε όποια κατάσταση ισχύει μια σχέση μεταξύ μεταβλητών πρόβλεψης (ανεξάρτητες εισροές) και προβλεπόμενες μεταβλητές (εξαρτημένες εκροές) . Μερικά παραδείγματα προβλημάτων στα οποία η εφαρμογή των τεχνητών νευρωνικών δικτύων είναι:

- Ιατρική. Τα νευρωνικά δίκτυα είναι ιδανικά για την αναγνώριση ασθενειών καθώς δεν χρειάζεται να δημιουργηθεί κάποιο μαθησιακό μοντέλο για την εύρεση της νόσου. Η εκπαίδευση τους γίνεται μέσω δειγμάτων που είναι αντιπροσωπευτικά του συνόλου παραλλαγής της νόσου. Τα δείγματα προέρχονται από αποτελέσματα εξετάσεων και δείκτες που θεωρούνται ότι συσχετίζονται με τη νόσο που πρέπει να ανιχνευθεί. Ασφαλώς για όσο πιο αξιόπιστο και αποτελεσματικό δίκτυο χρειάζονται πολύ προσεχτικά επιλεγμένα δείγματα.
- Χρηματιστηριακές προβλέψεις. Οι διακυμάνσεις των τιμών των μετοχών και των χρηματιστηριακών δεικτών είναι ακόμα ένα παράδειγμα ενός πολύπλοκου,πολυδιάστατου και ντετερμινιστικού φαινομένου. Τα νευρωνικά δίκτυα χρησιμοποιούνται από πολλούς

τεχνικούς αναλυτές για προβλέψεις σχετικά με τις τιμές των μετοχών.

- Πιστωτική ανάθεση. Μια ποικιλία από κομμάτια πληροφοριών τα οποία είναι συνήθως γνωστά για ένα απαιτούμενο δάνειο,όπως για παράδειγμα η ηλικία του αιτούντος ,η εκπαίδευση,το επάγγελμα και άλλα διαθέσιμα στοιχεία. Τα νευρωνικά δίκτυα εδώ μπορούν να χρησιμοποιηθούν για ανάλυση ιστορικών δεδομένων ώστε να χαρακτηρίζει του αιτούντες ως υψηλού ή χαμηλού κινδύνου.
- Παρακολούθηση της κατάστασης των μηχανημάτων. Τα νευρωνικά δίκτυα μπορούν να συμβάλλουν στη μείωση του κόστους με τον προγραμματισμό προληπτικής συντήρησης των μηχανημάτων. Ένα νευρωνικό δίκτυο μπορεί να εκπαιδευτεί ώστε να ώστε από τους ήχους τους οποίους παράγει μια μηχανή να μπορεί να διακρίνει είτε αν λειτουργεί κανονικά είτε βρίσκεται στα πρόθυρα να εμφανίσει κάποια δυσλειτουργία.
- Συστήματα διαχείρισης κινητήρα. Τα νευρωνικά δίκτυα έχουν χρησιμοποιηθεί για την ανάλυση των εισροών που δέχονται οι αισθητήρες ενός κινητήρα. Το νευρωνικό ελέγχει μια ποικιλία παραμέτρων με τις οποίες λειτουργεί ο κινητήρας προκειμένου να επιτευχθεί ένας συγκεκριμένος στόχος,όπως η ελαχιστοποίηση της κατανάλωσης των καυσίμων.
- Αμυντικές λειτουργίες. Τα νευρωνικά δίκτυα χρησιμοποιούνται για εφαρμογές όπως η πλοήγηση όπλων,ψηφιακή επεξεργασία σήματος όπως και λειτουργία σόναρ και ραντάρ. [4]

Επίσης πολλές εφαρμογές έχουν ήδη υλοποιηθεί και αποτελούν εμπορικά προιόντα όπως

- Αναγνώριση των ομιλητών στις επικοινωνίες
- Διάγνωση της ηπατίτιδας
- Ανάκτηση των τηλεπικοινωνιών από ελαττωματικό λογισμικό
- Ερμηνεία των κινέζικων λέξεων
- Ανίχνευση υποθαλάσσιων ναρκών
- Ανάλυση σύστασης
- Αναγνώριση τρισδιάστατων αντικειμένων

- Αναγνώριση χειρόγραφων λέξεων
- Αναγνώριση προσώπου

1.3 Παρόμοια προγράμματα

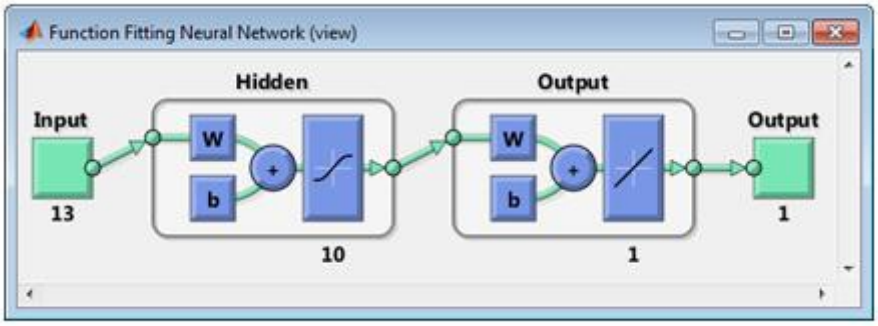
Ένα από τα καλύτερα και πιο γνωστά λογισμικά που χρησιμοποιούνται για την εκπαίδευση τεχνητών νευρωνικών δικτύων είναι το Neural Network Toolbox της Matlab. Το συγκεκριμένο λογισμικό προσφέρει μεγάλη ευελιξία καθώς προσφέρει μια πολύ μεγάλη εργαλειοθήκη με την οποία επιτρέπει στο χρήστη να σχεδιάσει, να εκπαιδεύσει, να απεικονίσει και να προσομοιώσει νευρωνικά δίκτυα.

Το neural network toolbox μπορεί να χρησιμοποιηθεί για διάφορες εφαρμογές όπως

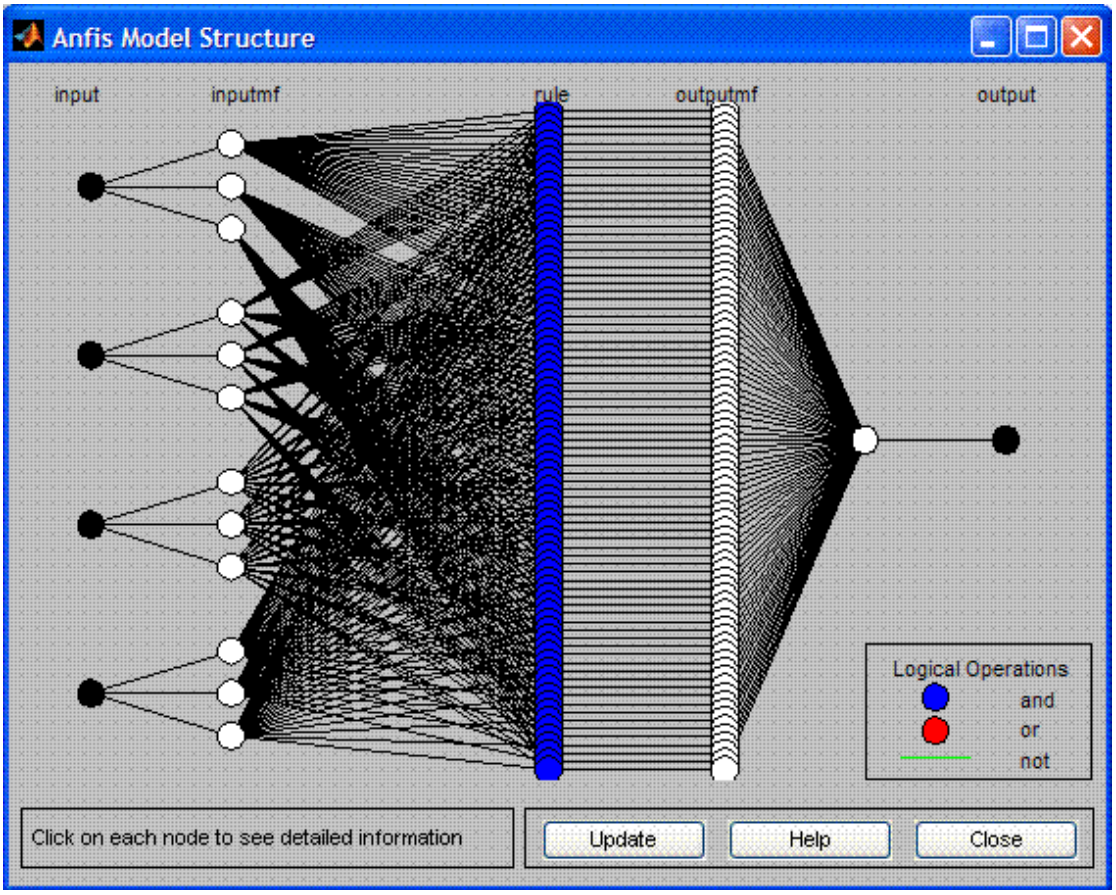
- Αξιολόγηση και ταξινόμηση δεδομένων
- Αναγνώριση προτύπων
- Ομαδοποίηση
- Πρόβλεψη χρονοσειρών
- Μοντελοποίηση και έλεγχος συστήματος

Τα πιο σημαντικά πλεονεκτήματα του neural network toolbox είναι η δυναμική σχεδίαση καθώς επιτρέπει στο χρήστη να βλέπει το τρόπο που είναι σχεδιασμένο το δίκτυο ,δηλαδή τους κόμβους επεξεργασίας και τα κρυμμένα επίπεδα, καθώς επίσης και γραφήματα με την επίδοση του δικτύου.

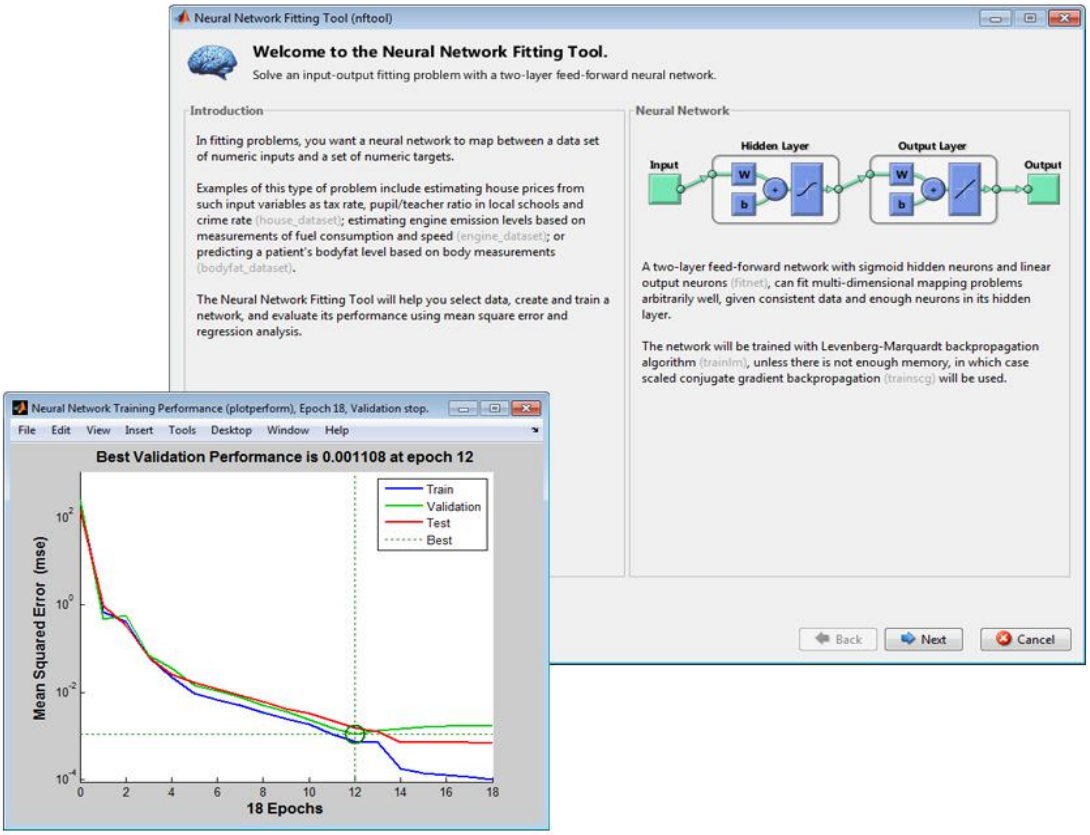
Ακολουθούν εικόνες από το neural network toolbox για την καλύτερη κατανόηση των όσων ειπώθηκαν προηγουμένως. [5]



Εικόνα 1:Παράδειγμα απεικόνισης σε Matlab



Εικόνα 2:Απεικόνιση αρχιτεκτονικής σε Matlab



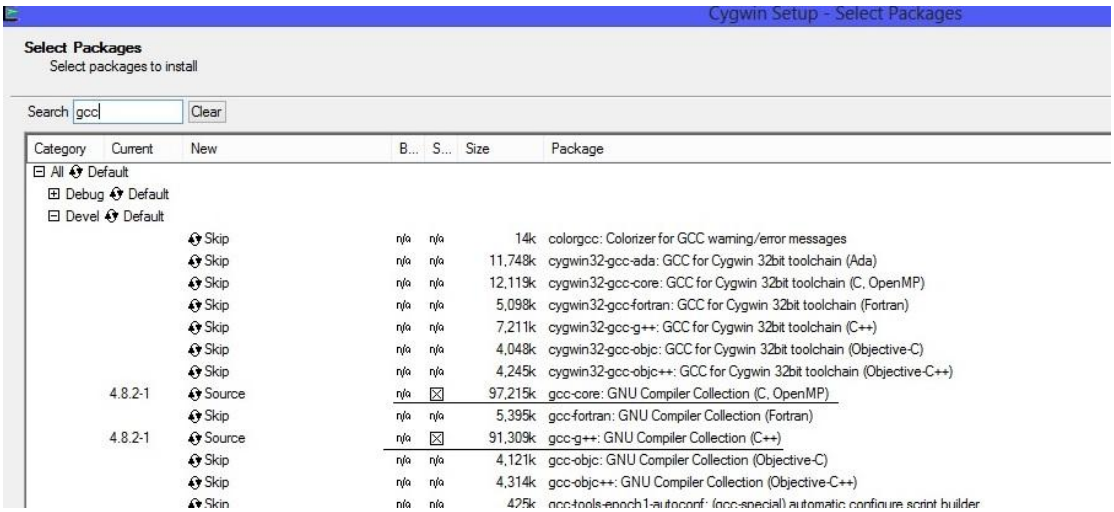
Εικόνα 3:Γράφημα επίδοσης του δικτύου σε Matlab

1.4 Χρησιμοποιούμενο λογισμικό

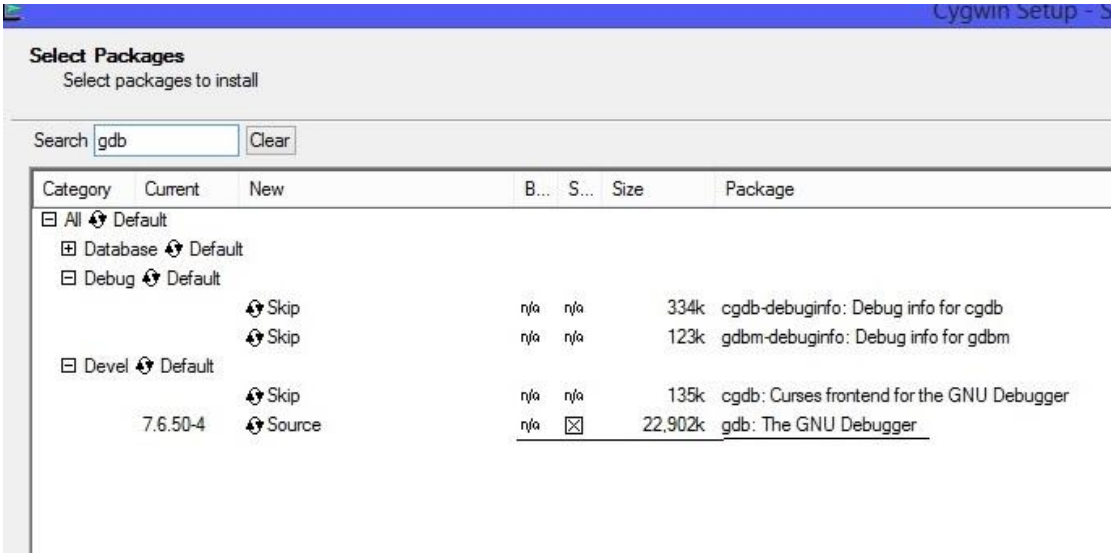
Για τον προγραμματισμό της εφαρμογής σε αυτήν την εργασία χρησιμοποιήθηκε το λογισμικό netbeans και συγκεκριμένα η έκδοση 7.4. Το netbeans είναι ένα προγραμματιστικό περιβάλλον γραμμένο στην προγραμματιστική γλώσσα java και λειτουργεί στα περισσότερα λογισμικά συμπεριλαμβανομένων τα Windows,Linux και Solaris. Παρότι το netbeans είναι γραμμένο κυρίως για java μπορεί να χρησιμοποιηθεί και να δημιουργηθούν προγράμματα και σε άλλες γλώσσες όπως HTML,PHP και ,όπως και η εφαρμογή που αναπτύχθηκε για την συγκεκριμένη εργασία, C/C++.

Για την εγκατάσταση του netbeans αρχικά χρειάζεται να εγκατασταθεί η τρέχουσα έκδοση Java Platform (JDK) που διατίθεται δωρεάν από την ιστοσελίδα της Oracle. Στην συνέχεια απλά γίνεται εγκατάσταση και του netbeans στον υπολογιστή. Κατά την διάρκεια της εγκατάστασης γίνεται αντιστοίχηση του JDK με το περιβάλλον netbeans.

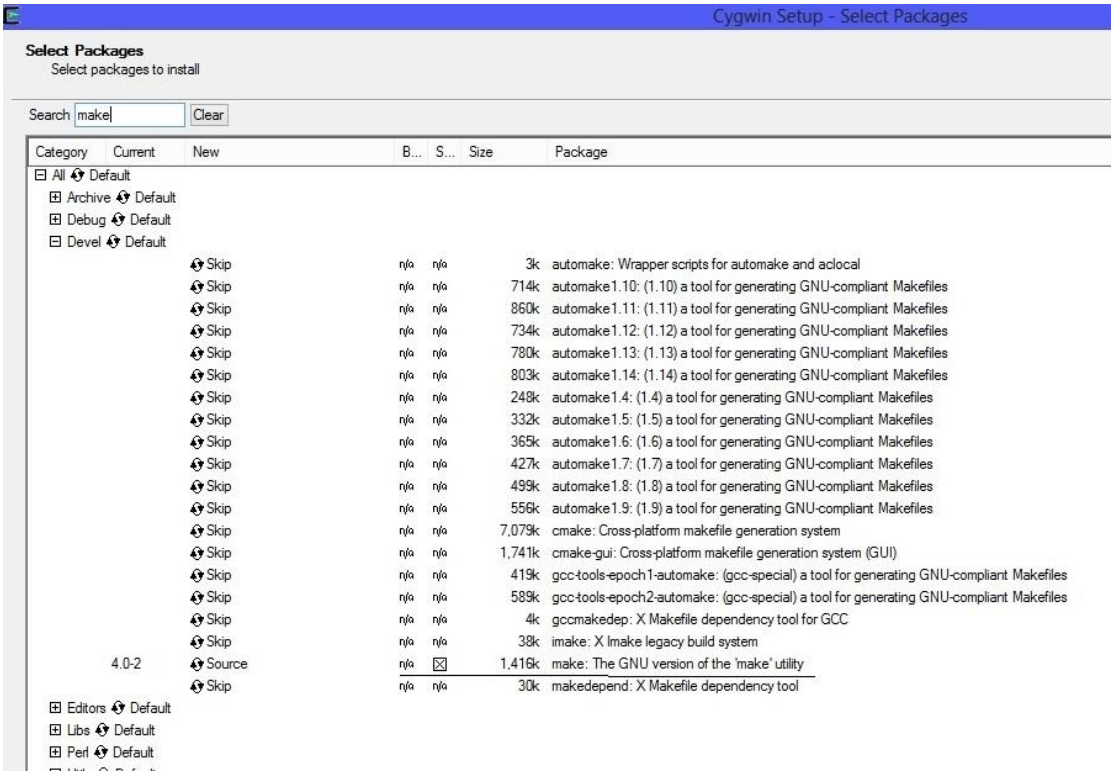
Για να λειτουργήσει όμως σωστά το λογισμικό χρειάζεται και τους κατάλληλους compilers. Αρχικά χρειάζεται να γίνει εγκατάσταση του cygwin το οποίο είναι μια πλατφόρμα τύπου Linux που χρησιμοποιείται στα windows. Κατά την εγκατάσταση του πρέπει να επιλεγούν και οι complilers οι οποίοι χρειάζονται. Για την εφαρμογή που αναπτύχθηκε χρειάζονται οι gcc-core,gcc-g++,gdb και make. Στο μενού που επιλέγονται οι compilers το search button βοηθάει για να βρεθούν ευκολότερα.



Εικόνα 4:Εγκατάσταση των gcc-core και gcc-g++ μέσω cygwin



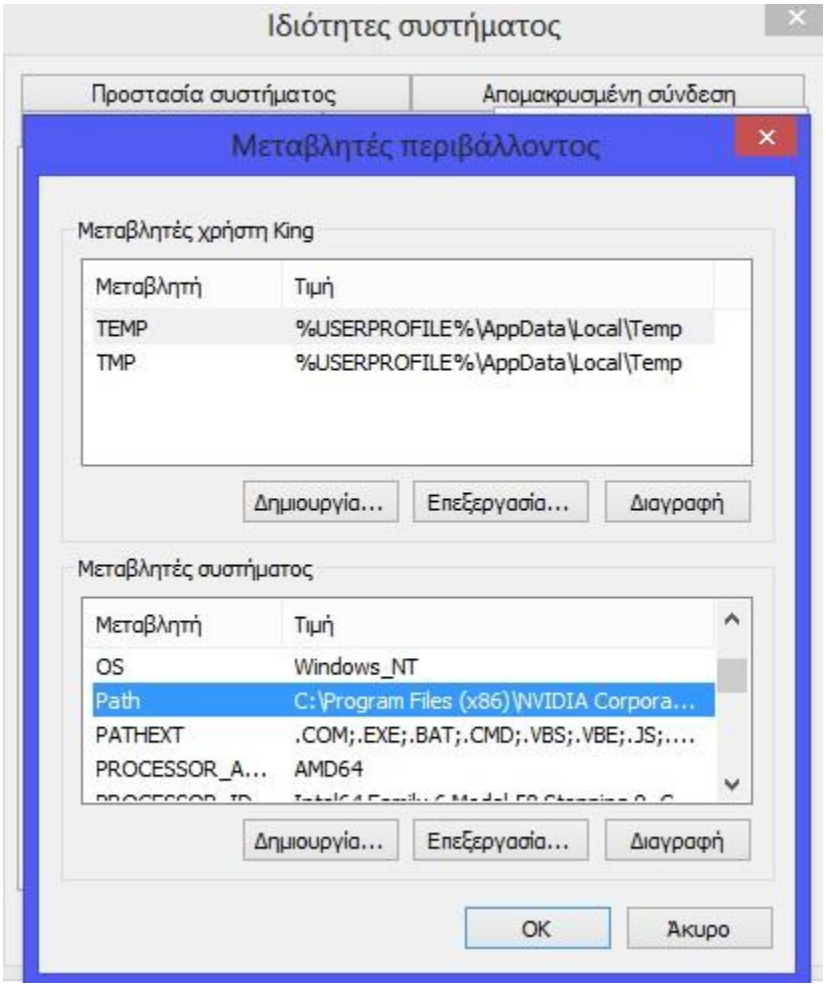
Εικόνα 5:Εγκατάσταση του gbd μέσω cygwin



Εικόνα 6:Εγκατάσταση της make μέσω cygwin

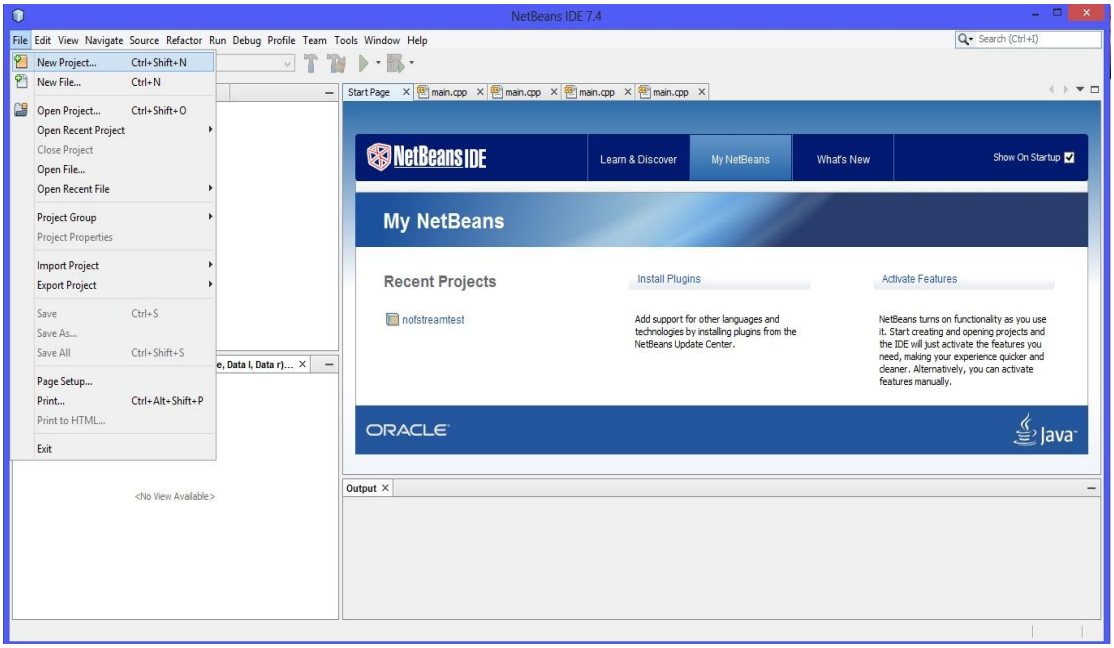
Τέλος για να ολοκληρωθεί η εγκατάσταση στον πίνακα ελέγχου (control panel) πρέπει να τροποποιηθούν οι μεταβλητές περιβάλλοντος (environment variables) ώστε να συμπεριληφθούν οι νέοι compilers. Για

να γίνει αυτό πρέπει στην μεταβλητή Path να προστεθεί στο τέλος της μεταβλητής το path για τον φάκελο που έχει εγκατασταθεί το cygwin.Εξ ορισμού το path είναι C:\cygwin\bin.



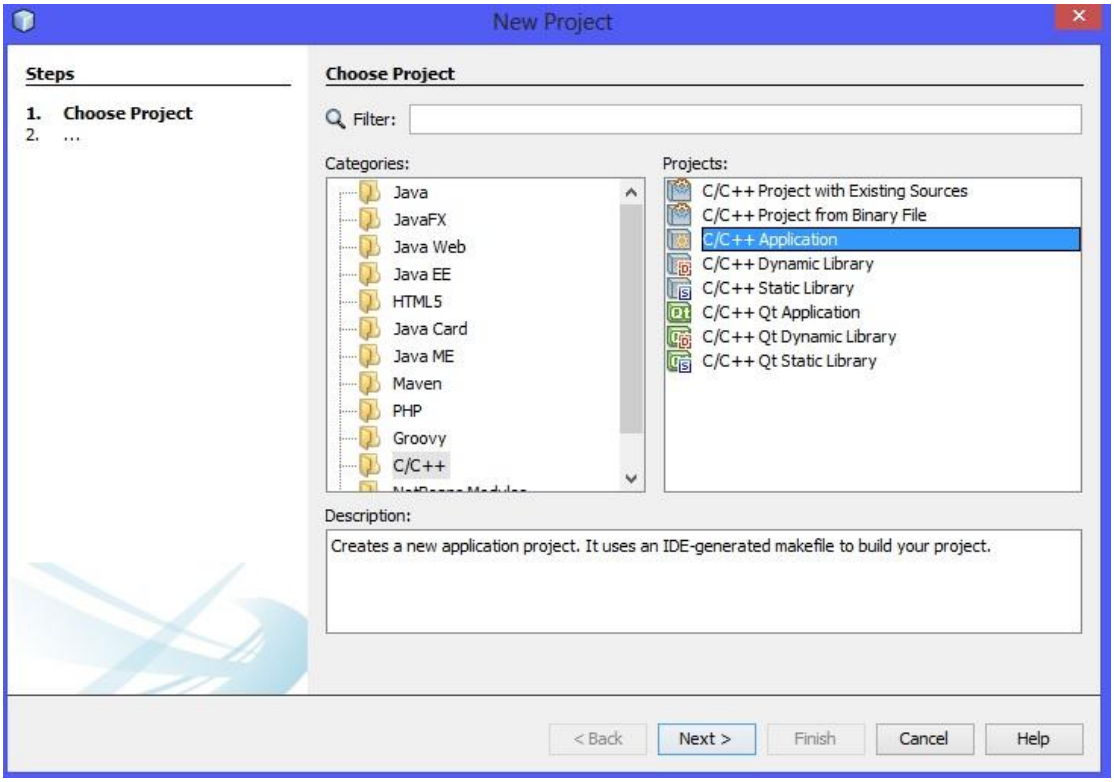
Εικόνα 7:Προσθήκη complilers στο σύστημα

Για να δημιουργηθεί η εφαρμογή στο netbeans στην αρχική σελίδα πρέπει να γίνει δημιουργία νέας εφαρμογής κάνοντας click στο openproject button όπως φαίνεται στην εικόνα παρακάτω



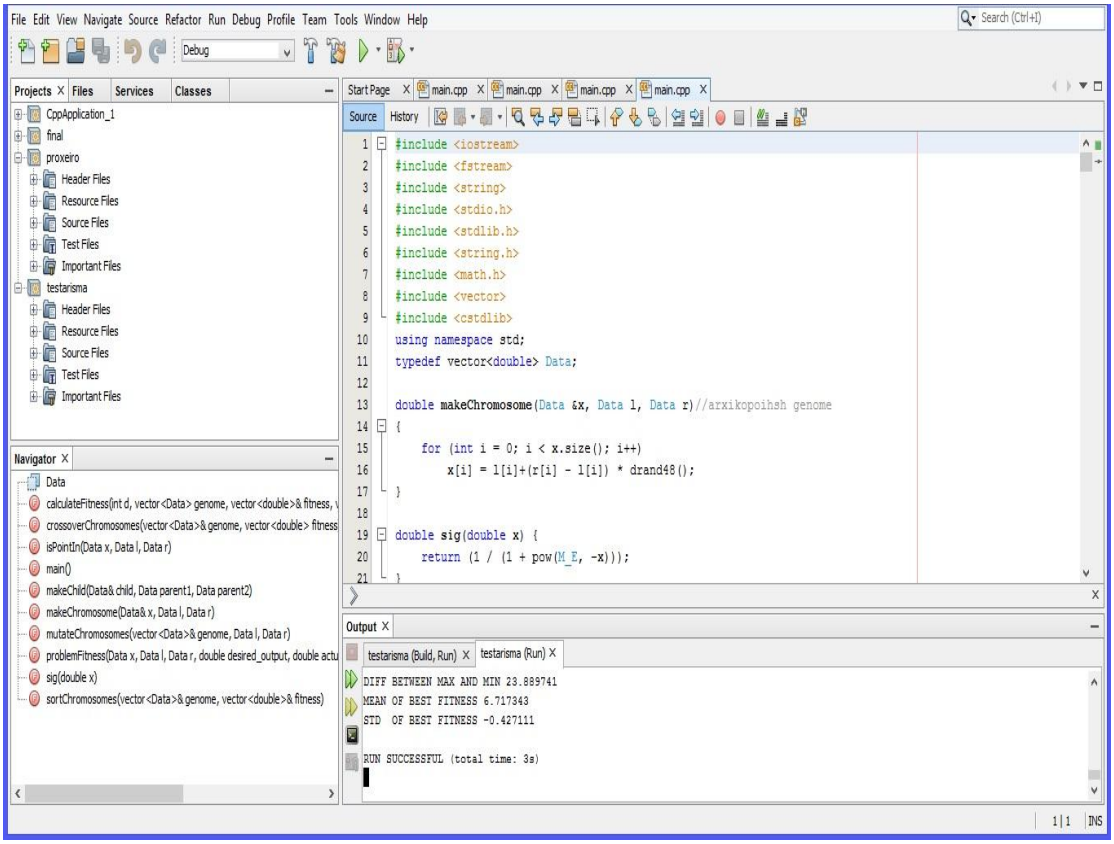
Εικόνα 8:Δημιουργία νέου project

και στην συνέχεια να επιλεγεί η γλώσσα και ο τύπος της εφαρμογής. Η εφαρμογή που αναπτύχθηκε για την εργασία είναι σε γλώσσα C++ και είναι τύπου application.



Εικόνα 9:Δημιουργία νέου application

Κλείνοντας στην εικόνα παρακάτω φαίνεται το περιβάλλον netbeans. Στην πάνω αριστερή γωνία φαίνονται όλα τα ανοιχτά project στην κάτω αριστερή γωνία τα περιεχόμενα και συστατικά του τρέχοντας project στην πάνω δεξιά γωνία φαίνεται ο κώδικας και τέλος στην κάτω δεξιά γωνία βρίσκονται τα αποτελέσματα του τρέχοντος project.



Εικόνα 10:Περιβάλλον netbeans

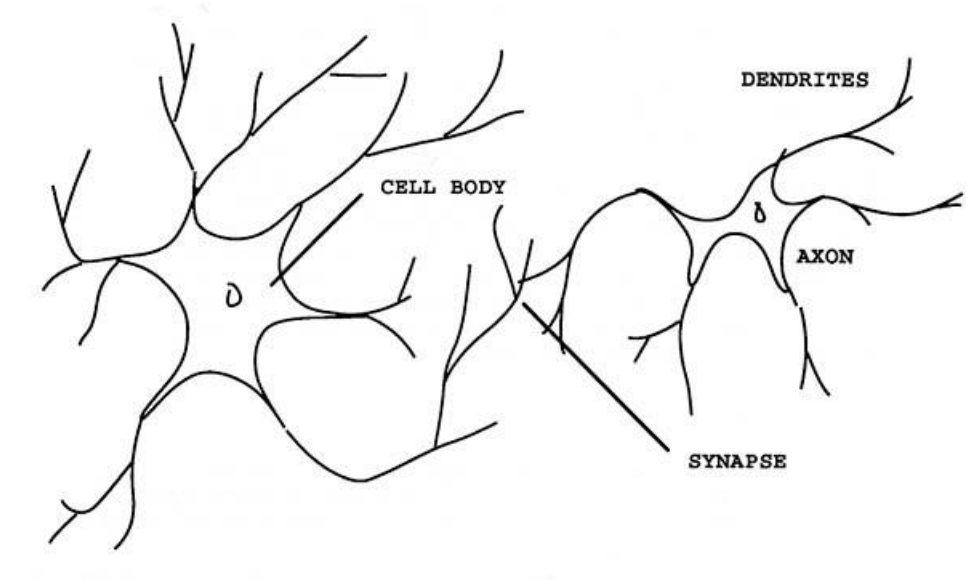
2. Τεχνητά νευρωνικά δίκτυα

2.1 Ιστορία των νευρωνικών δικτύων

Με την εμφάνιση των πρώτων υπολογιστικών μηχανών παρατηρήθηκε ομοιότητα στις λειτουργίες τους με τις υπολογιστικές διαδικασίες που λαμβάνουν χώρα στον ανθρώπινο εγκέφαλο. Ακόμα και σήμερα ο ανθρώπινος εγκέφαλος παρομοιάζεται σαν ένας πολύπλοκος υπολογιστής. Μπορεί και πραγματοποιεί αντίστοιχες διαδικασίες με αυτές που πειραματικά γίνονται σε υπολογιστές όπως η αναγνώριση προτύπων σε πολύ μικρότερο χρόνο και πολύ πιο αποτελεσματικά από αυτούς. Αυτά τα χαρακτηριστικά καθώς και η ικανότητα που

παρουσιάζει ο ανθρώπινος εγκέφαλος να απομνημονεύει και να μαθαίνει αποτέλεσε κίνητρο για τους επιστήμονες να κατασκευάσουν μοντέλα και αλγορίθμους που προσομοιάζουν την λειτουργία του ανθρώπινου εγκεφάλου.

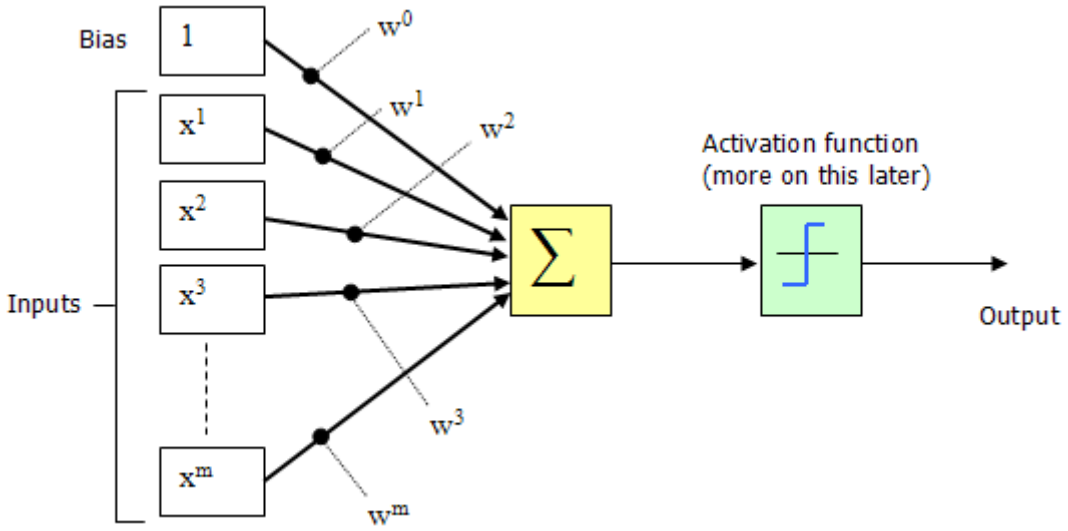
Ο φλοιός του ανθρώπινου εγκεφάλου αποτελείται από δισεκατομμύρια νευρώνες οι οποίοι με τη σειρά τους αποτελούνται από τρισεκατομμύρια συνάψεις. Υπολογίζεται ότι οι νευρώνες είναι διατεταγμένοι σε 1000 δομικές περιοχές με την κάθε μια από αυτές να αποτελείται από 500 περίπου νευρωνικά δίκτυα. Ο ανθρώπινος εγκέφαλος επιδεικνύει εξαιρετικές επιδόσεις στην ταυτόχρονη επίλυση διαφόρων προβλημάτων κάτι το οποίο επιτυγχάνεται διαμοιράζοντας τα προβλήματα σε διάφορα τμήματα του. Η επιστήμη της υπολογιστικής νοημοσύνης προσπαθεί να κατασκευάσει ένα αντίστοιχο μοντέλο παρότι δεν είναι ακόμη δυνατό λόγω των περιορισμών σε μνήμη και υπολογιστικής δύναμης των υπολογιστών. Παρόλα αυτά η επίλυση διαφόρων προβλημάτων με τη χρήση τεχνητών νευρωνικών δικτύων είναι δυνατή η επίλυση διαφόρων προβλημάτων και με συνεχόμενη έρευνα η επίλυση ακόμα πιο πολύπλοκων προβλημάτων είναι εφικτή.



Εικόνα 11:Ο βιολογικός τεχνητός νευρώνας

Η έρευνα πάνω σε τεχνητά νευρωνικά δίκτυα ουσιαστικά ξεκίνησε το 1949 όταν ο Donnald Hebb έγραψε κάποιες θεωρίες για μάθηση τα οποία άρχισαν και να εφαρμόζονταν. Στα τέλη της δεκαετίας του '50

αναπτύχθηκε το μοντέλο τεχνητού μοντέλου Perceptron το οποίο όμως το 1969 οι Minsky και Pappert απέδειξαν άχρηστο όταν αποτελείται από μόνο ένα επίπεδο για ένα μεγάλο πλήθος προβλημάτων. Αυτό είχε ως αποτέλεσμα την ελάττωση του επιστημονικού ενδιαφέροντος πάνω στα τεχνητά νευρωνικά δίκτυα παρότι το πρόβλημα που παρουσιάστηκε στο μοντέλο του Perceptron μπορούσε να αντιμετωπισθεί με την προσθήκη περισσότερων επιπέδων. Παρόλα αυτά οι έρευνες πάνω σε αυτά συνεχίστηκαν και μετά τα τέλη της δεκαετίας του '80 είχα αναπτυχθεί πολλές νέες και αποτελεσματικές τεχνικές που μπορούσαν να υποστηριχθούν και από την υπολογιστική ανάπτυξη της εποχής. [3]



Εικόνα 12:Το μοντέλο percerptron με ένα επίπεδο

2.2 Μηχανική Μάθηση

Η μηχανική μάθηση είναι μια από τις πιο σημαντικές λειτουργίες των τεχνητών νευρωνικών δικτύων. Με τον όρο μάθηση εννοείτε η ικανότητα να μαθαίνουν από το περιβάλλον και να βελτιώνουν την απόδοση τους μέσω της εκπαίδευσης. Η αλλαγή στις τιμές των συναπτικών βαρών του δικτύου είναι ουσιαστικά ο τρόπος για να μαθαίνει το νευρωνικό δίκτυο σχετικά με το περιβάλλον του. Ιδανική θεωρείτε η περίπτωση όπου στο τέλος κάθε επανάληψης της διαδικασίας της μάθησης το δίκτυο έχει αποκτήσει περισσότερη γνώση πάνω στο περιβάλλον του.

Η διαδικασία της μάθησης που ακολουθούν τα νευρωνικά δίκτυα μπορεί να παρουσιαστεί με τα εξής βήματα:

- το νευρωνικό δίκτυο διεγείρεται από το περιβάλλον

- το νευρωνικό δίκτυο υφίσταται αλλαγές εξαιτίας αυτής της διέγερσης
- το νευρωνικό δίκτο ανταποκρίνεται πλέον με διαφορετικό τρόπο στο περιβάλλον του λόγω των αλλαγών που έχει υποστεί.

Για να γίνει αυτή η μάθηση χρησιμοποιείται ένα σύνολο από καλά ορισμένους κανόνες. Αυτοί οι κανόνες ονομάζονται αλγόριθμος μάθησης. Γενικά η διαδικασία της μάθησης μπορεί να θεωρηθεί ως μια αναζήτηση στο χώρο των βαρών του δικτύου με σκοπό να βρεθεί μια λύση που να εξυπηρετεί μια αντικειμενική συνάρτηση. Οι αλγόριθμοι μάθησης που υπάρχουν είναι διάφοροι και ο καθένας εξυπηρετεί διαφορετικά προβλήματα.

Οι κυριότερες κατηγορίες μηχανικής μάθησης είναι:

- η μάθηση με επίβλεψη
- η μάθηση χωρίς επίβλεψη
- η μάθηση με ενίσχυση
- η ανταγωνιστική μάθηση

Μάθηση με επίβλεψη

Στην εκπαίδευση με επίβλεψη κάθε διάνυσμα εισόδου συνοδεύεται από ένα διάνυσμα εξόδου το οποίο είναι η επιθυμητή απάντηση του δικτύου για την συγκεκριμένη είσοδο. Έτσι τα βάρη προσαρμόζονται σταδιακά ώστε να μειωθεί το σφάλμα μεταξύ της απόκρισης του δικτύου και και του επιθυμητού αποτελέσματος.

Η εκπαίδευση με επίβλεψη μπορεί να υλοποιηθεί με δύο τρόπους. Μπορεί να είναι σύγχρονη (on-line) ή ασύχρονη (off-line). Όταν υπάρχει ασύγχρονη μια ανεξάρτητη επιστημονική μηχανή σχεδιάζει και εκπαιδεύει το δίκτυο. Όταν επιτευχθεί η επιθυμητή απόδοση το δίκτυο σταματάει να εκπαιδεύεται και αρχίζει να λειτουργεί με τα δεδομένα που πήρε από την εκπαίδευση. Όταν πάλι υπάρχει συγχρονη η εκπαίδευση γίνεται σε πραγματικό χρόνο (δυναμική εκπαίδευση) χωρίς να υπάρχει μια ξεχωριστή ανεξάρτητη μηχανή. Παρότι η σύγχρονη εκπαίδευση είναι πιο δύσκολη στην υλοποίηση της καθώς οι αλγόριθμοι που χρησιμοποιούνται είναι πιο πολύπλοκοι μπορεί να αφομοιώνει διαρκώς πληροφορίες από το περιβάλλον και μπορεί να διαχειρίζεται καλύτερα μεταβολές στις συνθήκες.

Μάθηση χωρίς επίβλεψη

Με αυτή τη μέθοδο σκοπός είναι να εντοπισθούν ομοιότητες στα δείγματα που του παρέχονται και μέσω της εκπαίδευσης να τα κατατάξει σε κλάσεις. Αυτό πραγματοποιείται μέσω μιας ανεξάρτητης αντικειμενικής συνάρτησης που θα ελέγχει την ποιότητα της απεικόνισης της εξωτερικής πληροφορίας που θα δέχεται το δίκτυο από το περιβάλλον και θα έχει ως αποτέλεσμα το δίκτυο να μπορεί να κατατάσσει τα δεδομένα κλάσεις και να δημιουργεί νέες κλάσεις όταν αυτό κρίνεται απαραίτητο.

Μάθηση με ενίσχυση

Η μέθοδος αυτή χρησιμοποιεί μια συνάρτηση που ονομάζεται σήμα ενίσχυσης ώστε να αποφασίσει ποίες λειτουργίες θα χρησιμοποιούνται από το νευρωνικό δίκτυο. Έχει βασιστεί σε αρχές της ψυχολογίας και ουσιαστικά βασίζεται στην αρχή ότι εάν μια ενέργεια βοηθάει το δίκτυο να βελτιώνει την απόδοση του τότε η ενέργεια αυτή τείνει να επαναλαμβάνεται ενώ σε αντίθετη περίπτωση η λειτουργία αυτή υποβαθμίζεται.

Ανταγωνιστική μάθηση

Σε αυτή τη μέθοδο όπως δηλώνει και το όνομα της οι νευρώνες του δικτύου ανταγωνίζονται ως το ποίος θα είναι ενεργός. Μόνο ένας νευρώνας θα είναι ενεργός μια συγκεκριμένη χρονική στιγμή. Τα βασικά στοιχεία του αλγορίθμου είναι τα παρακάτω:

- αποτελείται από ένα σύνολο όμοιων νευρώνων τα οποία όμως ανταποκρίνονται διαφορετικά σε σύνολα δεδομένων από την είσοδο
- υπάρχει ένα όριο για την ισχύ του κάθε ένα από τους νευρώνες
- υπάρχει και κάποιος μηχανισμός ώστε μόνο ένας από τους νευρώνες να είναι ενεργός για κάθε χρονική στιγμή

Αυτό έχει ως αποτέλεσμα κάθε νευρώνας του δικτύου να ειδικεύεται σε παρόμοια δεδομένα εισόδου. Η απλούστερη μορφή αυτής της μεθόδου είναι να αποτελείται από μόνο ένα επίπεδο με τον κάθε νευρώνα να συνδέεται με όλους τους κόμβους εισόδου. [3]

2.3 Είδη τεχνητών νευρωνικών δικτύων

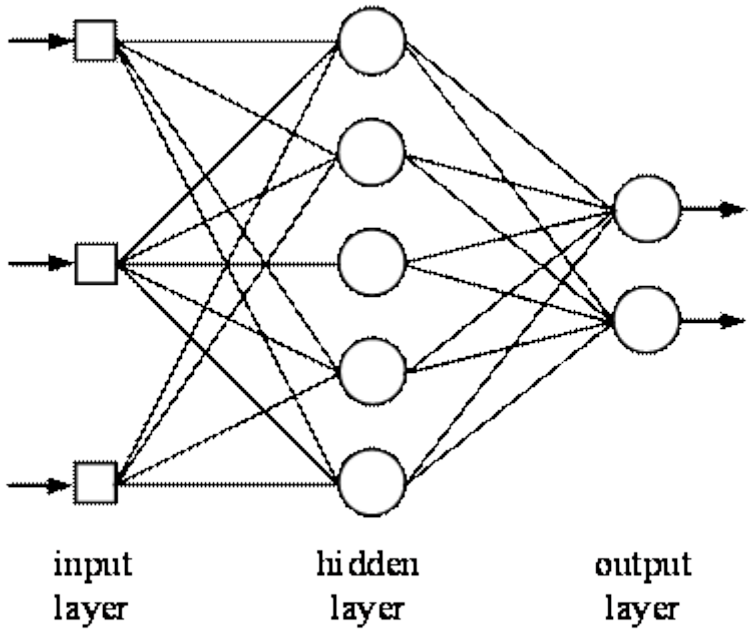
Τα τεχνηκά νευρωνικά δίκτυα μπορούν να διαχωριστούν ανάλογα με την αρχιτεκτονική τους και χωρίζονται και σε κατηγορίες. Η αρχιτεκτονική ενός δικτύου είναι πολύ σημαντική καθώς κάθε μία μπορεί να επιλύσει συγκεκριμένο τύπο προβλημάτων και κάθε αρχιτεκτονική χρειάζεται και συγκεκριμένο αλγόριθμο εκπαίδευσης. Οι κυριότερες αρχιτεκτονικές για τεχνητά νευρωνικά δίκτυα είναι:

- Δίκτυα εμπρός τροφοδότησης ενός επιπέδου (single layer feedforward networks)
- Δίκτυα εμπρός τροφοδότησης πολλών επιπέδων (multilayer feedforward networks)
- Αναδρομικά δίκτυα (Reccurent Networks)
- Δίκτυα με δομή δικτυωτού (Lattice structures)

Αξίζει να σημειωθεί ότι οι δύο πρώτες κατηγορίες είναι και οι σημαντικότερες.

Δίκτυα εμπρός τροφοδότησης ενός επιπέδου

Ένα νευρωνικό δίκτυο έχει τους νευρώνες του οργανωμένους σε επίπεδα. Η πιο απλή αρχιτεκτονική είναι αυτή όπου όλοι οι νευρώνες βρίσκονται σε ένα επίπεδο. Τα δίκτυα αυτά αποτελούνται από ένα επίπεδο εισόδων του οποίου οι κόμβοι συνδέονται με το επίπεδο των νευρώνων. Οι κόμβοι του επιπέδου εισόδου δεν μπορούν να θεωρηθούν νευρώνες καθώς απλώς μεταδίδουν το διάνυσμα εισόδου στο δίκτυο και δεν εκτελούν κανέναν απολύτως υπολογισμό. Όλοι οι υπολογισμοί γίνονται στο επίπεδο των νευρώνων και το αποτέλεσμα αυτών αποτελεί το διάνυσμα εξόδου από το δίκτυο. Η πληροφορία του δικτύου αποθηκεύεται με την μορφή αλλαγών που γίνονται στα βάρη των συνάψεων. Τα δίκτυα ενός επιπέδου αποτελούν την πιο απλή μορφή αρχιτεκτονικής τεχνητών νευρωνικών δικτύων.



Εικόνα 13:Δίκτυο εμπρός τροφοδότησης ενός επιπέδου

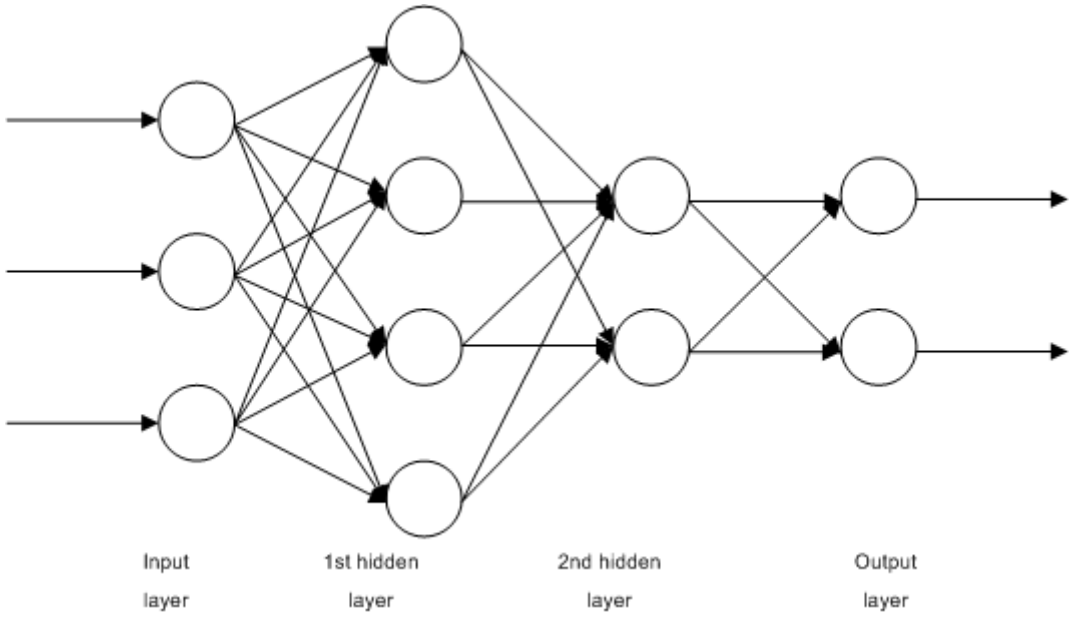
Δίκτυα εμπρός τροφοδότησης πολλών επιπέδων

Η βασική διαφορά αυτής της αρχιτεκτονικής με την προηγούμενη είναι η ύπαρξη περισσότερων από ενός επιπέδου νευρώνων. Τα επιπλέον επίπεδα ονομάζονται κρυφά επίπεδα και οι νευρώνες που αποτελούν μέρος τους κρυφοί νευρώνες. Τα κρυφά αυτά επίπεδα λειτουργούν μεταξύ της εισόδου και της εξόδου του δικτύου. Με την προσθήκη ενός ή περισσοτέρων κρυφών επιπέδων επιτυγχάνεται η μεγαλύτερη αφομοίωση πληροφορίας από το νευρωνικό δίκτυο μέσω των περισσότερων συνάψεων και της μεγαλύτερης πολυπλοκότητας αλληλεπιδράσεων που δημιουργούνται μεταξύ των νευρώνων. Αυτό είναι περισσότερο εμφανές σε προβλήματα όπου η διάσταση του διαστήματος εισόδου είναι μεγάλη.

Η διαδικασία υπολογισμού για την έξοδο δεν έχει μεγάλη διαφορά από αυτήν που ακολουθείται σε δίκτυα με ένα επίπεδο. Το διάνυσμα εισόδου επεξεργάζεται από τους νευρώνες του πρώτου επιπέδου και περνάει στο δεύτερο επίπεδο. Αυτή η διαδικασία συνεχίζεται μέχρις ότου να γίνει επεξεργασία από τους νευρώνες του τελευταίου επιπέδου και το αποτέλεσμα να περάσει στην έξοδο του δικτύου.

Αυτού του είδους η αρχιτεκτονική προσφέρει μια επιπλέον ευελιξία στην διασύνδεση μεταξύ των νευρώνων κάθε επιπέδου. Η διασύνδεση μπορεί να γίνεται είτε μερικώς είτε πλήρεις. Στην περίπτωση που υπάρχει πλήρης διασύνδεση κάθε νευρώνας συνδέεται με την έξοδο όλων των

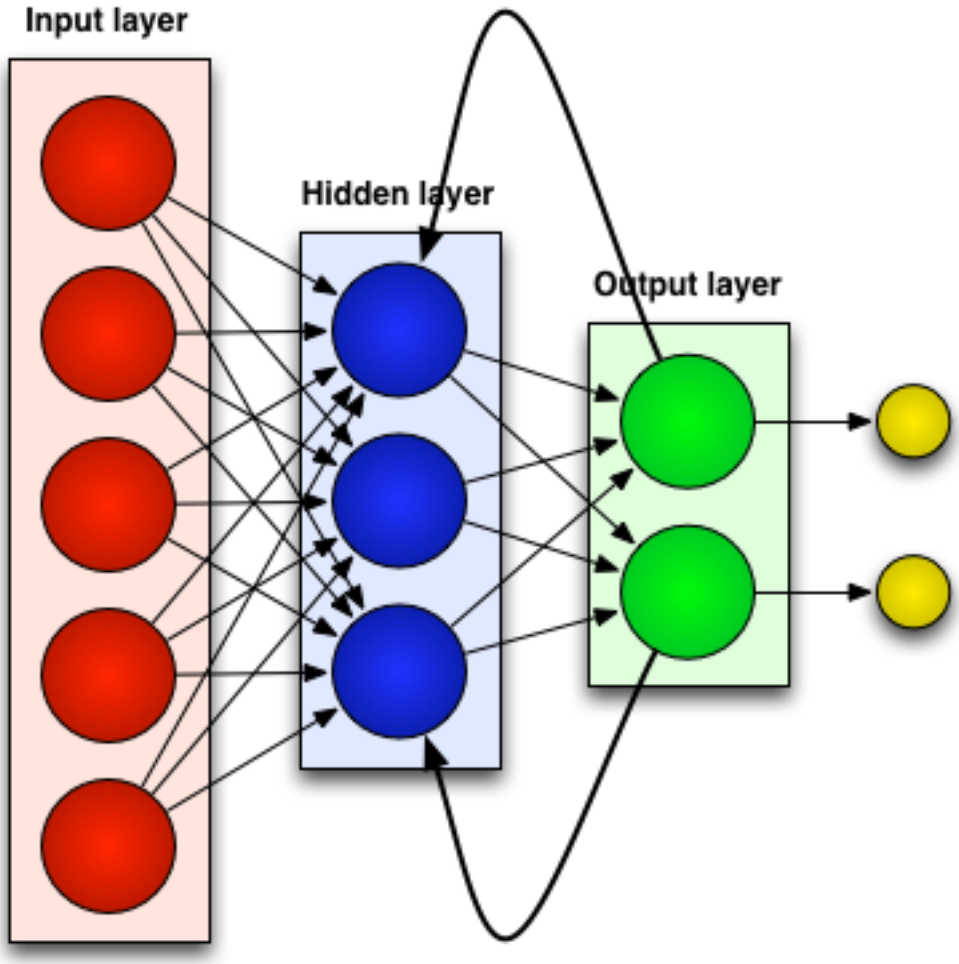
νευρώνων του προηγούμενου επιπέδου ενώ όταν υπάρχει μερική διασύνδεση δεν ενώνονται όλοι οι νευρώνες μεταξύ τους.



Εικόνα 14:Παράδειγμα εμπρός τροφοδότησης πολλαπλών επιπέδων

Αναδρομικά δίκτυα

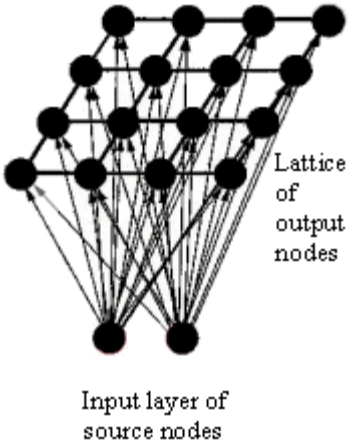
Σε αυτή την αρχιτεκτονική υπάρχει τουλάχιστον ένας βρόχος ανάδρασης. Ένα παράδειγμα για αυτό είναι ένα επίπεδο νευρώνων όπου κάθε ένας να δίνει την έξοδο του σαν είσοδο για όλους τους υπολοίπους. Υπάρχει και η πιθανότητα ένας νευρώνας να τροφοδοτεί την ίδια του του είσοδο με την έξοδο του,κάτι που ονομάζεται αυτοανάδραση. Μπορεί να υπάρχουν και πιο σύνθετες αρχιτεκτονικές όπου η ανάδραση να γίνεται μεταξύ των κρυφών επιπέδων. Συνηθίζεται οι βρόγχοι ανάδρασης να περιλαμβάνουν σε ορισμένες περιπτώσεις και κάποια στοιχεία καθυστέρησης. Η ύπαρξη των βρόχων ανάδρασης σε ένα νευρωνικό δίκτυο παίζει πολύ σημαντικό ρόλο στην ικανότητα του να μαθαίνει και στη γενικότερη απόδοση του.



Εικόνα 15:Παράδειγμα αναδρομικού δικτύου

Δίκτυα με δομή δικτυωτού

Η δομή αυτής της αρχιτεκτονικής αποτελείται από συστοιχίες νευρώνων καθένας από τους οποίους συνδέεται με τους κόμβους εισόδου στο δίκτυο. Αυτές η συστοιχίες μπορούν να είναι μίας ή περισσότερων διαστάσεων ανάλογα με τις διαστάσεις του χώρου του προβλήματος. Ουσιαστικά είναι σαν τα δίκτυα προς τα εμπρός τροφοδότησης με τη διαφορά ότι οι νευρώνες εξόδου διατεταγμένοι σε γραμμές και στήλες αναλόγως με τις διαστάσεις. [3]



Εικόνα 16:Παράδειγμα δικτύου με δομή δικτυωτού

3. Γενετικοί αλγόριθμοι

3.1 Ιστορία Γενετικών αλγορίθμων

Η ιδέα για την εξέλιξη των γενετικών αλγορίθμων βασίζεται στην αρχή εξέλιξης των ειδών (evolution of species) που αναπτύχθηκε από τον Δαρβίνο στα μέσα του 19ου αιώνα. Η θεωρία αυτή προκάλεσε πολλές αντιδράσεις καθώς ερχόταν σε άμεση σύγκρουση με τις επικρατούσες θρησκευτικές αντιλήψεις περί προέλευσης της ζωής αλλά και με τις γενικότερες πεποιθήσεις εκείνης της εποχής. Με την πάροδο των χρόνων η θεωρία αυτή έχει γίνει πλήρως αποδεκτή από τον επιστημονικό κόσμο καθώς κατάφερε να δώσει ικανοποιητικές απαντήσεις σε θεμελιώδη ερωτήματα παρόλα αυτά υπάρχουν ακόμη άνθρωποι που πιστεύουν σε θεικές παρεμβάσεις για την εξέλιξη αντι για μία φυσιολογική εξέλιξη. Η θεωρία αυτή έχει σκοπό να δώσει εξηγήσεις για την προέλευση και της βασικές λειτουργίες του φαινομένου της ζωής. Τα βασικότερα στοιχεία της θεωρίας αυτή που σχετίζονται με τον τρόπο λειτουργίας των γενετικών αλγορίθμων είναι τα παρακάτω:

- Δεν υπάρχει αντικειμενική βάση διαχωρισμού των ζωντανών οργανισμών μεταξύ του ίδιου βιολογικού είδους. Για παράδειγμα δεν υπάρχει διαχωρισμός μεταξύ των ανθρώπων σε ανώτερους και κατώτερους. Αυτό στις μέρες μας μπορεί να φαντάζει αυτονόητο

όμως η θεωρία αυτή αναπτύχθηκε σε μία εποχή όπου υπήρχε ακόμη ο θεσμός της δουλείας και ο διαχωρισμός των ανθρώπων ανάλογα την καταγωγή τους. Σε κάθε βιολογικό είδος μερικά από τα άτομα αφήνουν περισσότερους απογόνους σε σύγκριση με τα υπόλοιπα και έτσι τα κληροδοτούμενα χαρακτηριστικά των αναπαραγωγικά επιτυχημένων ατόμων γίνονται περισσότερα για την επόμενη γενιά. Οι παράγοντες οι οποίοι επηρεάζουν ποίοι οργανισμοί θα κατορθώσουν να επιβιώσουν και να πολλαπλασιαστούν είναι η αλλαγή του περιβάλλοντος και των συνθηκών διαβίωσης. Έτσι ανάλογα τις δυσκολίες και τις αντιξοότητες που αντιμετωπίζουν αλλάζουν τα χαρακτηριστικά τους καθώς προσπαθούν να προσαρμοστούν με τελικό στόχο την επιβίωση τους.

- Η αλλαγή που συμβαίνει στα χαρακτηριστικά των ατόμων (individuals) είναι ουσιαστικά αλλαγή στα χρωμοσώματα (chromosomes) τους. Τα χρωμοσώματα είναι πολύπλοκα οργανικά μόρια τα οποία κωδικοποιούν τα χαρακτηριστικά τους. Τα χρωμοσώματα με τη σειρά τους αποτελούνται από μικρότερα μέρη τα οποία ονομάζονται γονίδια (genes). Υπάρχουν κάποια γονίδια τα οποία επηρεάζουν συγκεκριμένα χαρακτηριστικά γνωρίσματα του ατόμου και βρίσκονται σε συγκεκριμένες θέσεις του χρωμοσώματος και ονομάζονται loci. Το σύνολο της γενετικής πληροφορίας που είναι κωδικοποιημένο στα γονίδια ονομάζεται γονότυπος (genotype). Για την δημιουργία ενός νέου οργανισμού γίνεται με την αποκωδικοποίηση των χρωμοσωμάτων. Το σύνολο των χαρακτηριστικών που μπορούμε να δούμε μέσω αυτής της αποκωδικοποίησης ονομάζεται φαινότυπος (phenotype).
- Οι βασικές λειτουργιές του φαινομένου της εξέλιξης είναι η αναπαραγωγή (reproduction) και η μετάλλαξη (mutation). Η αναπαραγωγή επιτυγχάνεται όταν δύο μέλη του οργανισμού ανταλλάσσουν γενετικό υλικό με στόχο την παραγωγή απογόνων. Η διαδικασία της αναπαραγωγής διαφέρει από είδος σε είδος. Οι απόγονοι μπορούν να έχουν χαρακτηριστικά από τους γονείς τους καθώς επίσης και από προηγούμενες γενιές. Για κάθε χαρακτηριστικό ο απόγονος έχει πάρει ένα γονίδιο από κάθε γονέα. Μερικές φορές τα γονίδια αυτά συμφωνούν μεταξύ τους ως προς το ποία τιμή θα πάρει το χαρακτηριστικό,όπως για παράδειγμα να

> υποδεικνύουν και τα δύο το ίδιο χρώμα ματιών,ενώ σε άλλες περιπτώσεις δεν συμφωνούν δηλαδή μπορεί να υποδεικνύουν διαφορετικό χρώμα ματιών. Στην περίπτωση που δεν συμφωνούν κυριαχεί η τιμή ενός γονιδιού ενώ το άλλο περνάει σαν πιθανότητα για κάποια επόμενη γενιά. Σε αυτή τα δύο γονίδια ονομάζονται αλληλόμορφα (alleles). Το γονίδιο που τελικώς υπερισχύει ονομάζεται κυρίαρχο (dominant) ενώ το άλλο υπολειπόμενο (recessive). Η μετάλλαξη μπορεί να συμβεί σε πολύ σπάνια διαστήματα και μπορεί να προκληθεί είτε από λανθασμένη αντιγραφή βιολογικών πληροφοριών είτε από εξωτερικούς παράγοντες του περιβάλλοντος που ζουν τα άτομα. Η μετάλλαξη δεν είναι υποχρεωτικά αρνητικό αφού πολλά από τα λάθη που έγιναν είχαν σημαντικό ρόλο στην προοδευτική εξέλιξη της ζωής. [2]

Τα τελευταία χρόνια έχει παρατηρηθεί ένα αυξανόμενο ενδιαφέρον για προγραμματιστικές τεχνικές που βασίζονται στις αρχές της φυσικής εξέλιξης για την επίλυση διαφόρων προβλημάτων. Αυτού του είδους οι τεχνικές λειτουργούν έχοντας έναν πληθυσμό από πιθανές λύσεις για το πρόβλημα που χρήζει επίλυσης και εφαρμόζοντας πάνω σε αυτόν το πληθυσμό διάφορες διαδικασίες εμπνευσμένες από την βιολογική εξέλιξη,ο πληθυσμός περνάει από γενιά σε γενιά. Τα συστήματα που λειτουργούν με αυτήν την μεθοδολογία ονομάζονται Εξελικτικοί αλγόριθμοι (evolutionary algorithms). Περιπτώσεις εξελικτικών αλγορίθμων είναι:

- Οι γενετικοί αλγόριθμοι (genetic algorithms)
- Ο εξελικτικός προγραμματισμός (evolutionary programming)
- Οι στρατηγικές εξέλιξης (evolution strategies)
- Τα συστήματα ταξινόμησης (classifier systems)
- Ο γενετικός προγραμματισμός (genetic programming)

Η πρώτη εμφάνιση των γενετικών αλγορίθμων ήταν το 1950 όταν επιστήμονες από τον χώρο της βιολογίας αποφάσισαν να χρησιμοποιήσουν υπολογιστές για την προσομοίωση διαφόρων

πολύπλοκων βιολογικών συστημάτων. Η σημερινή τους μορφή προήλθε το 1975 από τον John Holland και των συνεργατών του στο Πανεπιστήμιο του Michigan.Η βασική ιδέα που βασίζονται οι γενετικοί αλγόριθμοι είναι η μίμηση των μηχανισμών τς βιολογικής εξέλιξης που υπάρχουν στη φύση. Ένα από τα πιο γνωστά παραδείγματα είναι η παρατήρηση ενός οικοσυστήματος στο οποίο κατοικεί ένας συγκεκριμένος πληθυσμός από λαγούς. Κάποιοι από τους λαγούς είναι πιο εύστροφοι και γρήγοροι από τους υπόλοιπους συνεπώς αυτοί θα έχουν τις περισσότερες πιθανότητες επιβίωσης. Φυσικά θα επιβιώσουν και κάποιοι λαγοί που δεν έχουν αυτά τα χαρακτηριστικά οι οποίοι θα αναμειχθούν με τους υπολοίπους και θα αρχίσουν την παραγωγή της επόμενης γενιάς. Η γενιά αυτή θα συνδυάζει τα χαρακτηριστικά της προηγούμενης δηλαδή κάποιοι εύστροφοι λαγοί θα αναμειχθούν με λιγότερο εύστροφους,κάποιοι γρήγοροι με κάποιους λιγότερο γρήγορους κτλ. Τελικώς η νέα γενιά θα είναι κατά μέσο όρο γρηγορότεροι και εξυπνότεροι από τους προγόνους τους. Φυσικά για την διατήρηση της φυσικής ισορροπίας παρόμοια διαδικασία αναπαραγωγής γίνεται από όλα τα είδη του οικοσυστήματος. [2]

3.2 Γενικό σχήμα γενετικών αλγόριθμων

Οι γενετικοί αλγόριθμοι λειτουργούν διατηρώντας έναν πληθυσμό από πιθανές λύσεις για το πρόβλημα που προσπαθούμε να επιλύσουμε. Με αυτό τον τρόπο ο γενετικός αλγόριθμος καταφέρνει να πραγματοποιήσει αναζήτηση λύσεων σε πολλές κατευθύνσεις και μπορεί να καταγράψει και να ανταλλάσσει πληροφορίες μεταξύ αυτών σε αντίθεση με άλλες μεθόδους αναζήτησης που ένα μόνο σημείο του διαστήματος αναζήτησης. Με το πέρασμα των γενιών οι καλύτερες λύσεις χρησιμοποιούνται για αναπαραγωγή ενώ οι χειρότερες αγνοούνται. Για να γίνει αυτός ο διαχωρισμός γίνεται η χρήση μιας αντικειμενικής συνάρτησης (objective function) η οποία έχει το ρόλο του περιβάλλοντος στο οποίο εξελίσσεται ο πληθυσμός.

Ένας γενετικός αλγόριθμος μπορεί να αναπαρασταθεί ως εξής:Για μια γενιά t διατηρείτε ένας πληθυσμός P(t)=$\{x_1^t,x_2^t,...,x_n^t\}$ από n πιθανές λύσεις.Σε κάθε λύση x_i^t δίνεται ένα μέτρο της καταλληλότητας του.

Αφού γίνει αυτό για κάθε ένα από τα μέλη του πληθυσμού προκύπτει ένας νέος πληθυσμός (γενιά t+1) μέσω της επιλογής των καταλληλότερων στοιχείων. Στη συνέχεια χρησιμοποιείτε η διασταύρωση και η μετάλλαξη για την δημιουργία νέων πιθανών λύσεων. Με τη διασταύρωση γίνεται συνδυασμός των στοιχείων των χρωμοσωμάτων που αποτελούν τους δύο γονείς με αποτέλεσμα την δημιουργία δύο απογόνων. Με την διαδικασία της μετάλλαξης γίνεται μία τυχαία αλλαγή σε ένα ή περισσότερα γονίδια ενός χρωμοσώματος με πιθανότητα ίση με το ρυθμό μετάλλαξης (mutation rate) .

Τα βασικά στοιχεία που αποτελούν τη δομή ενός γενετικού αλγορίθμου μπορούν να αριθμηθούν στα παρακάτω:

- Μία γενετική αναπαράσταση των πιθανών λύσεων του προβλήματος
- Ένα τρόπο δημιουργίας ενός αρχικού πληθυσμού από πιθανές λύσεις(αρχικοποίηση)
- Μια αντικειμενική συνάρτηση αξιολόγησης για τα μέλη του πληθυσμού
- Γενετικούς τελεστές για την δημιουργία νέων παιδιών
- Τιμές για τις διάφορες παραμέτρους που χρησιμοποιεί ο γενετικός όπως το μέγεθος του πληθυσμού [2]

3.3 Πλεονεκτήματα και μειονεκτήματα

Η χρήση των γενετικών αλγορίθμων για επίλυση προβλημάτων προσφέρει πολλά πλεονεκτήματα τα σημαντικότερα τα οποία είναι τα παρακάτω:

- Γρήγορη και αξιόπιστη επίλυση σε δύσκολα προβλήματα. Η μεγάλη αποδοτικότητα των γενετικών αλγορίθμων είναι και ένας από τους σημαντικότερους λόγους χρήσης τους. Και στην θεωρία και στην πράξη έχει αποδειχτεί ότι οι γενετικοί αλγόριθμοι μπορούν να αντιμετωπίσουν καλύτερα προβλήματα με πολλές και δύσκολα προσδιοριζόμενες λύσεις. Αξίζει επίσης να σημειωθεί ότι ενώ άλλες μέθοδοι είναι ανεπαρκείς όσον αφορά την εύρεση των ακρότατων για συναρτήσεις που παρουσιάζουν μεγάλες

διακυμάνσεις για τους γενετικούς αλγορίθμους δεν αποτελούν πρόβλημα.

- Μπορούν εύκολα να συνεργαστούν με τα υπάρχοντα μοντέλα και συστήματα. Οι γενετικοί αλγόριθμοι είναι εξαιρετικά απλοί στην υλοποίηση τους και προσφέρουν το σημαντικό πλεονέκτημα της χρήσης τους με προσθετικό τρόπο στα ήδη υπάρχοντα μοντέλα χωρίς να απαιτείται η επανασχεδίαση τους. Δεν αντιμετωπίζουν προβλήμα όσον αφορά την συνεργασία τους με τον ήδη υπάρχοντα κώδικα διότι αρκεί να κωδικοποιηθεί το πρόβλημα με την χρήση χρωμοσωμάτων και η αντικειμενική συνάρτηση και δεν τους ενδιαφέρει άμεσα ο ρόλος της συνάρτησης μέσα στο σύστημα ή ολόκληρη η δομή του.
- Είναι εύκολα επεκτάσιμοι και εξελίξιμοι. Οι γενετικοί αλγόριθμοι μπορούν να υποστούν αλλαγές ανάλογα με την κρίση του προγραμματιστή. Μπορεί να επιλεγεί διαφορετικός τρόπος υλοποίησης που να εξυπηρετεί καλύτερα το εκάστοτε πρόβλημα ή ακόμα και να υπάρχουν λειτουργίες που δεν είναι εμπνευσμένες από την φύση με σκοπό την καλύτερη απόδοση. Αυτές οι αλλαγές είναι αποδεκτές και υπάρχουν και περιπτώσεις που επιβάλλουν τέτοιου είδους αντιμετώπισης.
- Μπορούν να συμμετέχουν με άλλες μεθόδους σε υβριδικές μορφές. Η μεγάλη ευελιξία και οι μικρές προγραμματιστικές απαιτήσεις των γενετικών αλγορίθμων τους επιτρέπει να μπορούν εύκολα να προσαρμοστούν μαζί με κάποια άλλη μέθοδο που έχει πολύ υψηλή αποδοτικότητα σε ειδικές περιπτώσεις προβλημάτων λόγω εξειδίκευσης.
- Μπορούν να εφαρμοστούν σε περισσότερα πεδία από κάθε άλλη μέθοδο. Το χαρακτηριστικό των γενετικών αλγορίθμων που τους επιτρέπει αυτό το πλεονέκτημα είναι η ελευθερία επιλογής των κριτηρίων που καθορίζουν την επιλογή μέσα στο τεχνικό περιβάλλον. Οι γενετικοί αλγόριθμοι λοιπόν έχουν μια μεγάλη γκάμα τομέων που μπορούν να εφαρμοστούν όπως η επίλυση μαθηματικών συναρτήσεων και στην εκπαίδευση νευρωνικών δικτύων.
- Δεν απαιτούν περιορισμούς στις συναρτήσεις που επεξεργάζονται και είναι ανθεκτικοί στα σφάλματα. Το χαρακτηριστικό που κάνει τους γενετικούς αλγορίθμους κατάλληλους για μεγάλο φάσμα

προβλημάτων είναι ότι περιορισμοί όπως ύπαρξη παραγώγων,συνέχεια και θόρυβος που καθιστούν άλλες μεθόδους ακατάλληλες ,τους είναι αδιάφορα.

- Δεν ενδιαφέρει η σημασία της υπό εξέτασης πληροφορίας.Καθώς η μοναδική επικοινωνία του γενετικού αλγορίθμου είναι η αντικειμενική συνάρτηση η σημασία του προβλήματος δεν επηρεάζει τη λειτουργία του. Αξίζει να σημειωθεί ότι υπάρχουν προβλήματα όπου οι γενετικοί αλγόριθμοι δεν έχουν καταφέρει να δώσουν λύση γιαυτό όμως ευθύνεται η φύση του χώρου και όχι το πληροφοριακό περιεχόμενο του προβλήματος.
- Μπορούν εύκολα να παραλληλιστούν. Οι γενετικοί αλγόριθμοι επεξεργάζονται μια σειρά από λύσεις οι οποίες είναι ανεξάρτητες μεταξύ τους με αποτέλεσμα να μπορούμε εύκολα να δημιουργήσουμε μια παράλληλη εκδοχή τους. Επειδή σε κάθε βήμα τους επεξεργάζονται μεγάλες ποσότητες πληροφορίας και έτσι μπορούν να καλύψουν μεγάλους χώρους σε μικρό χρόνο.
- Μπορούν να κάνουν ταυτόχρονα εξερεύνηση του χώρου αναζήτησης και εκμετάλλευση της ήδη επεξεργασμένης πληροφορίας. Με το τυχαίο ψάξιμο γίνεται καλή εξερεύνηση του χώρου χωρίς όμως να γίνεται εκμετάλλευση της πληροφορίας, ενώ αντιθέτως κάνοντας αναζήτηση με μικρά άλματα στη συνάρτηση γίνεται καλή εκμετάλλευση της πληροφορίας αλλά όχι εξερεύνηση. Γενικά είναι επιθυμητό να συνυπάρχουν για το καλύτερο αποτέλεσμα αλλά κάτι τέτοιο συναντάται πολύ σπάνια σε άλλες μεθόδους. Οι γενετικοί αλγόριθμοι το καταφέρνουν στο βέλτιστο βαθμό πράγμα που τους κάνει ιδιαίτερα αποδοτικούς και ελκυστικούς.
- Επιδέχονται παράλληλη υλοποίηση. Οι γενετικοί αλγόριθμοι μπορούν να εκμεταλλευτούν τα πλεονεκτήματα των παράλληλων μηχανών καθώς μπορούν εύκολα να δεχτούν παράλληλη υλοποίηση. Αυτό το χαρακτηριστικό βοηθάει στην αύξηση της αποδοτικότητας τους.

Όσων αφορά τα μειονεκτήματα των γενετικών αλγορίθμων είναι περισσότερο θεωρητικά παρά πρακτικά.

- Πρόβλημα εξοικείωσης με την γενετική. Καθώς η ιδέα των γενετικών αλγορίθμων προέρχεται από την Βιολογία οι όροι που

χρησιμοποιούνται για την κατανόηση τους μπορεί να ηχούν παράξενα σε αυτούς που ασχολούνται με την επιστήμη της πληροφορικής. Αυτό βέβαια δεν αποτελεί πρόβλημα στην πράξη καθώς οι γενετικοί αλγόριθμοι απλώς βασίζονται σε διάφορες τεχνικές που παρουσιάζονται στην φύση χωρις όμως να ενδιαφέρονται σε βάθος πως λειτουργούν και έτσι δεν είναι απαραίτητο το γνωστικό υπόβαθρο που έχουν οι βιολόγοι. Εξάλου η εξέλιξη των γενετικών αλγορίθμων δεν εξαρτάται από τις θεωρίες στις οποίες βασίστηκε.

- Το πρόβλημα του χρόνου. Υπάρχει δυσπιστία σχετικά με την αποδοτικότητα των γενετικών αλγορίθμων λόγω του ότι βασίζονται σε διαδικασίες που στη φύση απαιτούν πολλά χρόνια και πολλές γενιές για να εξελιχθούν. Αυτό όμως δεν ισχύει προγραμματιστικά καθώς τα άτομα κωδικοποιούνται με συμβολοσειρές και το περιβάλλον με απλές μαθηματικές συναρτήσεις. Οπότε οι ταχύτητες που μπορούν να επιτύχουν είναι πολύ υψηλές.

Τα πρακτικά μειονεκτήματα που παρουσιάζονται στους γενετικούς αλγορίθμους είναι ότι μπορούν να παρουσιάσουν αργό χρόνο σύγκλισης και σε κάποιες περιπτώσεις μπορούν να εγκλωβιστούν σε τοπικά ελάχιστα της αντικειμενικής συνάρτησης. Βέβαια αυτά τα μειονεκτήματα είναι αντιμετωπίσιμα προσαρμόζοντας τους παραμέτρους του γενετικού στο πρόβλημα. [2]

3.4 Δομή ενός γενετικού αλγορίθμου

Οι γενετικοί αλγόριθμοι αποτελούνται από κάποιες αρκετά απλές αλλά ισχυρές διαδικασίες. Αυτές οι διαδικασίες δεν είναι όλες εξίσου σημαντικές και σε κάποιες περιπτώσεις μπορούν και να μην υπάρχουν σε κάποιο πρόβλημα παρόλο αυτά είναι όλες σημαντικές για τον ρόλο που παίζουν σην επίλυση του προβλήματος.

Αντικειμενική Συνάρτηση

Η αντικειμενική συνάρτηση είναι ίσως η πιο σημαντική διαδικασία ενός γενετικού αλγορίθμου και δεν λείπει από κανένα πρόβλημα. Συνήθως υλοποιείται επιστρέφοντας μια τιμή που υποδεικνύει πόσο καλά μπορεί να επιλύσει το πρόβλημα η είσοδος της. Οπότε παίζει μεγάλο και καθοριστικό ρόλο στην επιβίωση των ατόμων του πληθυσμού. Η αντικειμενική συνάρτηση παίζει το ρόλο του περιβάλλοντος στο τεχνικό μοντέλο και είναι η μόνη πληροφορία που έχει ο γενετικός αλγόριθμος σχετικά με το πρόβλημα που προσπαθεί να επιλύσει. Όσο πιο απλή είναι αυτή η συνάρτηση τόσο πιο γρήγορα γίνονται οι διαδικασίες. Η αντικειμενική συνάρτηση δεν επηρεάζει την κωδικοποίηση των υπόλοιπων διαδικασιών του γενετικού αλγορίθμου και είναι τελείως ανεξάρτητη. Αυτό είναι ένα πολύ σημαντικό πλεονέκτημα καθώς ουσιαστικά σημαίνει ότι μπορούμε να χρησιμοποιήσουμε την ίδια αντικειμενική συνάρτηση με διάφορες κωδικοποιήσεις για διαφορετικά προβλήματα ή και το αντίθετο.

Μετά τον ορισμό της αντικειμενικής συνάρτησης ακολουθούν οι λεγόμενες γενετικές διαδικασίες οι οποίες μπορούμε πολύ να πούμε ότι είναι οι διαδικασίες "τρεξίματος" του γενετικού αλγορίθμου. Σε αυτές τις διαδικασίες γίνεται και το μεγαλύτερο μέρος των υπολογισμών. Τα βήματα που αποτελούν το υπόλοιπο κομμάτι του γενετικού είναι τα παρακάτω:

1. Δημιουργείται ένας αρχικός πληθυσμός από πιθανές λύσεις για το πρόβλημα
2. Κάθε λύση αξιολογείται ως προς την καταλληλότητα της με τη χρήση της αντικειμενικής συνάρτησης
3. Επιλέγεται ένας νέος πληθυσμός από τον ήδη υπάρχων ανάλογα με την καταλληλότητα των ατόμων του
4. Στον νέο πληθυσμό που επιλέχθηκε εφαρμόζεται η διαδικασία της διασταύρωσης
5. Εφαρμόζεται στον πληθυσμό η διαδικασία της μετάλλαξης
6. Πλέον μετα και την εκτέλεση του βήματος 5 έχει δημιουργηθεί η νέα γενιά και ο αλγόριθμος επιστρέφει στο βήμα 2
7. Ο γενετικός αλγόριθμος τερματίζει την εκτέλεση του μετά την επιτυχή ολοκλήρωση της τερματικής του συνθήκης

Παρακάτω γίνεται περισσότερη ανάλυση για αυτές τις σημαντικές διαδικασίες των γενετικών αλγορίθμων.

Αρχικοποίηση

Με την αρχικοποίηση δημιουργείται ο αρχικός πληθυσμός πάνω στον οποίο θα λειτουργήσει ο γενετικός αλγόριθμος. Αυτή η αρχικοποίηση στις περισσότερες περιπτώσεις γίνεται τυχαία και το μέγεθος του μπορεί να ορίζεται είτε από το χρήστη είτε από τα δεδομένα του προβλήματος. Υπάρχουν και κάποιες περιπτώσεις όπου για την αρχικοποίηση του πληθυσμού χρησιμοποιείται κάποια μέθοδος ώστε ο αρχικός πληθυσμό να προσφέρει κάποιο πλεονέκτημα.

Αξιολόγηση

Η αξιολόγηση γίνεται μέσα σε μία επανάληψη και εκτελείτε για κάθε γενιά του πληθυσμού. Συνήθως η διαδικασία της αξιολόγησης είναι αρκετά απλή,για κάθε γενιά του πληθυσμού κάθε άτομο αξιολογείται ως προς την καταλληλότητα του για να οδηγήσει το πρόβλημα προς την επίλυση του μέσω της αντικειμενικής συνάρτησης.

Επιλογή

Με την διαδικασία της επιλογής καθορίζονται τα άτομα του τρέχοντα πληθυσμού που θα συμμετέχουν στην δημιουργία της επόμενης γενιάς. Σκοπός της επιλογής είναι τα καταλληλότερα άτομα να περνάνε στην επόμενη γενιά και να αυξάνεται η καταλληλότητα της νέας γενιάς .Σε περίπτωση που η διαδικασία της επιλογής δεν αποτελεί μέρος κάποιου γενετικού αλγορίθμου αυτομάτως σημαίνει ότι η επιλογή γίνεται με εντελώς τυχαίο τρόπο. Ο λόγος που τα άτομα που θα περαστούν στην επόμενη γενιά επιλέγονται μέσω κάποιων διαδικασιών ακολουθώντας κάποια κριτήρια και δεν επιλέγονται τυφλά τα άτομα με την μεγαλύτερη καταλλητότητα είναι για να αποφευχθεί πρόωρη σύγκλιση του αλγορίθμου.

Οι βασικότερες και πιο συνηθισμένες μέθοδοι επιλογής είναι:

- η επιλογή μέσω μιας εξαναγκασμένης ρουλέτας
- η μέθοδος tournament

Μέθοδος Επιλογής Ρουλέτας

Σε αυτήν την μέθοδο τα άτομα του πληθυσμού περιλαμβάνονται στη ρουλέτα σε αναλογία με την απόδοση τους. Η ρουλέτα δουλεύει με

πιθανότητες. Για να υπολογιστεί η πιθανότητα για να επιλεγεί το κάθε άτομο η ρουλέτα ακολουθεί την εξής διαδικασία:

- Αρχικά υπολογίζεται η καταλληλότητα όλων των ατόμων του τρέχοντα πληθυσμού
- Υπολογίζεται η συνολική απόδοση του πληθυσμού
- Υπολογίζεται η πιθανότητα επιλογής για κάθε άτομο του πληθυσμού
- Τέλος υπολογίζεται η αθροιστική πιθανότητα επιλογής για το κάθε άτομο

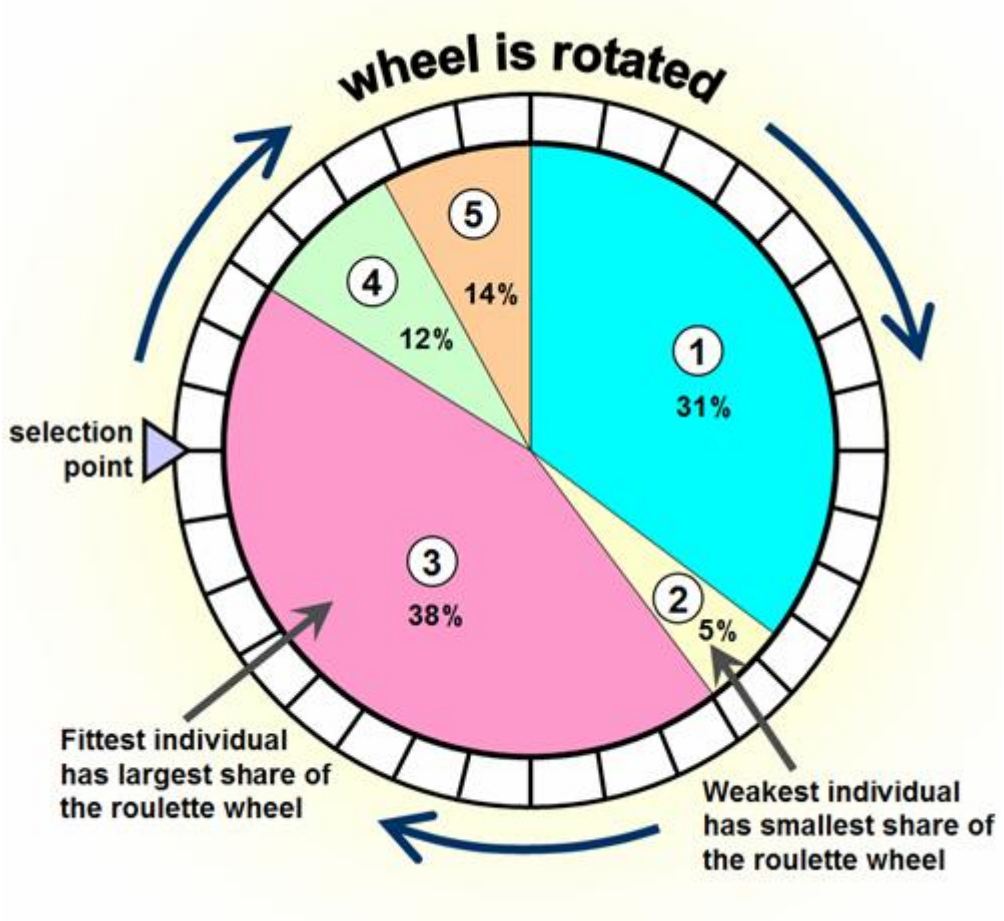
Αφού έχουν υπολογιστεί όλα τα παραπάνω εκτελείτε η ρουλέτα τόσες φορές όσο είναι το μέγεθος του πληθυσμού και η επιλογή των ατόμων γίνεται ως εξής:

- Επιλέγεται ένας τυχαίος αριθμός μέσα από το διάστημα [0 1]
- Αν η αθροιστική συχνότητα του ατόμου που ελέγχεται είναι μικρότερη είναι από τον τυχαίο αριθμό που προέκυψε προηγουμένως τότε επιλέγεται αυτό το άτομο αλλιώς προχωράει στο επόμενο

Αυτή η μέθοδος αναπαράστασης της ρουλέτας γίνεται με τη χρήση δεκαδικών στοιχείων. Υπάρχει και η αναπαράσταση με έναν πίνακα ακέραιων.

Με την αναπαράσταση μέσω ενός πίνακα ακεραίων τα άτομα του πληθυσμού εμφανίζονται στον πίνακα ανάλογα με την καταλληλότητα τους. Για παράδειγμα το άτομο με την μεγαλύτερη καταλληλότητα θα εμφανιστεί στον πίνακα περισσότερες φορές από κάθε άλλο άτομο του πληθυσμού ενώ το άτομο με την μικρότερη καταλληλότητα λιγότερες από κάθε άλλο. Ο πίνακας δεν είναι υποχρεωτικό να έχει κάποια σχέση με τον αριθμό των ατόμων του πληθυσμού αλλά προτιμάται να είναι ένα πολλαπλάσιο του. Στην συνέχεια για να γίνει η επιλογή η ρουλέτα διαλέγει τυχαία ένα άτομο μέσα από τον πίνακα. Είναι προφανές ότι τα άτομα που εμφανίζονται περισσότερες φορές μέσα στο πίνακα έχουν και περισσότερες πιθανότητες να επιλεγούν σε αντίθεση με αυτά που εμφανίζονται λιγότερες φορές. Γενικά αυτή η μέθοδος αναπαράστασης της ρουλέτας δεν προτιμάται συχνά. Παρότι ο αλγόριθμος για την

αναπαράσταση μέσω πίνακα είναι πιο απαιτητικός από την αναπαράσταση μέσω δεκαδικών στοιχείων,επιφλερει πιο γρήγορα αποτελέσματα.



Εικόνα 17:Μέθοδος επιλογής ρουλέτας

Παρότι η μέθοδος της ρουλέτας είναι η πιο διαδεδομένη μέθοδος επιλογής αυτό δεν σημαίνει ότι δεν υπάρχουν προβλήματα. Τα σημαντικότερα μειονεκτήματα που συναντώνται είναι:

- Η συχνότητα που εμφανίζονται τα άτομα στη ρουλέτα. Τα άτομα με πολύ υψηλή καταλληλότητα τείνουν να εμφανίζονται πάρα πολλές φορές μέσα στην ρουλέτα με αποτέλεσμα να επισκιάζουν τα υπόλοιπα και να επιλέγονται συνεχώς τα ίδια. Υπάρχουν και περιπτώσεις όπου τα άτομα με μικρότερες καταλληλότητες δεν θα εμφανίζονται καμία φορά στη ρουλέτα για επιλογή.
- Πρόωρη σύγκλιση. Σε περίπτωση που σε κάποια γενιά δημιουργείται ένα άτομο που έχει μια καταλληλότητα

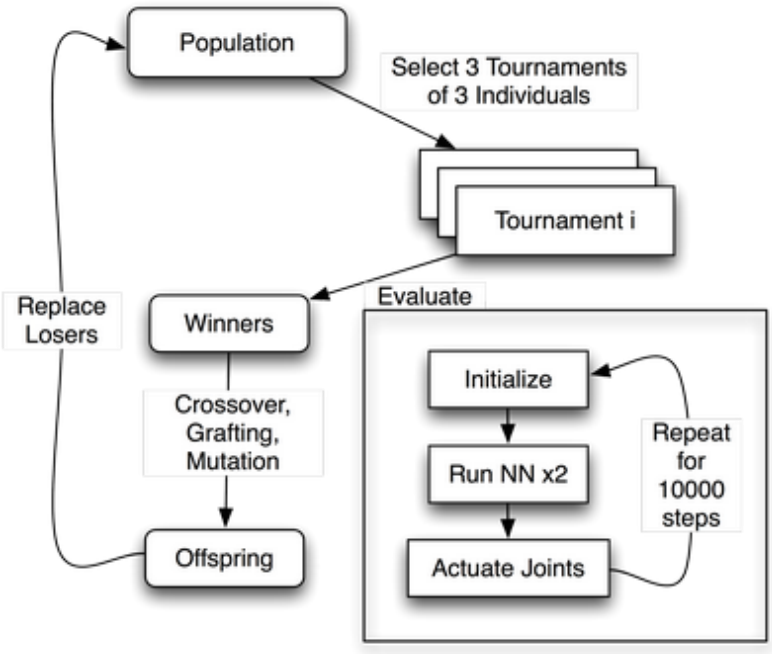
υψηλότερη από το μέσο όρο αλλά όχι αρκετή για να επιλύσει το πρόβλημα. Λόγω του τρόπου επιλογής της ρουλέτας αυτό το άτομο θα έχει πολύ περισσότερες πιθανότητες να επιλεγεί κάτι το οποίο μπορεί να οδηγήσει τον αλγόριθμο να μην μπορεί να βρεθεί πιο κοντά στη λύση του προβλήματος από ότι βρίσκεται το συγκεκριμένο άτομο

- Μπορεί να οδηγήσει τον αλγόριθμο σε στασιμότητα. Καθώς προχωράει ο αλγόριθμος υπάρχει πιθανότητα σε κάποιο σημείο όλα τα άτομα να είναι τόσο παρόμοια μεταξύ τους ώστε να μην μπορεί να βρεθεί πιο βέλτιστη λύση.

Βέβαια αυτά τα προβλήματα μπορούν να αντιμετωπιστούν με κατάλληλο προγραμματισμό. [1]

Μέθοδος Επιλογής Tournament

Η μέθοδος αυτή επιλέγει μια σειρά από χρωμοσώματα τα συγκρίνει και κρατάει σαν γονέα αυτό με την μεγαλύτερη καταλληλότητα. Η μέθοδος αυτή είναι η πιο αποδεκτή καθώς είναι πιο γρήγορη αφού δεν χρειάζεται ούτε να υπολογιστούν οι πιθανότητες επιτυχίας των χρωμοσωμάτων ούτε απαιτείται ταξινόμηση των ατόμων ανάλογα με την καταλληλότητα τους. Επίσης ταιριάζει περισσότερο με την διαδικασία επιλογής των γονέων που συμβαίνει στη φύση. Με τη χρήση της μεθόδου tournament δεν εμφανίζονται και τα προβλήματα της στασιμότητας και της πρόωρης σύγκλισης όπως στη ρουλέτα. [1]



Εικόνα 18:Παράδειγμα μεθόδου tournament

Διασταύρωση

Το επόμενο στάδιο είναι να χρησιμοποιηθούν τα άτομα που επιλέχθηκαν σε ένα είδους γονιμοποίησης που θα επιφέρει την νέα γενιά. Η διαδικασία της διασταύρωσης είναι πολύ σημαντική καθώς υπάρχουν περιπτώσεις που μπορεί να επιφέρει το επιθυμητό αποτέλεσμα χωρίς τη διαδικασία της μετάλλαξης. Υπάρχουν επίσης απόψεις που αναφέρουν ότι στην περίπτωση που η διαδικασία αυτή δεν αποτελεί μέρος ενός γενετικού αλγορίθμου μειώνεται σημαντικά η απόδοση του.

Ένα από πιο σημαντικά κομμάτια της διαδικασίας αυτής είναι ένας συντελεστής που ονομάζεται πιθανότητα διασταύρωσης. Ο συντελεστής αυτός συνηθίζεται να συμβολίζεται σαν p_c και καθορίζει το ποσοστό των χρωμοσωμάτων που θα αντικατασταθούν με καινούργια. Τα χρωμοσώματα που δεν επιλέγονται για διασταύρωση θα περάσουν στην επόμενη γενιά στη μορφή που είναι ενώ αυτά που επιλέχθηκαν θα ανταλλάξουν πληροφορίες με σκοπό η οι απόγονοι τους να συνδυάζουν τα καλύτερα χαρακτηριστικά τους. Η πιθανότητα μετάλλαξης δεν είναι καλό να έχει ακραίες τιμές καθώς αυξάνεται έτσι η πιθανότητα να μην συγκλίνει σωστά ο αλγόριθμος.

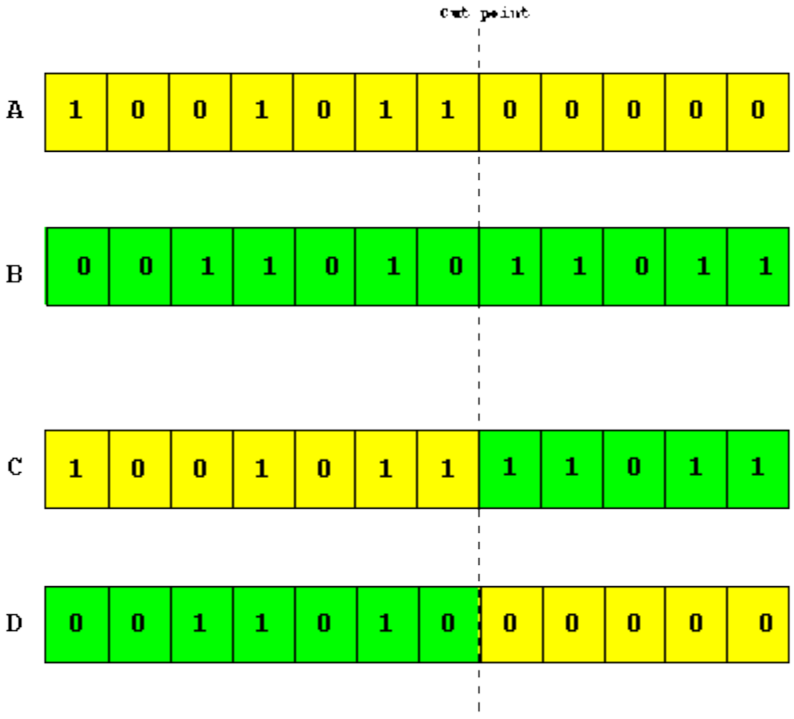
Οι μέθοδοι διασταύρωσης που θα αναλυθούν παρακάτω είναι:

- Διασταύρωση ενός σημείου
- Διασταύρωση δύο σημείων
- Ομοιόμορφη διασταύρωση
- Δεκαδική διασταύρωση

Διασταύρωση ενός σημείου

Η μέθοδος αυτή είναι η πιο απλή από όλες τις μεθόδους διασταυρώσεις και αποτελείται από δύο βήματα

- Επιλέγεται τυχαία η θέση που θα γίνει η διασταύρωση στα χρωμοσώματα
- Γίνεται ανταλλαγή του δεξιού τμήματος του πρώτου χρωμοσώματος με το αριστερό του δεύτερου και αντίστροφα,πριν και μετά το σημείο που επιλέχθηκε

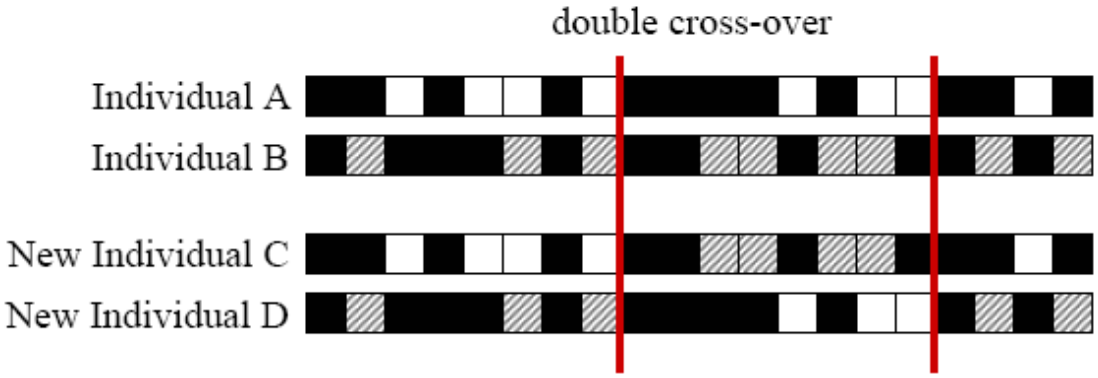


Εικόνα 19:Παράδειγμα διασταύρωσης ενός σημείου

Η μέθοδος αυτή χρησιμοποιείται κυρίως σε ακέραια χρωμοσώματα όμως λόγω της απλής της υλοποίησης μπορεί να χρησιμοποιηθεί και σε διαφορετικής μορφής χρωμοσώματα

Διασταύρωση δύο σημείων

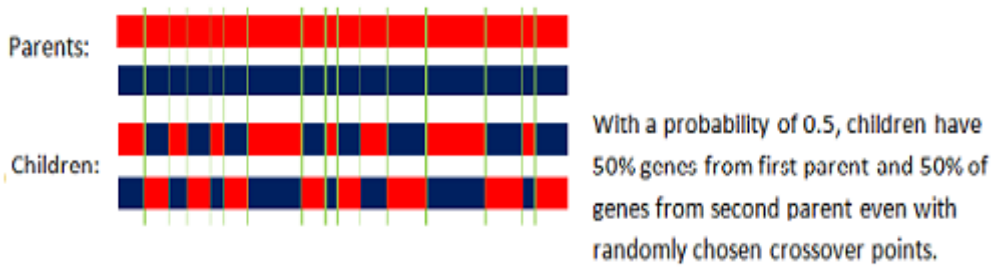
Είναι παρόμοια με την διασταύρωση ενός σημείου με την διαφορά ότι επιλέγονται δύο θέσεις για την διασταύρωση αντί για ένα. Γενικά χρησιμοποιείται πολύ σπάνια καθώς δεν έχει αποδειχτεί εάν η επιλογή δύο σημείων προσφέρει καλύτερα αποτελέσματα από την επιλογή ενός.



Εικόνα 20:Παράδειγμα διασταύρωσης δύο σημείων

Ομοιόμορφη διασταύρωση

Αυτή η μέθοδος δίνει 50% πιθανότητα σε κάθε γονέα να περάσει ένα γονίδιο στον απόγονο. Αυτή η διαδικασία εκτελείται για κάθε γονίδιο έτσι υπάρχει πάντα και η πιθανότητα ο απόγονος να παίρνει όλα τα χαρακτηριστικά του από τον έναν από τους γονείς αν και καθώς η επιλογή είναι εντελώς τυχαία είναι πολύ σπάνιο να συμβεί. Η μέθοδος αυτή δουλεύει κυρίως με άτομα με τον ίδιο αριθμό γονιδίων παρόλα αυτά μπορεί να δουλέψει και με ανόμοια άτομα με κατάλληλο προγραμματισμό.



Εικόνα 21:Παράδειγμα ομοιόμορφης διασταύρωσης

Δεκαδική διασταύρωση

Αυτή η μέθοδος είναι αρκετά διαφορετική από τις προηγούμενες μεθόδους και έχει παρατηρηθεί ότι μπορεί να είναι και πιο

αποτελεσματική. Ο αλγόριθμος αφού επιλέξει τους γονείς παράγονται δύο απόγονοι μέσω των μαθηματικών σχέσεων:

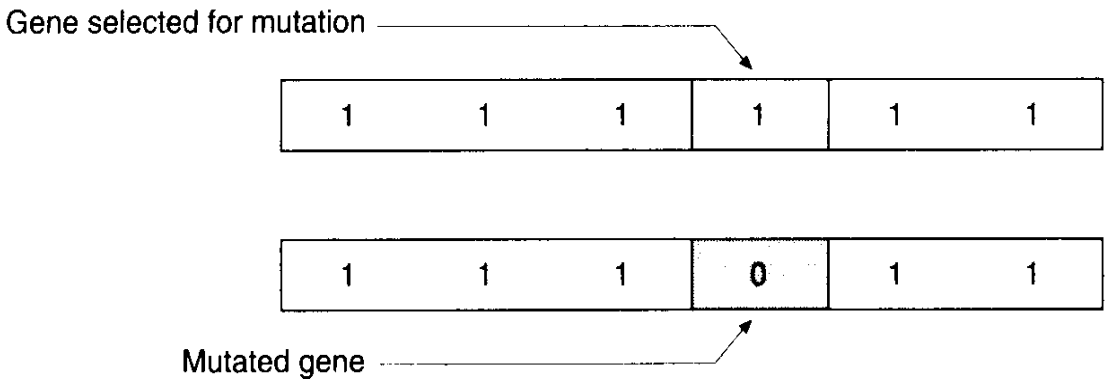
$$c1 = p1 + a * (p1 - p2)$$

$$c2 = p2 + a * (p2 - p1)$$

Αυτές οι σχέσεις υπολογίζονται για κάθε γονίδιο. Η μεταβλητή a παίρνει τιμές μέσα από το διάστημα [-d,1+d] όπου η ενδεικτική τιμή για το d=0,25. [1]

Μετάλλαξη

Η μετάλλαξη είναι ίσως η λιγότερη σημαντική διαδικασία ενός γενετικού αλγορίθμου αλλά είναι αρκετά χρήσιμη. Βασίζεται στην πιθανότητα μετάλλαξης ή ρυθμό μετάλλαξης τον οποίο πιο συχνά συμβολίζουμε ως p_m. Όσο μεγαλύτερος είναι ο ρυθμός μετάλλαξης τόσο μεγαλύτερες οι πιθανότητες για τα γονίδια να υποστούν μετάλλαξη. Αυτό γίνεται για όλα τα μέλη του πληθυσμού και κάθε γονίδιο έχει πιθανότητα να μεταλλαχθεί. Αξίζει να σημειωθεί ότι εάν $p_m = 1$ τότε όλα τα γονίδια θα μεταλλαχθούν και αυτό ισοδυναμεί με τυχαίο ψάξιμο. Συνήθως προτιμάται ο ρυθμός μετάλλαξης να είναι αρκετά μικρός.



Εικόνα 22:Παράδειγμα μετάλλαξης

Μια αρκετά αποτελεσματική τεχνική είναι ο ρυθμός μετάλλαξης να είναι μεγάλος στην αρχή του αλγορίθμου και στην συνέχεια σταδιακά

να μειώνεται. Αυτό εξασφαλίζει ότι αρχικά ερευνάται μεγαλύτερο μέρος του χώρου και μειώνεται ο ρυθμός μετάλλαξης καθώς ο αλγόριθμος συγκλίνει προς την λύση του προβλήματος. [2]

Συνθήκη τερματισμού

Η συνθήκη τερματισμού αποτελεί πολύ σημαντικό κομμάτι του γενετικού αλγορίθμου. Η ιδανική περίπτωση τερματισμού του αλγορίθμου είναι:

- να μην γίνονται επαναλήψεις που δεν βελτιώνουν ή γενικά δεν οδηγούν σε κάποιο αποτέλεσμα
- να τελειώνει η εκτέλεση του αλγορίθμου όταν βρεθεί η λύση του προβλήματος

Αυτό είναι υπερβολικά δύσκολο να επιτευχθεί καθώς δεν είναι γνωστή η λύση του προβλήματος ούτε πόσες γενιές θα χρειαστούν για να φτάσει σε αυτήν ο αλγόριθμος.

Η πιο απλή συνθήκη τερματισμού είναι να οριστεί από το χρήστη ένας μέγιστος αριθμός επαναλήψεων. Η συνθήκη αυτή μπορεί να οδηγήσει σε διάφορα προβλήματα όπως:

- αν όταν ολοκληρωθούν οι επαναλήψεις ο αλγόριθμος δεν έχει βρει τη λύση τότε ο αλγόριθμος τερματίζει πρόωρα
- αν η λύση βρεθεί σε μια από τις πρώτες επαναλήψεις τότε οι υπόλοιποι υπολογισμοί γίνονται άσκοπα

Γι'αυτούς τους λόγους συνηθίζεται να υπάρχει και κάποια άλλη συνθήκη μαζί με τον αριθμό επαναλήψεων ώστε να γίνεται έλεγχος για την πορεία του αλγορίθμου.

Σε περίπτωση που η λύση του προβλήματος είναι γνωστή μπορεί να χρησιμοποιηθεί σαν κριτήριο τερματισμού. Η μέθοδος αυτή ονομάζεται προσέγγιση στόχου αλλά μπορεί να χρησιμοποιηθεί μόνο σε συγκεκριμένα προβλήματα.

Μία τρίτη μέθοδος τερματισμού είναι το λεγόμενο κριτήριο ομοιότητας. Σε αυτό υπολογίζεται η μέση απόσταση μεταξύ των χρωμοσωμάτων και αν είναι μικρότερη από ένα προκαθορισμένο όριο τότε ο αλγόριθμος τερματίζει. Η μέθοδος αυτή χρησιμοποιείτε

αρκετά συχνά καθώς δεν εξαρτάται ούτε από το πρόβλημα ούτε από την κωδικοποίηση αλλά έχει αρκετά μειονεκτήματα όπως:

- είναι αρκετά αργό ειδικότερα σε προβλήματα με μεγάλο αριθμό πληθυσμού
- μπορεί να έχει βρεθεί η λύση αλλά να μην έχει ικανοποιηθεί η συνθήκη με αποτέλεσμα να αργήσει να τερματίσει ο αλγόριθμος
- υπάρχει περίπτωση να μην ικανοποιηθεί αρκετά γρήγορα με αποτέλεσμα την καθυστέρηση του αλγορίθμου

Τελευταία μέθοδος τερματισμού είναι ο υπολογισμός καλύτερου-χειρότερου. Με αυτή τη μέθοδο σε κάθε επανάληψη υπολογίζεται η διαφορά μεταξύ του καλύτερου και του χειρότερου ατόμου στο πληθυσμό και αν είναι μικρότερη από ένα όριο ο αλγόριθμος τερματίζει.Μοιάζει αρκετά με την μέθοδο με το κριτήριο ομοιότητας αλλά παρότι είναι πιο γρήγορη δεν αποτρέπει τα προβλήματα που υπήρχαν και πριν.

3.5 Κωδικοποίηση

Η κωδικοποίηση έχει ως στόχο να γίνει τέτοια αναπαράσταση των δεδομένων ώστε να μπορούν να είναι επεξεργάσιμα από τον υπολογιστή και από τις διαδικασίες του αλγορίθμου. Είναι από τις πιο σημαντικές διαδικασίες ενός γενετικού αλγορίθμου καθώς μη σωστή ή ακριβής κωδικοποίηση μπορεί να οδηγήσει σε μη σωστή ολοκλήρωση του αλγορίθμου και λανθασμένα αποτελέσματα. Οι μορφές κωδικοποίησης που υπάρχουν είναι:

- η δυαδική κωδικοποίηση
- η κωδικοποίηση gray
- η αναπαράσταση ακεραίων
- η δεκαδική αναπαράσταση
- η αλφαβητική κωδικοποίηση

Δυαδική κωδικοποίηση

Η δυαδική κωδικοποίηση είναι η πρώτη κωδικοποίηση που εφαρμόστηκε στους γενετικούς αλγορίθμους και είναι η μόνη για την οποία έχουν αποδειχθεί κριτήρια σύγκλισης. Είναι επίσης και η απλούστερη από όλες τις άλλες μεθόδους. Οι τρόποι αναπαράστασης για αυτήν την κωδικοποίηση είναι:

- το χρωμόσωμα είναι ένας ακέραιος και τα γονίδια τα δυαδικά ψηφία που τον αποτελούν
- το χρωμόσωμα να αποτελείτε από μία σειρά δυαδικών αριθμών

Το πρόβλημα που παρουσιάζει αυτή η κωδικοποίηση είναι πως μία μικρή αλλαγή σε ένα ψηφίο μπορεί να επιφέρει μεγάλες αλλαγές στο περιεχόμενο. Αυτό γίνεται καθώς οι συμβολισμοί στο δυαδικό σύστημα είναι όλοι αρκετά παρόμοιοι μεταξύ τους και ένα λάθος μπορεί να αλλάξει τελείως το αποτέλεσμα.

Κωδικοποίηση Gray

Η κωδικοποίηση gray χρησιμοποιείται πολύ τόσο στους γενετικούς όσο και στην ασφάλεια πληροφοριών καθώς καταφέρνει να αντιμετωπίσει ικανοποιητικά το πρόβλημα που αναφέρθηκε προηγουμένως μέσω τις ιδιότητας που προσφέρει ότι δύο γειτονικοί ακέραιοι απέχουν μεταξύ τους 1bit. Το χαρακτηριστικό που βοηθάει σε αυτό είναι ουσιαστικά το σημείο το οποίο θα ξεκινήσουμε την κωδικοποίηση με αποτέλεσμα να υπάρχουν πολλές διαφορετικές κωδικοποιήσεις.

Αναπαράσταση ακεραίων

Αυτή η αναπαράσταση είναι σχετικά εύκολα να υλοποιηθεί αλλά υπάρχουν διάφορες περιπτώσεις που χρειάζονται περισσότερη προσοχή όπως στην περίπτωση που οι αριθμοί που χρειάζονται κωδικοποίηση μπορούν να κωδικοποιηθούν με κάποια δύναμη του 2 ή όχι. Ανάλογα την περίπτωση αλλάζει και ο τρόπος αντιμετώπισης.

Δεκαδική αναπαράσταση

Αυτή η αναπαράσταση είναι πιο πολύπλοκη καθώς δεν υπάρχει άμεση αντιστοίχιση ανάμεσα σε δεκαδικούς και ακέραιους αριθμούς. Έχει δύο τρόπους αναπαράστασης

- Τη δυαδική κωδικοποίηση. Με αυτό τον τρόπο αναπαριστώνται οι ακέραιοι δεκαδικά όμως χρειάζονται πολλά bit και χρόνος για να επιτευχθεί αυτό
- Την άμεση κωδικοποίηση. Αντι για χρήση διανυσμάτων χρησιμοποιούνται άμεσα οι δεκαδικοί αριθμοί. Βέβαια αυτό επιφέρει λιγότερη ακρίβεια και δεν υπάρχει θεωρητική απόδειξη για την σύγκλιση του αλγορίθμου.

Αλφαβητική αναπαράσταση

Στην περίπτωση που οι μεταβλητές για κωδικοποίηση είναι αλφαριθμητικές τότε συμβολίζουμε κάθε χαρακτήρα με ένα μοναδικό συνδυασμό ακεραίων. Στο τέλος του γενετικού θα πρέπει να γυρίσει το αποτέλεσμα στην αρχική του μορφή. [1]

4.Η προτεινόμενη εφαρμογή

4.1 Εξισώσεις τεχνητού νευρωνικού δικτύου

Στην εφαρμογή που αναπτύξαμε γίνετε εκπαίδευση ενός νευρωνικού δικτύου μέσω γενετικού αλγορίθμου. Η έξοδος του νευρωνικού δικτύου υπολογίζετε μέσω της συνάρτησης:

$$N(x,w) = \sum_{i=1}^{H} w_{(d+2)i-(d+1)} \sigma\left(\sum_{j=1}^{d} (w_{(d+2)i-(d+1)+j}) + w_{(d+2)i}\right)$$

όπου:

w:είναι τα βάρη του προβλήματος τα οποία στο πρόβλημα μας παίζουν το ρόλο των χρωμοσωμάτων που θα μεταλλάσσονται από το γενετικό

x:είναι τα δεδομένα του προβλήματος

H:οι κόμβοι του νευρώνικού

d:οι διαστάσεις του προβλήματος

N:η έξοδος του νευρωνικού [8]

Αφού υπολογιστεί η έξοδος του νευρωνικού δικτύου υπολογίζουμε την καταλληλότητα του αποτελέσματος με την παρακάτω εξίσωση:

$$f = \sum_{i=1}^{M} (N(x_i) - y_i)^2$$

όπου:

f:ο δείκτης καταλληλότητας

y:το επιθυμητό αποτέλεσμα

4.2 Δεδομένα πειραμάτων

Στην εφαρμογή που αναπτύχθηκε έγιναν πειράματα με δύο σύνολα δεδομένων. Το BK και το BL. Ο αλγόριθμος είναι κωδικοποιημένος έτσι ώστε να εφαρμόζονται όλα τα σύνολα δεδομένων με έναν x αριθμό εισόδων και μία έξοδο.

Το σύνολο δεδομένων BK αποτελείτε από 4 εισόδους και μία έξοδο για τα δεδομένα που χρησιμοποιήθηκαν. Τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση μέσω του γενετικού αλγορίθμου είναι:

X1	X2	X3	X4	Y
0.088800	201.000000	36.020000	28.000000	0.640788
0.098300	191.000000	40.710000	30.000000	0.623918
0.127600	196.000000	38.400000	28.000000	0.613616
0.167100	201.000000	34.100000	31.000000	0.633323
0.049400	193.000000	32.380000	32.000000	0.596745
0.110700	196.000000	35.220000	25.000000	0.478650
0.182800	191.000000	29.540000	28.000000	0.419230
0.162700	196.000000	31.350000	28.000000	0.595402
0.156300	193.000000	34.560000	32.000000	0.548522
0.268100	183.000000	39.530000	27.000000	0.574201
0.130000	188.000000	30.770000	26.000000	0.359062
0.089600	198.000000	25.670000	30.000000	0.407883
0.207100	178.000000	36.220000	30.000000	0.372201
0.224400	185.000000	36.550000	23.000000	0.452523
0.105800	191.000000	28.350000	28.000000	0.494177
0.157700	193.000000	31.070000	25.000000	0.410869
0.125600	196.000000	27.870000	29.000000	0.694386
0.107000	198.000000	24.310000	34.000000	0.358017
0.134300	193.000000	31.260000	28.000000	0.421170
0.238300	185.000000	35.250000	26.000000	0.329651

0.216400	193.000000	24.490000	32.000000	0.237683
0.227000	191.000000	31.720000	27.000000	0.406987
0.118800	191.000000	22.740000	24.000000	0.372947
0.194000	193.000000	20.620000	27.000000	0.297850
0.249500	185.000000	30.460000	25.000000	0.467901
0.206900	170.000000	33.840000	30.000000	0.401911
0.087700	193.000000	21.670000	26.000000	0.623470
0.101000	193.000000	21.790000	24.000000	0.474769
0.094200	201.000000	20.170000	26.000000	0.435802
0.107100	196.000000	24.280000	24.000000	0.223350
0.213000	188.000000	21.590000	30.000000	0.377426
0.104300	196.000000	16.300000	23.000000	0.256793
0.113000	191.000000	23.010000	25.000000	0.255151
0.147700	196.000000	20.310000	31.000000	0.460436
0.212700	188.000000	14.570000	37.000000	0.131084
0.089800	196.000000	13.370000	34.000000	0.342938
0.214600	188.000000	20.510000	24.000000	0.526575
0.152800	191.000000	16.360000	33.000000	0.364586
0.156000	191.000000	16.030000	23.000000	0.162735
0.234800	188.000000	24.270000	26.000000	0.168110
0.123900	180.000000	17.760000	26.000000	0.418035
0.217800	185.000000	13.310000	25.000000	0.210660
0.177600	193.000000	17.460000	27.000000	0.147059
0.166800	185.000000	14.380000	35.000000	0.208420
0.107200	188.000000	12.120000	31.000000	0.427292
0.182100	185.000000	12.630000	25.000000	0.223052
0.199400	188.000000	20.060000	27.000000	0.387280
0.101400	193.000000	13.810000	32.000000	0.000000

Τα δεδομένα του ΒΚ που χρησιμοποιήθηκαν για το τελικό αποτέλεσμα του νευρωνικού δικτύου είναι:

X1	X2	X3	X4	Y
0.139900	198.000000	39.320000	30.000000	1.000000
0.074700	198.000000	38.800000	26.000000	0.504778
0.190600	193.000000	36.200000	30.000000	0.549866
0.106100	191.000000	36.750000	27.000000	0.586742
0.244600	185.000000	38.430000	29.000000	0.360406
0.167000	203.000000	33.540000	24.000000	0.474321
0.248500	188.000000	35.010000	27.000000	0.406091
0.122700	198.000000	36.670000	29.000000	0.495073
0.124000	185.000000	33.880000	24.000000	0.608391
0.146100	191.000000	35.590000	30.000000	0.525530
0.231500	191.000000	38.010000	28.000000	0.327710
0.252100	183.000000	31.730000	29.000000	0.618394
0.100700	193.000000	28.810000	34.000000	0.705434
0.106700	196.000000	35.600000	23.000000	0.408032
0.195600	188.000000	35.280000	32.000000	0.401165
0.140300	198.000000	33.500000	23.000000	0.488653
0.123600	196.000000	26.700000	34.000000	0.421917
0.343700	185.000000	34.910000	31.000000	0.407883
0.232600	185.000000	33.530000	27.000000	0.479098
0.232700	185.000000	36.520000	32.000000	0.481636

0.058600	196.000000	22.180000	23.000000	0.361302
0.100600	198.000000	22.870000	30.000000	0.284413
0.148500	198.000000	23.570000	27.000000	0.224545
0.164900	188.000000	27.900000	25.000000	0.329352
0.237800	185.000000	32.380000	27.000000	0.241714
0.159200	191.000000	25.750000	31.000000	0.272469
0.208400	185.000000	27.830000	25.000000	0.346969
0.055000	193.000000	29.070000	31.000000	0.224395
0.072800	193.000000	19.240000	27.000000	0.444909
0.277100	180.000000	27.070000	28.000000	0.242013
0.052800	196.000000	18.950000	22.000000	0.573903
0.135600	193.000000	13.270000	31.000000	0.088385
0.131700	188.000000	17.460000	33.000000	0.121380
0.218700	191.000000	21.950000	28.000000	0.211108
0.254700	160.000000	34.550000	28.000000	0.194237
0.159100	191.000000	22.000000	24.000000	0.311884
0.187100	183.000000	19.780000	28.000000	0.426396
0.162300	180.000000	18.490000	28.000000	0.270976
0.160800	185.000000	17.410000	26.000000	0.285160
0.080500	193.000000	13.670000	25.000000	0.417289
0.188000	180.000000	12.240000	30.000000	0.311287
0.116700	196.000000	12.000000	24.000000	0.309645
0.261700	185.000000	24.460000	27.000000	0.238280
0.170600	170.000000	17.000000	25.000000	0.517468
0.155400	183.000000	11.580000	24.000000	0.239176
0.228200	185.000000	10.080000	24.000000	0.117647
0.177800	185.000000	18.560000	23.000000	0.180502
0.186300	185.000000	11.810000	23.000000	0.330994

Το σύνολο δεδομένων BL έχουν 7 εισόδους και μία έξοδο. Τα δεδομένα για την εκπαίδευση του γενετικού είναι:

X1	X2	X3	X4	X5	X6	X7	Y
25.000000	2.000000	10.000000	1.500000	0.000000	6.000000	5.700000	0.049744
24.000000	2.000000	10.000000	1.500000	0.000000	10.000000	17.560000	0.338942
40.000000	2.000000	10.000000	2.500000	0.000000	6.000000	16.670000	0.317240
37.000000	2.000000	10.000000	2.500000	0.000000	10.000000	12.040000	0.204340
13.000000	2.000000	30.000000	2.500000	0.000000	10.000000	22.390000	0.092782
4.000000	2.000000	30.000000	2.500000	2.000000	6.000000	23.850000	0.104609
19.000000	6.000000	10.000000	1.500000	2.000000	6.000000	39.740000	0.879785
38.000000	6.000000	10.000000	2.500000	2.000000	6.000000	17.200000	0.330163
39.000000	6.000000	30.000000	1.500000	0.000000	6.000000	129.450000	0.962936
8.000000	6.000000	30.000000	1.500000	0.000000	0.000000	107.380000	0.783589
6.000000	6.000000	30.000000	2.500000	0.000000	6.000000	100.430000	0.727018
20.000000	6.000000	30.000000	2.500000	0.000000	0.000000	109.280000	0.798951
10.000000	6.000000	30.000000	2.500000	2.000000	6.000000	106.460000	0.776030
32.000000	6.000000	30.000000	2.500000	2.000000	0.000000	134.010000	1.000000
1.000000	4.000000	20.000000	2.000000	1.000000	8.000000	10.780000	0.042185
5.000000	4.000000	20.000000	2.000000	1.000000	8.000000	9.840000	0.030724
12.000000	4.000000	20.000000	2.000000	1.000000	8.000000	12.330000	0.061083
18.000000	4.000000	20.000000	2.000000	1.000000	8.000000	7.320000	0.000000
29.000000	4.000000	20.000000	2.000000	1.000000	8.000000	7.910000	0.007193
36.000000	4.000000	20.000000	2.000000	1.000000	8.000000	15.580000	0.100707

Το σύνολο δεδομένων BL που χρησιμοποιήθηκε για το αποτέλεσμα του νευρωνικού είναι:

X1	X2	X3	X4	X5	X6	X7	Y
30.000000	2.000000	10.000000	1.500000	2.000000	6.000000	11.280000	0.185808
2.000000	2.000000	10.000000	1.500000	2.000000	10.000000	8.390000	0.115338
16.000000	2.000000	10.000000	2.500000	2.000000	6.000000	9.220000	0.135577
22.000000	2.000000	10.000000	2.500000	2.000000	10.000000	3.940000	0.006828
33.000000	2.000000	30.000000	1.500000	0.000000	6.000000	27.020000	0.130334
17.000000	2.000000	30.000000	1.500000	0.000000	10.000000	19.460000	0.068886
28.000000	2.000000	30.000000	1.500000	2.000000	6.000000	18.540000	0.061448
27.000000	2.000000	30.000000	1.500000	2.000000	10.000000	25.700000	0.119605
14.000000	2.000000	30.000000	2.500000	0.000000	6.000000	19.020000	0.065350
21.000000	2.000000	30.000000	2.500000	2.000000	10.000000	30.120000	0.155572
23.000000	6.000000	10.000000	1.500000	0.000000	6.000000	13.420000	0.237991
35.000000	6.000000	10.000000	1.500000	0.000000	10.000000	34.260000	0.746159
34.000000	6.000000	10.000000	1.500000	2.000000	10.000000	10.600000	0.169227
31.000000	6.000000	10.000000	2.500000	0.000000	6.000000	28.890000	0.615216
9.000000	6.000000	10.000000	2.500000	0.000000	10.000000	35.610000	0.779078
15.000000	6.000000	10.000000	2.500000	2.000000	10.000000	6.000000	0.057059
26.000000	6.000000	30.000000	1.500000	2.000000	6.000000	111.660000	0.818337
11.000000	6.000000	30.000000	1.500000	2.000000	0.000000	109.100000	0.797488
3.000000	4.000000	20.000000	2.000000	1.000000	8.000000	9.390000	0.025238
7.000000	4.000000	20.000000	2.000000	1.000000	8.000000	13.940000	0.080712

4.3 Αναλυτική επεξήγηση του αλγορίθμου

4.3.1 Δηλώσεις μεταβλητών και αρχικοποιήσεις

Η εφαρμογή ξεκινάει αφού δηλωθούν αρχικά όλες τις βιβλιοθήκες της C++ που θα χρησιμοποιηθούν. Επίσης δηλώνεται και ο τύπος vector που θα χρησιμοποιηθεί αρκετά στην εφαρμογή. Οι vector που θα χρησιμοποιηθούν θα περιέχουν δεδομένα τύπου double για να αποθηκεύονται εκεί τα δεδομένα του προβλήματος . Η δήλωση αυτή γίνεται πριν τη δήλωση της main σαν global τύπος μεταβλητής με τον εξής τρόπο:

typedef vector<double> Data;

Καθώς το κύριο πρόγραμμα αρχίζει να εκτελείται δηλώνονται όλες οι μεταβλητές που θα χρησιμοποιηθούν στην εφαρμογή όπως οι πίνακες για τα δεδομένα και οι διαστάσεις του προβλήματος.

Το πρώτο βήμα είναι να διαβαστούν τα δεδομένα του προβλήματος και να αποθηκευτούν σε πίνακες. Αυτό γίνεται με την βοήθεια της βιβλιοθήκης fstream. Η fstream χρησιμοποιείται για να διαβάζονται δεδομένα από αρχεία ή να τροποποιούνται. Καθώς στην εφαρμογή χρησιμοποιούνται δύο σύνολα δεδομένων αρχικά εμφανίζεται ένα μενού όπου ζητάει από το χρήστη να επιλέξει ένα από τα δύο. Αφού ο χρήστης επιλέξει πιο σύνολο δεδομένων θα χρησιμοποιηθεί τότε η fstream

δημιουργεί το εικονικό αρχείο και ξεκινάει να τραβάει τα δεδομένα. Αυτό γίνεται αρχικά με την εντολή

ifstream myfile ("όνομα αρχείου" , ios::binary);

και για την προσπέλαση των δεδομένων χρησιμοποιείται η εντολή

myfile >> μεταβλητή αποθήκευσης;

η οποία διαβάζει από το αρχείο αγνοώντας τους κενούς χαρακτήρες και τους χαρακτήρες αλλαγής γραμμής. Έτσι επιτυγχάνεται η αποθήκευση των δεδομένων του προβλήματος καθώς και των διαστάσεων του. Με το πέρας αυτής της διαδικασίας το αρχείο κλείνει με την εντολή

myfile.close();

Στο επόμενο βήμα υπολογίζεται το μέγεθος του προβλήματος το οποίο επηρεάζει και μεγάλο μέρος του νευρωνικού δικτύου όπως τα margin και τα βάρη. Αυτό είναι ανάλογο με τις διαστάσεις του συνόλου δεδομένου και των κόμβων επεξεργασίας του νευρωνικού. Υπολογίζεται μέσω της εντολής

problem_dimension = (d+2)*h;

Οι κόμβοι επεξεργασίας δίνονται από τον χρήστη κάθε φορά που εκτελείται η εφαρμογή.

Στην συνέχεια δημιουργούνται οι πίνακες με τα margin ανάλογα με το μέγεθος του προβλήματος και αρχικοποιούνται.

Ο χρήστης στην συνέχεια καλείται να δώσει τον αριθμό των χρωμοσωμάτων του προβλήματος. Αυτό βοηθάει ώστε να υπάρχουν μεγαλύτερα περιθώρια για πειραματισμό.

Σύμφωνα με τον αριθμό των χρωμοσωμάτων καθώς και με το μέγεθος του προβλήματος δημιουργείται και αρχικοποιείται ο πίνακας genome ο οποίος εξυπηρετεί ως τα συναπτικά βάρη του νευρωνικού. Η αρχικοποίηση του πίνακα γίνεται με την βοήθεια της συνάρτησης makeChromosome η οποία χρησιμοποιεί τη συνάρτηση τυχαίας αρχικοποίησης drand48() και των margin.Γίνεται χρήση και της ειδικής συνάρτησης srand( ) η οποία αρχικοποιεί την drand( ).

double makeChromosome(Data &x, Data l, Data r)

```
{
   srand(1);
   for (int i = 0; i < x.size(); i++)
      x[i] = l[i]+(r[i] - l[i]) * drand48();
}
```

Επόμενο βήμα είναι η δημιουργία του πίνακα fitness ο οποίος είναι ανάλογος με τα δεδομένα του προβλήματος και θα χρησιμοποιηθεί αργότερα από τον γενετικό αλγόριθμο ώστε να υπολογιστεί η καταλληλότητα του αποτελέσματος

```
fitness.resize(genome_count);
```

Τελευταίο κομμάτι για τις αρχικοποιήσεις είναι η δήλωση των αριθμό των γενιών για τις οποίες θα τρέξει ο αλγόριθμος. Η αρχικοποίηση αυτή γίνεται από τον χρήστη για περισσότερη ευελιξία στον κώδικα

```
do
{
        printf("MAXIMUM NUMBER OF GENERATIONS ? ");
        scanf("%d",&maxgenerations);
}while(maxgenerations<=0);
```

και γίνεται σε μία επανάληψη για να αποφευχθεί η πιθανότητα ο χρήστης να δώσει αρνητικό ή μηδενικό αριθμό.

4.3.2 Εκπαίδευση με τη χρήση γενετικού αλγορίθμου

Το μεγαλύτερο κομμάτι της εφαρμογής εκτελείται μέσα σε μία επανάληψη που εκτελείται ανάλογα με τον αριθμό των γενιών. Για κάθε μία από τις γενιές θα εκτελούνται κάποιες συγκεκριμένες διαδικασίες για την κατάλληλη εκπαίδευση του νευρωνικού δικτύου. Πρώτο βήμα είναι να υπολογιστεί η καταλληλότητα των βαρών η οποία θα γίνει μέσω των συναρτήσεων που είδαμε σε προηγούμενο κεφάλαιο. Αυτό το κάνει η συνάρτηση calculateFitness. Για τον υπολογισμό της συνάρτησης N(x,w) χρησιμοποιήθηκαν τρεις βοηθητικές μεταβλητές,η sum η first και η

second καθώς και η συνάρτηση sig. Η μεταβλητή first χρησιμοποιείτε για τον υπολογισμό του πρώτο αθροίσματος της συνάρτησης δηλαδή του όρου $\sum_{i=1}^{H} w_{(d+2)i-(d+1)}$ και η second υπολογίζει το δεύτερο άθροισμα δηλαδή τον όρο $\sum_{j=1}^{d}\left(w_{(d+2)i-(d+1)+j}\right) + w_{(d+2)i}$ και το αποτέλεσμα αυτό περνάει μέσα από τη συνάρτηση sig ώστε να υπολογιστεί η σιγμοειδής,η οποία στο συγκεκριμένο πρόβλημα είναι η σ(x)=1/(1+e^{-x}).Η συνάρτηση sig συντάσσετε ως εξής

```
double sig(double x) {
   return (1 / (1 + pow(M_E, -x)));
}
```

παίρνει δηλαδή σαν όρισμα τη μεταβλητή second και επιστρέφει το αποτέλεσμα της σιγμοειδής. Τέλος χρησιμοποιούμε την μεταβλητή sum για να υπολογίσουμε το τελικό αποτέλεσμα. Το τελευταίο βήμα για τον υπολογισμό της καταλληλότητας γίνεται μέσω της συνάρτησης problemFitness,η οποία επιστρέφει το αποτέλεσμα της διαφοράς των τετραγώνων του N(x_i) και του αποτελέσματος που που θέλουμε να βρούμε μετά την εκπαίδευση και στη συνέχεια προσθέτουμε το αποτέλεσμα για με το προηγούμενο ώστε να υπολογίσουμε το άθροισμα τους. Η συνάρτηση calculateFitness συντάσσεται ως εξής:

```
Data calculateFitness(int d,vector<Data> genome, vector<double>
&fitness,vector<Data> data, Data l, Data r, int genome_count,int patterns,int h)
{
   for (int i = 0; i < genome_count; i++) {
      fitness[i] = 0;
      for (int j = 0; j < patterns; j++)
      {
         double sum = 0;
         for (int k = 0; k < h; k++)
         {
            double first = genome[i][(d + 2) * k - (d + 1) - 1];
            double second = 0;
```

```
            for (int m = 0; m < d; m++) {
                second = second + genome[i][(d + 2) * k - (d + 1) + m - 1] * data[j][m-1];
            }
            second = second + genome[i][(d + 2) * k - 1];
            double sigmoid = sig(second);
            sum = sum + first*sigmoid;
        }
        double actual_output = sum;
        double desired_output = data[j][d];
        fitness[i] = fitness[i] + problemFitness(genome[i], l, r, desired_output,
actual_output);
      }
   }
    return fitness;
}
```

και τέλος η problemFitness ως εξής:

```
double problemFitness(Data x, Data l, Data r, double desired_output, double
actual_output)
{
    if (!isPointIn(x, l, r)) return 1e+100;
   return pow(desired_output - actual_output, 2);
}
```

Η συνάρτηση isPointIn χρησιμοποιείτε για να ελεγχθεί ότι τα βάρη έχουν δοθεί σωστά.

Ακολουθεί η ταξινόμηση των χρωμοσωμάτων από το βέλτιστο προς το χειρότερο. Τα χρωμοσώματα συγκρίνονται ως προς την καταλληλότητα τους και ταξινομούνται ανάλογα. Η διαδικασία αυτή γίνεται με την συνάρτηση sortChromosomes που παίρνει σαν ορίσματα τους πίνακες με τα βάρη και τις καταλληλότητες τους και τους επιστρέφει

ταξινομημένους ως προς φθίνουσα σειρά. Η συνάρτηση συντάσσεται ως εξής:

```
void	sortChromosomes(vector<Data> &genome,vector<double> &fitness)
{
	Data temp;
	temp.resize(genome[0].size());
	for(int i=0;i<genome.size();i++)
	{
		for(int j=0;j<genome.size()-1;j++)
		{
			if(fitness[j+1]<fitness[j])
			{
				double dtemp=fitness[j];
				fitness[j]=fitness[j+1];
				fitness[j+1]=dtemp;
				temp=genome[j];
				genome[j]=genome[j+1];
				genome[j+1]=temp;
			}
		}
	}
}
```

Το επόμενο κομμάτι είναι οι συναρτήσεις που αποτελούν τον γενετικό αλγόριθμο. Αρχικά εκτελείται η διαδικασία της διασταύρωσης των χρωμοσωμάτων. Η συνάρτηση που κάνει αυτή την διαδικασία είναι η crossoverChromosomes η οποία παίρνει σαν εισόδους τα δεδομένα του πίνακα με τα βάρη και του πίνακα με τις καταλληλότητες των χρωμοσωμάτων. Με την βοήθεια μιας σταθεράς της pc που έχει την τιμή 0.99 δημιουργούμε τον πίνακα για τα παιδία που θα δημιουγηθούν. Στη

συνέχεια δημιουργείται ένας πίνακας δύο θέσεων για να μπούν οι γονείς που θα επιλεγούν από την διαδικασία μετάλαξης. Για την εφαρμογή επιλέκτηκε η διαδικάσια tournament για την επιλογή των γονέων. Επιλέγονται μέσω από την διαδικασία αυτή δύο γονείς οι οποίοι μέσω της συνάρτησης makeChild δημιουργούν δύο νέα παιδιά. Η συνάρτηση αυτή δέχεται τρία ορίσματα ,το παιδί το οποίο θα δημιουργηθεί και τους δύο γονείς. Για την δημιουργία ενός νέου παιδιού υπάρχει η επιλογή για δύο μεθόδους την uniform crossover και την double crossover. Η uniform crossover επιλέγει ένα γονίδιο τυχαία από κάθε γονέα. Αυτό αφήνει την πιθανότητα όλα τα γονίδια να παραχθούν από τον ίδιο γονέα. Η double crossover χρησιμοποιεί μία μεταβλητή που για κάθε παιδί είναι τυχαίος και διαφορετικός και πολλαπλασιάζεται με την διαφορά των γονιδίων των δύο γονιών. Το αποτέλεσμα αυτό προσθέτετε στο γονίδιο του πρώτου γονέα. Η συνάρτηση makeChild συντάσσεται ως εξής:

```
void    makeChild(Data &child,Data parent1,Data parent2)
{
        child.resize(parent1.size());

        //UNIFORM CROSSOVER
        for(int i=0;i<child.size();i++)// gia oses sthles
        {
                if(rand()%2==0) child[i]=parent1[i];
                else            child[i]=parent2[i];          }

        /*
        //DOUBLE CROSSOVER
        for(int i=0;i<child.size();i++)
        {
                double d=drand48();
                d=-0.25+1.50*d;
                child[i]=parent1[i]+d*(parent1[i]-parent2[i]);
```

```
        }
        */
}
```

Αφού το πρώτο παιδί δημιουργηθεί οι δύο γονείς χρησιμοποιούνται για την δημιουργία ενός δεύτερου παιδιού με την διαφορά ότι ο δεύτερος γονέας παίρνει τη θέση του πρώτου και αντιστρόφως. Στην συνέχεια επιλέγονται δύο νέοι γονείς για την δημιουργία νέων παιδιών. Αυτή η διαδικασία συνεχίζεται μέχρι να δημιουργηθούν όλα τα νέα παιδιά. Τέλος τα παιδιά παίρνουν τη θέση των ήδη υπάρχων βαρών και περνάνε στην επόμενη γενιά. Η συνάρτηση crossoverChromosomes συντάσσεται ως εξής:

```
void    crossoverChromosomes(vector<Data> &genome,vector<double> fitness)
{
        const double pc=0.99;
        vector<Data> children;
        children.resize((int)(genome.size()*pc));
        {genome.size*0.99}
        const int tournament_size=4;
        int count_children=0;
        do
        {
                int parent[2];
                for(int i=0;i<2;i++)
                {
                        int min_pos=0;
                        double min_fitness=1e+100;
                        for(int j=0;j<tournament_size;j++)
                        {
                                int r=rand() % genome.size();
```

```
                        if(fitness[r]<min_fitness)
                        {
                                min_fitness=fitness[r];
                                min_pos=r;
                        }
                }
                parent[i]=min_pos;
        }

        makeChild(children[count_children],genome[parent[0]],genome[parent[1]]);
                count_children++;
                if(count_children==children.size()) break;

        makeChild(children[count_children],genome[parent[1]],genome[parent[0]]);
                count_children++;
        }while(count_children<children.size());
        //replace the worst chromosomes
        for(int i=0;i<count_children;i++)
        {
                genome[genome.size()-i-1]=children[i];
        }
}
```

Τελευταία διαδικασία του γενετικού αλγορίθμου είναι η μετάλλαξη η οποία πραγματοποιείτε με την συνάρτηση mutateChromosomes. Η μετάλλαξη γίνεται ως εξής:για κάθε ένα από τα βάρη εάν ένας τυχαίος αριθμός είναι μικρότερος από τον ρυθμό μετάλλαξης τότε η τιμή του συναπτικού βάρους υπολογίζεται ξανά με αντίστοιχο τρόπο όπως και στην αρχικοποίηση του. Η συνάρτηση συντάσσεται όπως παρακάτω

```
void    mutateChromosomes(vector<Data> &genome,Data l,Data r)
```

```
{
	double pm=1.0/l.size();
	for(int i=1;i<genome.size();i++)
	{
		for(int j=0;j<genome[i].size();j++)
		{
			double rt=drand48();
			if(rt<pm)
			{
				genome[i][j]=l[j]+(r[j]-l[j])*drand48();
			}
		}
	}
}
```

Όλες οι παραπάνω διαδικασίες αποτελούν την εκπαίδευση του νευρώνικού δικτύου και εκτελούνται για όλες τις γενιές.

4.3.3 Τερματισμός και αποτελέσματα

Το τελευταίο μέρος της εφαρμογής είναι να χρησιμοποιήσει τα εκπαιδευμένα βάρη στα δεδομένα ελέγχου ώστε να ελεγχθεί η αποτελεσματικότητα της εκπαίδευσης. Τα δεδομένα του πίνακα data_test θα περάσουν μέσα από την συνάρτηση calculatefitness μαζί με τα ήδη εκπαιδευμένα βάρη και ένα πίνακα καταλληλότητας για τα δεδομένα ελέγχου. Αξίζει να σημειωθεί ότι τα βάρη δεν θα αλλάξουν ξανά μετά την εκπαίδευση αφού ο γενετικός δεν εκτελείται ξανά. Η κλήση της συνάρτησης θα γίνει ως εξής

```
calculateFitness(d,genome, fitness_test,data_test ,left, right,genome_count,patterns,h);
```

Τέλος θα γίνει και ταξινόμηση για τα δεδομένα ελέγχου σύμφωνα με την καταλληλότητα τους με την συνάρτηση sortchromosomes για την καλύτερη συλλογή του αποτελέσματος. Η κλήση της γίνεται ως εξής

sortChromosomes(genome,fitness_test);

Τελευταίο κομμάτι της εφαρμογής είναι η έξοδος των αποτελεσμάτων. Εμφανίζονται στο χρήστη η καλύτερη καταλληλότητα για τα δεδομένα που ισοδυναμεί με την απόκλιση από το επιθυμητό αποτέλεσμα.

Για τον έλεγχο της λειτουργίας της εφαρμογής κρατήθηκαν στατιστικά από δοκιμές με διαφορετικές αρχικοποιήσεις των βαρών και αλλαγή των μεταβλητών του προβλήματος , δηλαδή του αριθμού των γενεών ,των αριθμό των χρωμοσωμάτων και των κόμβων επεξεργασίας του νευρωνικού.

Ακολουθεί ο πίνακας με τα αποτελέσματα για τα δεδομένα BK με διαφορετικό αριθμό χρωμοσωμάτων κάθε φορά:

Generations	Nodes	Chromosomes	Error Train	Error Test
10	3	5	8.375080	8.707662
10	3	10	8.375080	8.707662
10	3	15	8.375080	8.707662
10	3	20	8.375080	8.707662
10	3	25	8.375080	8.707662
10	3	30	3.796587	3.920515
10	3	35	6.222079	1.295269
10	3	40	7.933601	6.844477
10	3	45	8.375073	8.707659
10	3	50	2.506670	5.000709
10	3	55	2.092940	2.082140
10	3	60	2.413114	2.037613
10	3	65	3.321631	3.414839
10	3	70	5.394242	5.000709
10	3	75	1.305878	1.202806
10	3	80	1.296643	1.202806
10	3	85	1.358165	1.202805
10	3	90	1.330752	1.202806
10	3	95	1.358165	1.202806
10	3	100	1.129153	0.956407
10	3	105	1.358137	1202.806
10	3	110	1.342985	1200.373
10	3	115	1.330044	1202.806
10	3	120	1.328500	1.202806

10	3	125	1.178825	1.003191
10	3	130	1.295180	0.970581
10	3	135	1.357714	1.202573
10	3	140	1.358165	1.202806
10	3	145	1.299664	1.000759
10	3	150	1.358165	1.202806

και για το BL με διαφορετικό αριθμό χρωμοσωμάτων

Generations	Nodes	Chromosomes	Error Train	Error Test
10	3	5	3.078095	5.489339
10	3	10	16.982823	16.336173
10	3	15	3.076228	5.489339
10	3	20	5.707669	4.382600
10	3	25	3.078092	5.403655
10	3	30	1.793203	3.400166
10	3	35	3.078096	5.487941
10	3	40	2.374053	4.672839
10	3	45	2.247962	2.691458
10	3	50	2.884196	2.648593
10	3	55	1.663205	3.039286
10	3	60	2.876043	5.489339
10	3	65	1.698244	2.657644
10	3	70	1.723674	3.230393
10	3	75	2.455734	2.995238
10	3	80	1.610797	2.305803
10	3	85	1.674946	2.387104
10	3	90	1.685717	2.593568
10	3	95	1.396971	2.368745
10	3	100	1.646537	3.230393
10	3	105	1.898290	1.814940
10	3	110	1.730427	3.308669
10	3	115	2.111001	2.542708
10	3	120	3.077979	3.583487
10	3	125	1.514528	2.406109
10	3	130	1.957686	2.660852
10	3	135	1.496842	2.132629
10	3	140	1.336691	2.463911
10	3	145	1.684669	3.116122
10	3	150	1.874311	2.797392

Παρακάτω είναι τα αποτελέσματα αλλάζοντας των αριθμό των γενεών για το BK

Generations	Nodes	Chromosomes	Error Train	Error Test

10	3	20	8.375080	8.707662
20	3	20	1.948796	1.921118
30	3	20	1.399094	1.177689
40	3	20	2.001295	1.668735
50	3	20	7.658271	7.964177
60	3	20	1.399094	1.177689
70	3	20	1.647129	1.600085
80	3	20	2.091059	1.748190
90	3	20	1.949929	1.623589
100	3	20	3.837311	3.599039
110	3	20	8.375080	8.707662
120	3	20	8.375080	8.707662
130	3	20	1.237791	1.117919
140	3	20	1.399011	1.118496
150	3	20	8.375080	8.707662
160	3	20	1.309598	1.228088
170	3	20	1.495003	1.392356
180	3	20	1.940458	1.668406
190	3	20	8.352815	8.699097
200	3	20	2.274783	1.912593
210	3	20	2.274783	1.912593
220	3	20	1.371289	1.225177
230	3	20	2.347673	1.978354
240	3	20	1.362656	1.160587
250	3	20	1.286045	0.944492
260	3	20	1.319563	1.160587
270	3	20	1.346575	1.179068
280	3	20	1.236417	1.160173
290	3	20	1.102022	0.998236
300	3	20	1.253049	1.170261

και τα αποτελέσματα για το BL με διαφορετικό αριθμό γενεών

Generations	Nodes	Chromosomes	Error Train	Error Test
10	3	20	2.207740	2.626142
20	3	20	2.092483	2.626142
30	3	20	1.880380	4.697122
40	3	20	1.937034	2.626142
50	3	20	1.596289	3.001003
60	3	20	2.429077	4.902448
70	3	20	1.759482	2.478453
80	3	20	1.964681	3.568223
90	3	20	2.073863	2.626142
100	3	20	2.379185	2.730910
110	3	20	1.203526	2.510288
120	3	20	1.716314	4.065262

130	3	20	1.524019	5.379156
140	3	20	2.462802	5.489339
150	3	20	6.993988	8.890778
160	3	20	1.770066	3.346725
170	3	20	1.916439	24.588580
180	3	20	1.495809	2.785996
190	3	20	2.457640	5.489339
200	3	20	1.625139	2.899504
210	3	20	1.728892	2.599154
220	3	20	0.925341	1.814221
230	3	20	1.416333	4.457656
240	3	20	2.105563	5.489267
250	3	20	2.428651	5.489339
260	3	20	1.323961	2.684116
270	3	20	2.445375	11.692075
280	3	20	2.002540	5.036551
290	3	20	0.747075	0.690364
300	3	20	0.712966	1.343230

Τέλος τα αποτελέσματα για διαφορετικό αριθμό κόμβων επεξεργασίας για το BK

Generations	Nodes	Chromosomes	Error Train	Error Test
50	1	20	8.375080	8.508227
50	2	20	1.444082	1.206821
50	3	20	1.399000	1.177689
50	4	20	3.150393	2.715276
50	5	20	5.575296	7.415356
50	6	20	1.345556	1.168510
50	7	20	1.862999	1.547821
50	8	20	2.427506	1.620179
50	9	20	1.399034	1.177653
50	10	20	6.065283	5.471183
50	11	20	1.364886	1.214666
50	12	20	1.959059	1.932644
50	13	20	3.523829	3.630522
50	14	20	1.362040	1.160427
50	15	20	1.345567	1.168000
50	16	20	1.468851	1.359391
50	17	20	1.288807	1.156431
50	18	20	3.207356	2.768196
50	19	20	1.362760	1.160614
50	20	20	2.686912	2.300042
50	21	20	1.672074	1.605348
50	22	20	1.396025	1.176119
50	23	20	1.533980	1.167807

50	24	20	2.789832	2.381958
50	25	20	1.348562	1.160903
50	26	20	1.706383	1.658669
50	27	20	1.432062	1.311791
50	28	20	1.416353	1.290548
50	29	20	1.346635	1.163382
50	30	20	1.864567	1.831239

και για το BL

Generations	Nodes	Chromosomes	Error Train	Error Test
50	1	20	3.078096	5.488344
50	2	20	3.078096	5.489339
50	3	20	1.880390	2.626142
50	4	20	60.214561	2.665045
50	5	20	2.778112	2.918701
50	6	20	2.034704	3.876439
50	7	20	1.678239	2.628533
50	8	20	8.747278	17.671741
50	9	20	2.746968	15.083261
50	10	20	20.100085	43.575783
50	11	20	21.182619	71.849775
50	12	20	17.114681	318.791878
50	13	20	13.258644	86.872428
50	14	20	10.810093	114.173448
50	15	20	11.886123	53.297914
50	16	20	20.592589	55.930581
50	17	20	1.872065	59.227551
50	18	20	20.278427	77.383926
50	19	20	6.611260	296.328695
50	20	20	59.235299	274.839382
50	21	20	9.830385	434.559854
50	22	20	35.306974	201.991736
50	23	20	81.715608	328.922974
50	24	20	61.951419	195.384481
50	25	20	19.071633	106.406018
50	26	20	48.742513	309.599265
50	27	20	25.046432	47.442957
50	28	20	57.320842	151.236601
50	29	20	33.582630	395.916130
50	30	20	23.892750	173.428456

Παρατηρείται ότι στα πειράματα με διαφορετικό αριθμό γενεών και χρωμοσωμάτων όσο μεγαλύτερα είναι αυτά τα νούμερα τόσο

περισσότερο τείνουν προς το 1 ή και σε μικρότερο αριθμό.Φυσικά όσο μικρότερος ο αριθμός τόσο πιο εκπαιδευμένο είναι το δίκτυο.Οι διαφορές όμως σε αυτές τις περιπτώσεις δεν είναι και πολύ μεγάλες.

Στα πειράματα με διαφορετικό αριθμό κόμβων επεξεργασίας όμως παρατηρείται μεγάλη απόκλιση ιδιαίτερα στο σύνολο δεδομένων BL όπου μετά από κάποιο αριθμό κόμβων το σφάλμα είναι πολύ μεγάλο και δεν γίνεται καθόλου καλή εκπαίδευση.Αυτό συμβαίνει γιατί παρότι οι περισσότεροι κόμβοι σημαίνουν περισσότερες πράξεις και έλεγχος ταυτόχρονα σημαίνει και μικρότερο mutation rate καθώς στην εφαρμογή τα δύο αυτά μεγέθη είναι αντιστρόφως ανάλογα.Μικρότερο mutation rate σημαίνει ότι τα βάρη αλλάζουν όλο και λιγότερο με αποτέλεσμα να μένουν στάσιμα ασχέτως από αριθμό γενεών κτλ.

Για να βελτιωθεί η εφαρμογή θα μπορούσε να δοθεί περισσότερη ελευθερία στο χρήστη για πειραματισμό.Θα μπορούσε να δίνονται κάποιες επιπλέον τιμές όπως τα όρια για τα margin και εισαγωγή περισσότερων σύνολο δεδομένων για παρατήρηση.Μια καλή επιλογή θα ήταν επίσης επιπλέον πειραματισμός στο γενετικό κομμάτι, όπως διαφορετικές μέθοδοι επιλογής για τους γονείς και διασταύρωσης.

4.4 Παράθεση κώδικα

Στο κεφάλαιο αυτό γίνεται παράθεση του κώδικα που χρησιμοποιήθηκε για την δημιουργία της εφαρμογής. Οτιδήποτε κείμενο ακολουθεί μετά από το σύμβολο % είναι σχόλιο για την καλύτερη κατανόηση και κατηγοριοποίηση του προγράμματος.

```
#include <iostream>

#include <fstream>

#include <string>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <vector>
```

```
#include <cstdlib>
using namespace std;
typedef vector<double> Data;

double makeChromosome(Data &x, Data l, Data r)//arxikopoihsh genome
{
   srand(30);
   for (int i = 0; i < x.size(); i++)
      x[i] = l[i]+(r[i] - l[i]) * drand48();
}

double sig(double x) {
   return (1 / (1 + pow(M_E, -x)));
}

int isPointIn(Data x, Data l, Data r) //idia me to doublegen
{
   for (int i = 0; i < x.size(); i++)
      if (x[i] < l[i] || x[i] > r[i])
         return 0;
   return 1;
}

double problemFitness(Data x, Data l, Data r, double desired_output, double
actual_output) {
   if (!isPointIn(x, l, r)) return 1e+100;
   return pow(desired_output - actual_output, 2); // (N(x,w)-y)^2
```

```
}


Data calculateFitness(int d,vector<Data> genome, vector<double>
&fitness,vector<Data> data, Data l, Data r, int genome_count,int patterns,int h)

{
   for (int i = 0; i < genome_count; i++) {
      fitness[i] = 0;
      for (int j = 0; j < patterns; j++)
      {
         double sum = 0; //N(x,w)
         for (int k = 0; k < h; k++)
         {
            double first = genome[i][(d + 2) * k - (d + 1) - 1];
            double second = 0;
            for (int m = 0; m < d; m++) {
               second = second + genome[i][(d + 2) * k - (d + 1) + m - 1] * data[j][m-1];
            }
            second = second + genome[i][(d + 2) * k - 1];
            double sigmoid = sig(second);
            sum = sum + first*sigmoid; //N(x,w)
         }
         double actual_output = sum;
         double desired_output = data[j][d];
         fitness[i] = fitness[i] + problemFitness(genome[i], l, r, desired_output,
actual_output);
      }
   }
   return fitness;
```

```
}


void    sortChromosomes(vector<Data> &genome,vector<double> &fitness)
{
        Data temp;
        temp.resize(genome[0].size());
        for(int i=0;i<genome.size();i++)
        {
                for(int j=0;j<genome.size()-1;j++)
                {
                        if(fitness[j+1]<fitness[j])
                        {
                                double dtemp=fitness[j];
                                fitness[j]=fitness[j+1];
                                fitness[j+1]=dtemp;
                                temp=genome[j];
                                genome[j]=genome[j+1];
                                genome[j+1]=temp;
                        }
                }
        }
}

void    makeChild(Data &child,Data parent1,Data parent2)
{
        child.resize(parent1.size());
```

```
	//UNIFORM CROSSOVER
	for(int i=0;i<child.size();i++)
	{
		if(rand()%2==0) child[i]=parent1[i];
		else		child[i]=parent2[i];
	}

	/*
	//DOUBLE CROSSOVER
	for(int i=0;i<child.size();i++)
	{
		double d=drand48();
		d=-0.25+1.50*d;
		child[i]=parent1[i]+d*(parent1[i]-parent2[i]);
	}
	*/
}

void	crossoverChromosomes(vector<Data> &genome,vector<double> fitness)
{
	const double pc=0.99;
	vector<Data> children;
	children.resize((int)(genome.size()*pc));
	const int tournament_size=4;
	int count_children=0;
	do
```

```
{
        int parent[2];
        for(int i=0;i<2;i++)
        {
                int min_pos=0;
                double min_fitness=1e+100;
                for(int j=0;j<tournament_size;j++)
                {
                        int r=rand() % genome.size();
                        if(fitness[r]<min_fitness)
                        {
                                min_fitness=fitness[r];
                                min_pos=r;
                        }
                }
                parent[i]=min_pos;
        }

makeChild(children[count_children],genome[parent[0]],genome[parent[1]]);
        count_children++;
        if(count_children==children.size()) break;

makeChild(children[count_children],genome[parent[1]],genome[parent[0]]);
        count_children++;
}while(count_children<children.size());
//replace the worst chromosomes
for(int i=0;i<count_children;i++)
{
```

```
                genome[genome.size()-i-1]=children[i];
        }
}

void    mutateChromosomes(vector<Data> &genome,Data l,Data r)
{
        double pm=1.0/l.size();
        for(int i=1;i<genome.size();i++)
        {
                for(int j=0;j<genome[i].size();j++)
                {
                        double rt=drand48();
                        if(rt<pm)
                        {
                                genome[i][j]=l[j]+(r[j]-l[j])*drand48();
                        }
                }
        }
}

int main ()
{
  vector<Data>data_train;
  vector<Data>data_test;
  int num;
  int d;
  int patterns;
```

```
int problem_dimension;
int genome_count;
int maxgenerations;
int i, j;
int h;
vector<Data> genome;

Data left;
Data right;
vector<double> fitness;
vector<double> fitness_test;



cout<<"TO RUN BK GIVE 1 OR TO RUN BL GIVE 2 ";
cin>>num;

if (num==1)
{
    ifstream myfile ("BK.TRAIN" , ios::binary);
    if (myfile.is_open())
    {
        myfile >> d;
        myfile >> patterns;
        data_train.resize(patterns);
        for (int i=0;i<patterns;i++)
        {
            data_train[i].resize(d+1);
```

```
        }
        for(int i=0; i<patterns; i++)
        {
                for (int j=0;j<=d;j++)
                {
                        myfile >> data_train[i][j];
                }
        }
        myfile.close();
}
else cout << "Unable to open file";

ifstream myfile2 ("BK.TEST" , ios::binary);
if (myfile2.is_open())
{
        myfile2 >> d;
        myfile2 >> patterns;
        data_test.resize(patterns);
        for (int i=0;i<patterns;i++)
        {
                data_test[i].resize(d+1);
        }
        for(int i=0; i<patterns; i++)
        {
                for (int j=0;j<=d;j++)
                {
                        myfile2 >> data_test[i][j];
```

```
                }
            }
            myfile2.close();
        }
else cout << "Unable to open file";
}
else if (num==2)
{
    ifstream myfile ("BL.TRAIN" , ios::binary);
      if (myfile.is_open())
      {
            myfile >> d;
            myfile >> patterns;
            data_train.resize(patterns);
            for (int i=0;i<patterns;i++)
            {
                  data_train[i].resize(d+1);
            }
            for(int i=0; i<patterns; i++)
            {
                  for (int j=0;j<=d;j++)
                  {
                        myfile >> data_train[i][j];
                  }
            }
      myfile.close();
      }
```

```
else cout << "Unable to open file";

 ifstream myfile2 ("BL.TEST" , ios::binary);
 if (myfile2.is_open())
 {
      myfile2 >> d;
      myfile2 >> patterns;
      data_test.resize(patterns);
      for (int i=0;i<patterns;i++)
      {
            data_test[i].resize(d+1);
      }
      for(int i=0; i<patterns; i++)
      {
            for (int j=0;j<=d;j++)
            {
                  myfile2 >> data_test[i][j];
            }
      }
  myfile2.close();
  }

 else cout << "Unable to open file";
 }

 //Display the arrays
 cout<<"DATASET TRAIN"<<endl;
```

```
        for(int i=0; i<patterns; i++)
        {
                for (int j=0;j<=d;j++)
                {
                cout <<  data_train[i][j]<<"  ";
           }
           cout <<endl;
        }
cout<<"DATASET TEST"<<endl;
 for(int i=0; i<patterns; i++)
 {
     for (int j=0;j<=d;j++)
        {
           cout <<  data_test[i][j]<<"  ";
     }
     cout <<endl;
 }



do {
     printf("NUMBER OF NODES? "); //arithmos kombwn problimatos
     scanf("%d", &h);
  }    while (h <= 0);
problem_dimension = (d+2)*h;

 //orismos twn margin
  left.resize(problem_dimension);
```

```
    right.resize(problem_dimension);
    for (i = 0; i < problem_dimension; i++)
    {
       left[i]=-10;
       right[i]=10;
       /*do {
          printf("ENTER LEFT MARGIN %d ", i);
          scanf("%lf", &left[i]);
          printf("ENTER RIGHT MARGIN %d ", i);
          scanf("%lf", &right[i]);
       }        while (right[i] < left[i]); //aristero oriou megalutero tou deksiou */
    }

    do {
       printf("NUMBER OF CHROMOSOMES? "); //arithmos chromosomaton
problimatos
       scanf("%d", &genome_count);
    }    while (genome_count <= 0);
    genome.resize(genome_count);

    for (i = 0; i < genome_count; i++)
    {
       genome[i].resize(problem_dimension);
       makeChromosome(genome[i], left, right);
    }

    fitness.resize(genome_count);
```

```
    fitness_test.resize(genome_count);


    do {
        printf("MAXIMUM NUMBER OF GENERATIONS ? ");
        scanf("%d", &maxgenerations);
    } while (maxgenerations <= 0);



     for (i = 0; i < maxgenerations; i++)
    {
        calculateFitness(d,genome, fitness,data_train ,left,
right,genome_count,patterns,h);
        sortChromosomes(genome,fitness);
        crossoverChromosomes(genome,fitness);
        mutateChromosomes(genome,left,right);
    }


    printf("THE BEST FITNESS AT GENERATION %d is %lf \n",i,fitness[0]);


   calculateFitness(d,genome, fitness_test,data_test ,left,
right,genome_count,patterns,h);
   sortChromosomes(genome,fitness_test);


   printf("FOR TESTING THE BEST FITNESS AT GENERATION %d is %lf
\n",i,fitness_test[0]);


   return 0;
}
```

Βιβλιογραφία

[1] Ιωάννης Τσούλος, Γενετικοί Αλγόριθμοι

http://users.cs.uoi.gr/~itsoulos/genetic.php

[2] Σπυρίδων Λυκοθανάσης (2001), Γενετικοί αλγόριθμοι και εφαρμογές

http://www.icsd.aegean.gr/lecturers/kavallieratou/NN&EP_files/pli31c.pdf

[3] Δημήτριος Α. Τσορτανίδης (2005) ,Αρχιτεκτονική και εκπαίδευση Νευρωνικών Δικτύων με γενετικούς αλγορίθμων στην πρόγνωση οικονομικών δεδομένων

http://nemertes.lis.upatras.gr/jspui/bitstream/10889/4489/1/Nimertis_Tsortanidis%28ma%29.pdf

[4] Βικιπαιδεία , Νευρωνικό Δίκτυο

http://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CF%89%CE%BD%CE%B9%CE%BA%CF%8C_%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

[5] Wikipedia ,Matlab

http://en.wikipedia.org/wiki/MATLAB

[6] Obtiko ,Genetic Algorithms

http://www.obitko.com/tutorials/genetic-algorithms/

[7] Raul Rojas , Neural Networks

http://page.mi.fu-berlin.de/rojas/neural/neuron.pdf

[8] Ioannis Tsoylos ,Dimitris Gavrilis ,Eyripidis Glavas (2008),Neural Network Construction and training using grammatical evolution

Πηγές Εικόνων

Εικόνα 1:Παράδειγμα απεικόνισης σε Matlab

http://i.stack.imgur.com/It9uz.png

Εικόνα 2:Απεικόνιση αρχιτεκτονικής σε Matlab

http://c.mql4.com/articles/2007/06/Sheme.GIF

Εικόνα 3:Γράφημα επίδοσης του δικτύου σε Matlab

http://www.humusoft.cz/produkty/matlab/aknihovny/neuralnet/nn_fittingtool_wl_16802.gif

Εικόνα 11:Ο βιολογικός τεχνητός νευρώνας

http://web.simmons.edu/~chen/nit/NIT%2791/9102801.jpg

Εικόνα 12:Το μοντέλο percerptron με ένα επίπεδο

http://www.codeproject.com/KB/recipes/NeuralNetwork_1/NN2.png

Εικόνα 13:Δίκτυο εμπρός τροφοδότησης ενός επιπέδου

http://ljs.academicdirect.org/A15/053_070_files/image015.gif

Εικόνα 14:Παράδειγμα εμπρός τροφοδότησης πολλαπλών επιπέδων

http://jsalatas.ictpro.gr/wp-content/uploads/2011/09/3_3.png

Εικόνα 15:Παράδειγμα αναδρομικού δικτύου

http://img0.tuicool.com/qMFBvq.png

Εικόνα 16:Παράδειγμα δικτύου με δομή δικτυωτού

http://rslab.movsom.com/paper/somrs/html/figure/fully_connected_feedforward_lattice_network.png

Εικόνα 17:Μέθοδος επιλογής ρουλέτας

http://www.edc.ncl.ac.uk/assets/hilite_graphics/rhjan07g02.png

Εικόνα 18:Παράδειγμα μεθόδου tournament

http://www.morphids.com/evoalgweb.png

Εικόνα 19:Παράδειγμα διασταύρωσης ενός σημείου

http://soihub.org/wiki/images/1/12/Crossover-transparent.gif

Εικόνα 20:Παράδειγμα διασταύρωσης δύο σημείων

http://wiki.eigenvector.com/images/3/39/Ga_crossover.png

Εικόνα 21:Παράδειγμα ομοιόμορφης διασταύρωσης

http://upload.wikimedia.org/wikipedia/commons/8/8f/UniformCrossover.png

Εικόνα 22:Παράδειγμα μετάλλαξης

http://www.emeraldinsight.com/content_images/fig/1540010203004.png