Ατομική Διπλωματική Εργασία

# Πρόβλεψη ποδοσφαιρικών αποτελεσμάτων με την χρήση τεχνητών νευρωνικών δικτύων

**Αντρέας Ιακώβου**

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**Μάιος 2010**

# ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

## ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ


# Πρόβλεψη ποδοσφαιρικών αποτελεσμάτων με την χρήση τεχνητών νευρωνικών δικτύων


**Αντρέας Ιακώβου**

Επιβλέπων Καθηγητής
Δρ. Χρίστος Χριστοδούλου



Η Ατομική αυτή Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2010

## Ευχαριστίες

Θα ήθελα να ευχαριστήσω περισσότερο από όλους τον επιβλέποντα καθηγητή μου Δρ. Χρίστο Χριστοδούλου που μου εμπιστεύτηκε την ανάθεση της διπλωματικής αυτής εργασίας καθώς επίσης και για τη στήριξη και καθοδήγηση που μου προσέφερε καθ' όλη τη διάρκεια της υλοποίησης της.

Ευχαριστίες επίσης θα ήθελα να δώσω στο κ. Κωνσταντίνο Παττίχη ο οποίος αποδέχτηκε να παραχωρήσει μέρος από τον χρόνο του για να αποτελέσει τον δεύτερο εξεταστή της εργασίας μου.

Ακόμη, θα ήθελα να ευχαριστήσω τους 4 προηγούμενους ερευνητές που ασχολήθηκαν με το συγκεκριμένο θέμα και έφεραν την έρευνα στο σημείο που βρισκόταν όταν άρχισα την δική μου εργασία.

Τέλος, ευχαριστώ θερμά την οικογένεια μου αλλά και τους φίλους μου για την συμπαράσταση και την υποστήριξη που μου παρείχαν το συγκεκριμένο αυτό διάστημα.

# Περίληψη

Σκοπός αυτής της εργασίας είναι η υλοποίηση τεχνητών νευρωνικών δικτύων τα οποία, χρησιμοποιώντας μεθόδους που δεν έχουν δοκιμαστεί στο παρελθόν στο συγκεκριμένο πρόβλημα, να προβλέπουν σημεία στο ποδοσφαιρικό στοίχημα με ποσοστά επιτυχίας μεγαλύτερα από αυτά που έχουν επιτευχθεί μέχρι σήμερα επιτυγχάνοντας έτσι μεγαλύτερο στοιχηματικό κέρδος. Με άλλα λόγια, σκοπός μας ήταν η δημιουργία ενός προγράμματος το οποίο θα έδινε προβλέψεις σημείων στο στοίχημα, οι οποίες θα απέφεραν στον παίκτη αύξηση του αρχικού κεφαλαίου που στοιχημάτισε και κατ' επέκταση στοιχηματικό κέρδος.

Πριν αρχίσω λοιπόν την δική μου έρευνα, μελέτησα την δουλειά τεσσάρων ατόμων οι οποίοι ασχολήθηκαν με το συγκεκριμένο πρόβλημα στο παρελθόν με τον καθένα από αυτούς να "κτίζει" στην έρευνα των προηγουμένων και να συνεισφέρει τα δικά του συμπεράσματα-ανακαλύψεις, κάτι που έφερε την γενική έρευνα σε ένα αρκετά καλό σημείο. Πολύ περιληπτικά σε αυτό το σημείο θα αναφέρω τις πιο σημαντικές αποφάσεις που προέκυψαν από την μελέτη αυτή και ήταν το είδος του στοιχήματος με το οποίο θα ασχοληθώ, το πρωτάθλημα στο οποίο θα εκπαιδεύσω τα δίκτυα μου και φυσικά ποιό είδος νευρωνικών δικτύων θα χρησιμοποιήσω. Όσον αφορά το είδος του στοιχήματος στο οποίο βασίστηκε η έρευνα ήταν ο συνολικός αριθμός τερμάτων που σημειώνονται ανά αγώνα και κατά πόσο αυτός ξεπερνάει ή όχι τα 2,5 (More – Less). Η εκπαίδευση και ο έλεγχος του δικτύου μου έγιναν στο πρωτάθλημα της Championship, της δεύτερης κατά σειρά ποδοσφαιρικής κατηγορίας στην Αγγλία. Τέλος μετά από εισήγηση του επιβλέποντα καθηγητή μου στην έρευνα αποφασίσαμε όπως χρησιμοποιήσω τα Radial Basis Function Networks (RBF Networks) για την επίλυση του προβλήματος μου αφού δεν είχαν χρησιμοποιηθεί στο παρελθόν για το συγκεκριμένο πρόβλημα. Για την υλοποίηση των RBF Networks και την δημιουργία των αρχείων με τα δεδομένα τα οποία χρησιμοποίησαν τα δίκτυα, χρησιμοποίησα την γλώσσα προγραμματισμού Java.

Επίσης, χρησιμοποίησα ένα simulator ο οποίος χρησιμοποιεί Support Vector Machine για την επίλυση του ίδιου προβλήματος. Τα Support Vector Machines θεωρητικά έχουν μεγαλύτερες δυνατότητες από τις υπόλοιπες μεθόδους που αφορούν ή στηρίζονται στα νευρωνικά δίκτυα. Σκοπός μου σε αυτό το μέρος της εργασίας ήταν η εξακρίβωση αυτής της υπόθεσης στο δικό μας πρόβλημα, αν δηλαδή η χρησιμοποίηση των SVM's

μπορεί να προσφέρει στην πρόβλεψη σημείων στο ποδοσφαιρικό στοίχημα με ποσοστά επιτυχίας μεγαλύτερα από αυτά που έχουν επιτευχθεί με τις υπόλοιπες μεθόδους που έχουν υλοποιηθεί.

Συγκρίνοντας λοιπόν τα αποτελέσματα του προγράμματος μου με αυτά των προαναφερθέντων καθώς και αυτά που θα προκύψουν με την χρήση του SVM simulator θα εξαχθούν συμπεράσματα για τις δυνατότητες και τα περιθώρια βελτίωσης που υπάρχουν στη συγκεκριμένη έρευνα.

Η φύση των Νευρωνικών Δικτύων είναι τέτοια ώστε εξυπακούεται η ύπαρξη τεράστιου αριθμού διαφορετικών αποτελεσμάτων και δεδομένων εισόδου που είτε δόθηκαν στο δίκτυο είτε προέκυψαν από την λειτουργιά του. Έτσι λοιπόν και στη δική μας περίπτωση υπήρξαν πάρα πολλά αποτελέσματα τόσο κατά την δημιουργία του δικτύου όσο και μετά το πέρας της, κατά την φάση του ελέγχου και της σύγκρισης των αποτελεσμάτων των δυο μεθόδων μεταξύ τους, καθώς και με αυτά των προηγούμενων ερευνητών. Κάποια από τα αποτελέσματα αυτά θα μας απασχολήσουν με σκοπό την κατανόηση τους και την ανάλυση της προοπτικής βελτίωσης τους σε μεταγενέστερες έρευνες – εργασίες ενώ κάποια άλλα που κρίθηκαν φτωχά δεν θα αναφερθούν καθόλου.

Και πάλι πολύ περιληπτικά θα αναφέρω ότι τα ποσοστά επιτυχίας του δικτύου RBF έφτασαν στο 68,33% για τα δεδομένα ελέγχου σε μια ολόκληρη ποδοσφαιρική περίοδο ενώ για τα δεδομένα εκπαίδευσης το ποσοστό επιτυχίας έφτασε το 73,33% και πάλι για μια ολόκληρη ποδοσφαιρική περίοδο. Τα ποσοστά αυτά επιτεύχθηκαν με την χρήση RBF δικτύου με 45 κρυφούς νευρώνες και διάνυσμα εισόδου 46 θέσεων (23 για την κάθε ομάδα). Όσον αφορά τα αποτελέσματα που προέκυψαν με την χρήση του Support Vector Machine simulator, το αντίστοιχο ποσοστό επιτυχίας ανήλθε στο 70,29% για τα δεδομένα ελέγχου.

# Περιεχόμενα

**Κεφάλαιο 1**……. **1**

**Εισαγωγή**……. **1**

1.1 Στόχος της έρευνας…….……. 1

1.2 Προηγούμενη δουλειά – Σχετική έρευνα……..... 2

**Κεφάλαιο 2**…..……..…….. **5**

**Νευρωνικά Δίκτυα** ……..……... **5**

2.1 Νευρωνικά Δίκτυα – Ιστορική αναδρομή……..……..…... 5

2.2 Τεχνητά Νευρωνικά Δίκτυα…….……. 7

2.2.1 Νευρωνικά Δίκτυα και ανθρώπινος εγκέφαλος……..... 7

2.2.2 Τρόπος λειτουργίας Τεχνητών Νευρωνικών Δικτύων……... 8

**Κεφάλαιο 3**…….…… **11**

**Radial Basis Function Networks** ……..… **11**

3.1 Εισαγωγή……...……..... 11

3.2 Τρόπος λειτουργίας……...……..... 13

3.3 Εκπαίδευση με σταθερά κέντρα……...……. 14

3.4 Εκπαίδευση με μεταβλητά κέντρα……...……. 17

**Κεφάλαιο 4**…….…… **20**

**Support Vector Machines (Μηχανές διανυσμάτων υποστήριξης)**…… **20**

4.1 Εισαγωγή στις Μηχανές διανυσμάτων υποστήριξης…… 20

4.2 Τρόπος λειτουργίας Μηχανών διανυσμάτων υποστήριξης……..……... 22

4.2.1 Μέθοδος Κυρτών πολυγώνων……... 23

4.2.2 Μέθοδος μεγίστου διαστήματος μεταξύ παράλληλων επιπέδων 24

4.2.3 Μη γραμμικά διαχωρίσιμα πρότυπα…..……...……... 25

4.2.4 Τα Support Vector Machines στην έρευνα μου……..... 26

**Κεφάλαιο 5**..... **27**

**Δεδομένα Εισόδου**..... **27**

5.1 Δεδομένα εισόδου που χρησιμοποιεί το δίκτυο..... 27

5.2 Το Πρωτάθλημα της Championship..... 33

5.3 Επεξεργασία Δεδομένων..... 35

5.4 Κανονικοποίηση Τιμών..... 38

**Κεφάλαιο 6**..... **39**

**Σχεδιασμός και υλοποίηση**..... **39**

6.1 Σχεδιασμός και υλοποίηση RBF Network..... 39

6.1.1 Κλάση FileEditor.java..... 40

6.1.2 Κλάση Match.java..... 41

6.1.3 Κλάση Season.java..... 41

6.1.4 Κλάση Team.java ..... 43

6.1.5 Κλάση HidenLayerNode.java..... 45

6.1.6 Κλάση OutputNode.java..... 47

6.1.7 Κλάση TrainingProcedure.java..... 48

6.1.8 Κλάση QuickLauncher.java..... 49

6.1.9 Δομή Δικτύου..... 50

6.2 Υλοποίηση με τον SVM Classifier..... 51

**Κεφάλαιο 7**..... **59**

**Αποτελέσματα και συζήτηση**..... **59**

7.1 Αποτελέσματα RBF Network..... 59

7.2 Αποτελέσματα SVM Simulator..... 84

**Κεφάλαιο 8**..... **93**

**Συμπεράσματα και μελλοντική εργασία**..... **93**

8.1 Γενικά συμπεράσματα..... 93

8.2 Μελλοντική εργασία..... 99

**Βιβλιογραφία – Αναφορές.**…… **100**

**Παράρτημα Α**.……..…….…… **Α-1**

Υλοποίηση RBF Δικτύου…...……...…….. Α-1

**Παράρτημα Β**.……..…….…… **Β-1**

Παράδειγμα διαδικασίας εκπαίδευσης…...……...…..…….. Β-1

# Κεφάλαιο 1

## Εισαγωγή

1.1 Στόχος της έρευνας……..…….…… 1
1.2 Προηγούμενη δουλειά – Σχετική έρευνα……..... 4

### 1.1 Στόχος της έρευνας

Σκοπός αυτής της εργασίας ήταν η υλοποίηση τεχνητών νευρωνικών δικτύων τα οποία, χρησιμοποιώντας μεθόδους που δεν έχουν δοκιμαστεί στο παρελθόν στο συγκεκριμένο πρόβλημα, να προβλέπουν σημεία στο ποδοσφαιρικό στοίχημα με ποσοστά επιτυχίας μεγαλύτερα από αυτά που έχουν επιτευχθεί μέχρι σήμερα επιτυγχάνοντας έτσι μεγαλύτερο στοιχηματικό κέρδος. Ο στόχος λοιπόν ήταν να αποδείξουμε στην επιστημονική κοινότητα ότι με την χρήση των νευρωνικών δικτύων υπάρχει η δυνατότητα για αύξηση του αρχικού κεφαλαίου των παικτών και κατ' επέκταση δημιουργία στοιχηματικού κέρδους, σε ποσοστά καλύτερα από αυτά που είχαν επιτευχθεί μέχρι σήμερα. Αναλυτικά για τα ποσοστά επιτυχίας και το στοιχηματικό κέρδος που απέφεραν θα μιλήσουμε σε μεταγενέστερο στάδιο της εργασίας αυτής. Σε αυτό το σημείο θα αναφέρουμε μόνο τον τρόπο που υπολογίζονται τα ποσοστά αυτά. Όταν αναφερόμαστε λοιπόν σε ποσοστό επιτυχίας εννοούμε το ποσοστό των σωστών προβλέψεων του δικτύου στο σύνολο όλων των αγώνων που έχει προβλέψει, ενώ όταν αναφερόμαστε σε στοιχηματικό κέρδος εννοούμε το ποσοστό κέρδους που απέφεραν οι "κινήσεις" τις οποίες πρότεινε το δίκτυο με βάση το αρχικό κεφάλαιο το οποίο επενδύσαμε. Κάθε στοίχημα αποτελείται από μία και μόνο πρόβλεψη και ο συνολικός αριθμός στοιχημάτων – προβλέψεων για το πρωτάθλημα της Championship ανέρχεται στις 480 για κάθε ποδοσφαιρική περίοδο. Περισσότερα για τα αποτελέσματα που πήραμε και για το μέτρο σύγκρισης που υπήρχε με βάση προηγούμενες προσπάθειες στο συγκεκριμένο πρόβλημα θα δούμε σε επόμενες ενότητες της εργασίας.

## 1.2 Προηγούμενη δουλειά – Σχετική έρευνα

Πριν αρχίσω λοιπόν την δική μου έρευνα, μελέτησα την δουλειά τεσσάρων ατόμων οι οποίοι ασχολήθηκαν με το συγκεκριμένο πρόβλημα στο παρελθόν με τον καθένα από αυτούς να "κτίζει" στην έρευνα των προηγουμένων και να συνεισφέρει τα δικά του συμπεράσματα-ανακαλύψεις, κάτι που έφερε την γενική έρευνα σε ένα αρκετά καλό σημείο. Σε όλες τις περιπτώσεις οι ερευνητές λειτουργούσαν στα πλαίσια της ατομικής διπλωματικής τους εργασίας και η ευκαιρία να μελετήσω τις έρευνες αυτές μου δόθηκε λόγω του επιβλέποντα καθηγητή μου Δρ. Χρίστου Χριστοδούλου ο οποίος ήταν επίσης υπεύθυνος για την επίβλεψη των εργασιών αυτών σε προηγούμενες χρονιές. Γνωρίζοντας λοιπόν τι είχε προηγηθεί στο παρελθόν, ποια λάθη έγιναν και ποια συμπεράσματα εξάχθηκαν αποτελούσε για μένα τεράστιο πλεονέκτημα αφού μπόρεσα να χρησιμοποιήσω ή να αγνοήσω κάποια στοιχεία των ερευνών αυτών χωρίς να πρέπει ο ίδιος να ασχοληθώ μαζί τους, κάτι που μου επέτρεψε να έχω πολύ περισσότερο χρόνο να ασχοληθώ με την βελτίωση του δικού μου δικτύου για την εξαγωγή όσο το δυνατό καλύτερων αποτελεσμάτων. Πιο κάτω, θα προσπαθήσω να δώσω την ουσία των αποτελεσμάτων – συμπερασμάτων της κάθε μιας εκ των τεσσάρων ερευνών ξεχωριστά όπως την αντιλήφτηκα μέσα από την μελέτη μου πριν αρχίσω να πραγματοποιώ την δική μου έρευνα.

Πρώτος που ασχολήθηκε με το πρόβλημα της πρόβλεψης ποδοσφαιρικών αποτελεσμάτων μέσω τεχνητών νευρωνικών δικτύων ήταν ο Matt Jackson το 2001 [1] στα πλαίσια της δικής του διπλωματικής εργασίας στο πανεπιστήμιο του Birkbeck. Ο συγκεκριμένος φοιτητής προσπάθησε να εκπαιδεύσει δίκτυα τα οποία να προβλέπουν την έκβαση του αγώνα όσον αφορά το νικητή (προέβλεπε δηλαδή νίκη του γηπεδούχου, του φιλοξενουμένου ή ισοπαλία). Για την εκπαίδευση των δικτύων του χρησιμοποίησε τις μεθόδους Back Propagation και Kohonen, με τα δίκτυα του να εκπαιδεύονται στο πρωτάθλημα της Premiership. Τα αποτελέσματα του κρίνονται "φτωχά" και με τις 2 μεθόδους.

Στο Back Propagation είχε ποσοστό επιτυχίας 47% αποτέλεσμα που δεν μπορεί να θεωρηθεί καλό αφού κάτι τέτοιο μπορεί να επιτευχθεί απλά επιλέγοντας το σημείο 1 σε όλες τις επιλογές μας (αφού είναι το σημείο που εμφανίζεται πιο συχνά).

Στο Kohonen είχε επιτυχία 57% αποτέλεσμα που δεν μπορεί επίσης να θεωρηθεί επιτυχία αν λάβουμε υπόψη μας ότι τα περισσότερα από τα επιτυχημένα αποτελέσματα

ήταν αυτά που θεωρούνταν πολύ πιθανά και η απόδοση που προσφερόταν από τα πρακτορεία στοιχημάτων γι' αυτά ήταν πολύ χαμηλή. Αυτό είχε ως αποτέλεσμα το κέρδος σε αυτό το 57% των επιτυχημένων στοιχημάτων να είναι χαμηλότερο από το συνολικό ποσό στοιχηματισμού.

Ο επόμενος που ασχολήθηκε με το συγκεκριμένο πρόβλημα ήταν ο Kevin Davies το 2004 [2] και πάλι στο Πανεπιστήμιο του Birkbeck. Ο Davies προσπάθησε και αυτός να προβλέψει την τελική έκβαση των παιχνιδιών όσον αφορά τον νικητή, καθώς επίσης προσπάθησε να βρει στοιχήματα με "καλή απόδοση", δηλαδή αγώνες στους οποίους η πιθανότητα που έδινε το δίκτυο για κάποιο σημείο σε σχέση με την απόδοση που έδιναν τα πρακτορεία στοιχημάτων ήταν συμφέρουσα για τον παίκτη. Χρησιμοποίησε και αυτός τις μεθόδους Back Propagation και Kohonen και δούλεψε πολύ με τα data sets χρησιμοποιώντας summarised data αλλά και expanded result data δίνοντας έτσι στο δίκτυο την δυνατότητα να εκμεταλλευτεί την γνώση του πότε έγιναν οι συγκεκριμένοι αγώνες. Η έρευνα του Davies έγινε επίσης στην πρώτη εν τη τάξη κατηγορία της Αγγλίας, την Premiership. Οι προσπάθειες του βελτίωσαν κάπως τα ποσοστά επιτυχίας αλλά και πάλι το αποτέλεσμα δεν θεωρήθηκε ικανοποιητικό με βάση τον στόχο που είχε τεθεί και αφορούσε την παραγωγή κέρδους στο αρχικό κεφάλαιο στοιχηματισμού σε συγκεκριμένα ποσοστά.

Την επόμενη χρονιά (2005) η Sian Turner [3] αφού κατάλαβε ότι στην πρόβλεψη του νικητή δεν υπήρχαν μεγάλα περιθώρια βελτίωσης στα ποσοστά επιτυχίας, αποφάσισε να ασχοληθεί με την πρόβλεψη του αριθμού των τερμάτων που θα σημειώνονταν σε κάποιο παιχνίδι και συγκεκριμένα αν ο αριθμός αυτός θα ήταν μικρότερος ή μεγαλύτερος του 2,5 (MORE – LESS). Η Turner χρησιμοποίησε και αυτή Back Propagation με διάφορες μάλιστα συναρτήσεις κατωφλίου, ενώ ασχολήθηκε πολύ με την τροποποίηση των input sets πετυχαίνοντας ποσοστά επιτυχίας που έφταναν το 56,83% ένα ποσοστό που όντως επέφερε στοιχηματικό κέρδος. Η έρευνα της έγινε επίσης με δεδομένα από την Premiership.

Τέλος, ο Θεοφάνης Νεοκλέους το 2007 [4] στο Πανεπιστήμιο Κύπρου "έκτισε" στην έρευνα κυρίως της Sian Turner, αφού ασχολήθηκε και αυτός με την πρόβλεψη του συνολικού αριθμού τερμάτων που σημειώνονται σε κάθε αγώνα, πετυχαίνοντας με τις αλλαγές που δοκίμασε τόσο στη δομή του δικτύου όσο και στα data sets να αυξήσει τα ποσοστά επιτυχίας και κατ' επέκταση το στοιχηματικό κέρδος. Ο Νεοκλέους [4] χρησιμοποίησε και αυτός κυρίως Back Propagation ενώ η εκπαίδευση των δικτύων του

έγινε στο πρωτάθλημα της Championship. Μεγάλο ρόλο στην βελτίωση των αποτελεσμάτων του έπαιξε η κανονικοποίηση που εφάρμοσε στα διανύσματα εισόδου στο δίκτυο του. Η βελτίωση αυτή ήταν της τάξης του 5% αφού τα αποτελέσματα του έφτασαν σε ποσοστό σωστής πρόβλεψης 62% και απέφεραν επιστροφή κερδών της τάξεως του 23% του αρχικού κεφαλαίου.

Έχοντας λοιπόν σαν δεδομένα τα όσα πέτυχαν αλλά και όσα δεν κατάφεραν να πετύχουν οι 4 αυτοί φοιτητές και γνωρίζοντας πλέον καλά τα προβλήματα που αντιμετώπισαν, μπορώ να υποθέσω με αρκετή σιγουριά ότι τα περιθώρια βελτίωσης στα ποσοστά επιτυχίας χρησιμοποιώντας τις συγκεκριμένες τεχνικές έχουν στενέψει. Τα Data Sets μεταβλήθηκαν με κάθε δυνατό τρόπο ενώ το ίδιο έγινε και με την δομή του δικτύου, με τα ποσοστά επιτυχίας να φτάνουν σε κάποιο σημείο και να μην μπορούν να το ξεπεράσουν. Με αυτά τα δεδομένα και μετά από εισήγηση του επιβλέποντα καθηγητή μου στην έρευνα αποφασίσαμε όπως χρησιμοποιήσω τα Radial Basis Function Networks (RBF Networks) για την πρόβλεψη του συνολικού αριθμού τερμάτων που θα σημειωθούν σε ένα αγώνα και αν αυτός θα ξεπερνάει ή όχι το 2,5. Η επιλογή να χρησιμοποιηθούν τα RBF Networks έγινε γιατί δεν έχουν χρησιμοποιηθεί στο παρελθόν για το συγκεκριμένο πρόβλημα ενώ υπάρχει η άποψη ότι σε κάποιες περιπτώσεις μπορούν να λύσουν συγκεκριμένα προβλήματα καλύτερα από τα Multilayer Perceptrons.
Επίσης θα χρησιμοποιήσω ένα simulator ο οποίος θα χρησιμοποιεί Support Vector Machines για την επίλυση του ίδιου προβλήματος. Σκοπός μου θα είναι η εξακρίβωση της υπόθεσης ότι η χρησιμοποίηση των SVM' s μπορεί να προσφέρει στην πρόβλεψη σημείων στο ποδοσφαιρικό στοίχημα με ποσοστά επιτυχίας μεγαλύτερα από αυτά που έχουν επιτευχθεί με τις υπόλοιπες μεθόδους που έχουν υλοποιηθεί αφού γενικά υπάρχει η αντίληψη ότι τα Support Vector Machines έχουν μεγαλύτερες δυνατότητες.

# Κεφάλαιο 2

## Νευρωνικά Δίκτυα

2.1 Νευρωνικά Δίκτυα – Ιστορική αναδρομή…….……..….. 5
2.2 Τεχνητά Νευρωνικά Δίκτυα…….……. 7
2.2.1 Νευρωνικά Δίκτυα και ανθρώπινος εγκέφαλος……..... 7
2.2.2 Τρόπος λειτουργίας Τεχνητών Νευρωνικών Δικτύων……..... 8

### 2.1 Νευρωνικά Δίκτυα – Ιστορική αναδρομή

**-** Η περίοδος των νευρωνικών δικτύων άρχισε με την *πρωτοποριακή για την τότε εποχή δουλειά των McCulloch και Pitts το 1943* [8]. Ο πρώτος ήταν ψυχίατρος ενώ ο δεύτερος μαθηματικός. Η κλασσική εργασία των McCulloch και Pitts έγινε μέσα σε μια κοινωνία που ασχολούνταν με τους νευρώνες στο πανεπιστήμιο του Σικάγο για πάνω από 5 χρόνια. Αυτή η εργασία περιέγραφε το *λογικό λογισμό των νευρωνικών δικτύων*.
Από τότε μέχρι σήμερα τα νευρωνικά δίκτυα έχουν σίγουρα διανύσει πολύ δρόμο και έχουν υποστεί πολλές και σημαντικές αλλαγές μέσω ανακαλύψεων που έγιναν από μεγάλους επιστήμονες της κάθε εποχής. Πιο κάτω αναφέρονται οι πιο σημαντικές ημερομηνίες στην ιστορία των νευρωνικών δικτύων.

**-** Η επόμενη μεγάλη ανάπτυξη πάνω στα νευρωνικά δίκτυα μετά το 1943, ήρθε το 1949 με την έκδοση του βιβλίου του Hebb με τίτλο “The Organization of Behavior”, στο οποίο μια ιδιαίτερη δήλωση ενός *φυσιολογικού κανόνα μάθησης για συναπτικές τροποποιήσεις* έγινε για πρώτη φορά. Πιο συγκεκριμένα ο Hebb πρότεινε ότι *η συνδετικότητα του εγκεφάλου συνεχώς αλλάζει καθώς ο οργανισμός μαθαίνει διάφορες εργασίες, και ότι οι νευρωνικοί συγκεντρωτές δημιουργούνται από τέτοιες αλλαγές*. Επίσης πρότεινε το διάσημο *αίτημα μάθησης* σύμφωνα με το οποίο *η αποτελεσματικότητα μιας σύναψης μεταβλητής ανάμεσα σε δύο νευρώνες αυξάνεται από την επαναλαμβανόμενη ενεργοποίηση του ενός νευρώνα από τον άλλο κατά μήκος της σύναψης.*

- Το 1952 το βιβλίο του Ashby :"Design for a brain: The Origin of Adaptive Behavior" εκδόθηκε, το οποίο ασχολήθηκε με την *βασική έννοια ότι η προσαρμοζόμενη συμπεριφορά δεν είναι έμφυτη αλλά μαθαίνεται*.

- 15 χρόνια μετά την έκδοση της εργασίας των McCulloch και Pitt μια νέα προσέγγιση πάνω στο πρόβλημα της αναγνώρισης προτύπων έγινε από τον Rosenblatt (1958) στην εργασία του πάνω στο αισθητήριο (perceptron). Το ιδιαίτερο επίτευγμα του ήταν το αποκαλούμενο *θεώρημα σύγκλισης αισθητηρίου*(perceptron convergence theorem).

- Το 1969 εκδόθηκε το βιβλίο των Minsky και Papert που με μαθηματικά απέδειξε ότι υπάρχουν όρια πάνω στο τι μπορεί να υπολογιστεί από τα αισθητήρια αφού μέχρι τότε υπήρχε η αντίληψη ότι τα νευρωνικά δίκτυα μπορούσαν να κάνουν τα πάντα.

- Το 1982 ο Hopfield χρησιμοποίησε την ιδέα μια συνάρτησης ενέργειας για να φτιάξει ένα νέο τρόπο κατανόησης του υπολογισμού που γίνεται από τα δίκτυα με συμμετρικές συναπτικές συνδέσεις.
- Μια ακόμα σημαντική ανάπτυξη το 1982 ήταν η έκδοση της εργασίας του Kohonen πάνω στους χάρτες αυτοοργάνωσης, χρησιμοποιώντας *μιας ή δύο διαστάσεων δικτυωτές δομές*.

- Το 1986 η ανάπτυξη του αλγορίθμου για πίσω διάδοση (back-propagation algorithm) αναφέρθηκε από τον Rumelhart.

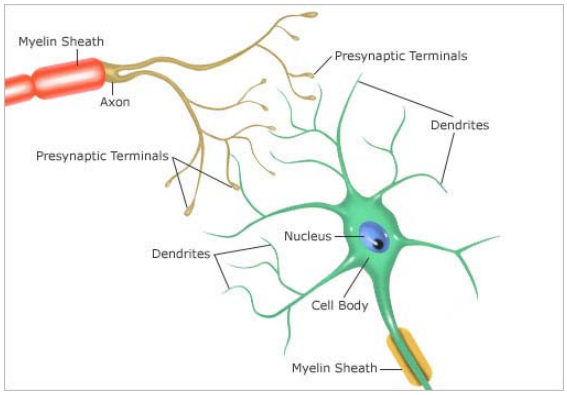
- Επίσης το 1988 οι Broomhead και Lowe περιέγραψαν μία διαδικασία για το σχεδιασμό "προς τα εμπρός τροφοδότησης"(feedforward) δικτύων χρησιμοποιώντας *συναρτήσεις ακτινικής βάσης(RBF),* που είναι μια εναλλαγή των πολυεπίπεδων αισθητηρίων.

## 2.2 Τεχνητά Νευρωνικά Δίκτυα

### 2.2.1 Νευρωνικά Δίκτυα και ανθρώπινος εγκέφαλος

Η ανάπτυξη των τεχνητών νευρωνικών δικτύων, κοινώς γνωστά ως "νευρωνικά δίκτυα", υποκινήθηκε άμεσα από την διαπίστωση ότι οι εγκεφαλικοί υπολογιστές είναι ένας εξ' ολοκλήρου διαφορετικός δρόμος από τους συμβατικούς ψηφιακούς υπολογιστές.[5] [6]

Ο εγκέφαλος είναι ένας πολύ πολύπλοκος, μη-γραμμικός και παράλληλος υπολογιστής. Έχει τη δυνατότητα να οργανώνει τους νευρώνες έτσι ώστε να εκτελεί συγκεκριμένους υπολογισμούς πολύ πιο γρήγορα από τους πιο γρήγορους ψηφιακούς υπολογιστές που υπάρχουν. Αυτό επιτυγχάνεται με μοναδικό τρόπο, αφού κατά τη γέννησή του ο εγκέφαλος έχει την ικανότητα να κατασκευάζει τους δικούς του κανόνες, κοινώς "εμπειρία", η οποία μεγαλώνει με τα χρόνια. Κατά τα 2 πρώτα χρόνια ζωής, έχουμε τη μέγιστη ανάπτυξη, όπου περίπου 1 εκατομμύριο συνάψεις δημιουργούνται ανά δευτερόλεπτο. Με τον όρο συνάψεις αναφερόμαστε στις βασικές δομικές και λειτουργικές μονάδες που μεσολαβούν στην ενδοεπικοινωνία των νευρώνων.




*Σχήμα 2.1 – Δομή πραγματικού εγκεφάλου*

Ένα νευρωνικό δίκτυο είναι ένα συμπαγής παράλληλος κατανεμημένος επεξεργαστής, που έχει τη φυσική κλίση να αποθηκεύει εμπεριστατωμένη γνώση και να την κάνει διαθέσιμη για χρήση. Είναι δηλαδή ένα μαθηματικό μοντέλο για την επεξεργασία πληροφορίας που προσεγγίζει την υπολογιστική και αναπαραστατική δυνατότητα μέσω συνάψεων. Το μοντέλο είναι εμπνευσμένο από τα βιοηλεκτρικά δίκτυα που δημιουργούνται στον εγκέφαλο ανάμεσα στους νευρώνες (νευρικά κύτταρα) και στις συνάψεις.

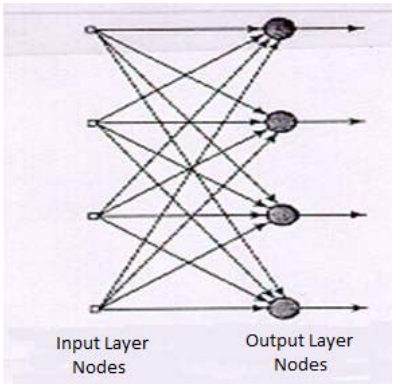
### 2.2.2 Τρόπος λειτουργίας Τεχνητών Νευρωνικών Δικτύων

Όπως αναφέρθηκε και προηγουμένως, τα τεχνητά νευρωνικά δίκτυα (ΤΝΔ) είναι μαθηματικά μοντέλα που προσπαθούν να προσεγγίσουν μια συνάρτηση υπολογίζοντας σχέσεις μεταξύ παραμέτρων [5][6]. Έχουν εμπνευστεί από βιοηλεκτρικά δίκτυα, εξού και χρησιμοποιούν έννοιες όπως εγκέφαλος, νευρώνες και συνάψεις. Τα τεχνητά νευρωνικά δίκτυα χρησιμοποιούνται συχνά για τη μελέτη της μάθησης και της μνήμης. Συνήθως αποτελούνται από έναν αριθμό απλών μονάδων επεξεργασίας που συνδέονται σε μεγάλο βαθμό σχηματίζοντας ένα δίκτυο.

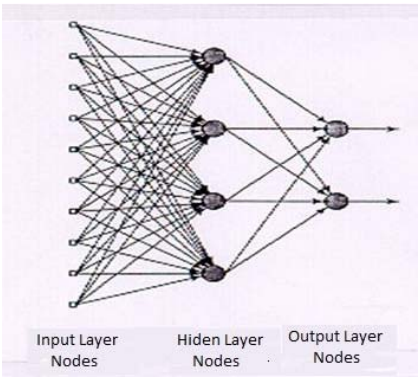
Ο νευρώνας είναι η βασική μονάδα επεξεργασίας της πληροφορίας σε κάθε νευρωνικό δίκτυο. Έχει μία ή περισσότερες εισόδους τις οποίες συνδυάζει μέσω μιας συνάρτησης μεταφοράς και το αποτέλεσμα αυτό περνάει μέσα από μια συνάρτηση ενεργοποίησης. Η τιμή που προκύπτει τελικά αποτελεί τη μοναδική έξοδο του νευρώνα. Κάθε νευρώνας αποτελείται από τα ακόλουθα βασικά στοιχεία:

- Ένα σύνολο από συνάψεις κάθε μία από τις οποίες συνδέει το νευρώνα με μία είσοδο και της αντιστοιχείται κάποιο βάρος.
- Μία συγκεκριμένη συνάρτηση μεταφοράς η οποία χρησιμοποιεί ως παραμέτρους τα βάρη των συνάψεων και εφαρμόζεται πάνω στις εισόδους των νευρώνων.
- Μία συνάρτηση ενεργοποίησης η οποία παίρνει ως είσοδο το αποτέλεσμα της συνάρτησης μεταφορά και παράγει την έξοδο του νευρώνα. Συνήθως σκοπός της συνάρτησης ενεργοποίησης είναι να περιορίζει το βαθμό ενεργοποίησης του νευρώνα.

Η πιο απλή μορφή ΤΝΔ είναι ένας σύνδεσμος που τροφοδοτείται προς τα μπροστά και φέρει στιβάδες μονάδων εισόδου και εξόδου οι οποίες αλληλοσυνδέονται μεταξύ τους. Μία συνειρμική μνήμη κωδικογραφείται τροποποιώντας την ισχύ των συνδέσεων ανάμεσα στις στιβάδες έτσι ώστε, όταν παρουσιάζεται ένα είδος εισερχόμενης πληροφορίας, να ανακαλείται ένα αποθηκευμένο μοτίβο το οποίο έχει συσχετιστεί με το συγκεκριμένο είδος πληροφορίας.




*Σχήμα 2.2 – Δίκτυο εμπρός-τροφοδότησης με ένα επίπεδο νευρώνων*

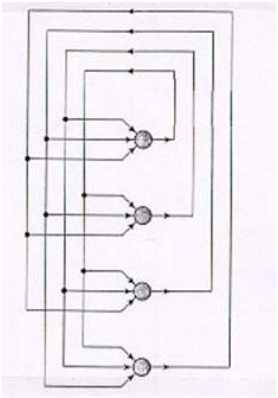



*Σχήμα 2.3 – Πλήρως συνδεδεμένο δίκτυο εμπρός-τροφοδότησης*

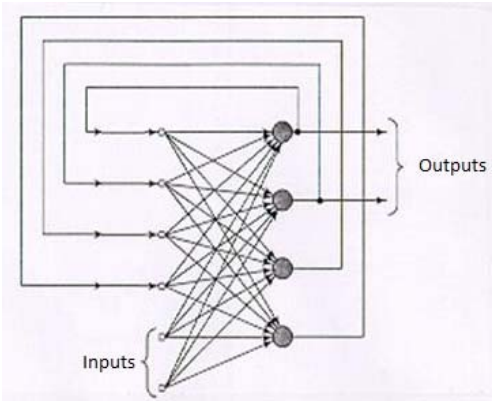
Στη πρώτη εικόνα βλέπουμε ένα Δίκτυο εμπρός-τροφοδότησης με ένα επίπεδο νευρώνων, ενώ στη δεύτερη έχουμε ένα πλήρως συνδεδεμένο δίκτυο εμπρός-τροφοδότησης με ένα κρυφό επίπεδο και ένα επίπεδο εξόδου.

Ένα πιο πολύπλοκο ΤΝΔ είναι ένα επαναλαμβανόμενο Νευρωνικό Δίκτυο. Όταν οι έξοδοι κάποιων νευρώνων, γίνονται είσοδοι σε νευρώνες προηγούμενων επιπέδων (προς το μέρος της εισόδου του δικτύου), τότε έχουμε ανάδραση. Αποτελείται από μία και μόνο στιβάδα όπου όλες οι μονάδες συνδέονται μεταξύ τους και κάθε μία λειτουργεί και ως είσοδος και ως έξοδος πληροφορίας.

Αυτός ο σχεδιασμός, επιτρέπει στο δίκτυο να αποθηκεύει μοτίβα και όχι ζευγάρια αντικειμένων. Η αποκωδικοποίηση αυτού του είδους αυτοσυσχετιζόμενου δικτύου επιτυγχάνεται με μία περιοδική αναζήτηση ενός αποθηκευμένου μοτίβου.



*Σχήμα 2.4 – Αναδρομικό Δίκτυο Χωρίς κρυφούς νευρώνες*




*Σχήμα 2.5 – Αναδρομικό Δίκτυο με ένα επίπεδο νευρώνων*

Στη πρώτη εικόνα βλέπουμε ένα Αναδρομικό Δίκτυο *χωρίς* κρυφούς νευρώνες, ενώ στη δεύτερη βλέπουμε ένα Αναδρομικό Δίκτυο με ένα κρυφό επίπεδο νευρώνων.

Η ομοιότητα των Τεχνητών Νευρωνικών Δικτύων με τον εγκέφαλο βρίσκεται στον τρόπο με τον οποίο αποθηκεύουν και επεξεργάζονται την πληροφορία [6]. Η «γνώση» που επεξεργάζονται βρίσκεται στο ίδιο το δίκτυο. Δεν έχουν ξεχωριστή τοποθεσία για τη μνήμη όπως ο ψηφιακός υπολογιστής, στον οποίο ο αριθμητικός επεξεργαστής και η μνήμη είναι χωριστά. Αντίθετα, έχουν μνήμη με διευθυνσιοδότηση προς το περιεχόμενο. Σε ένα Τεχνητό Νευρωνικό Δίκτυο, η πληροφορία αποθηκεύεται ανάλογα με το εύρος των συνδέσεων, όπως συμβαίνει και στις συνάψεις που αλλάζει η ισχύς τους κατά τη μάθηση. Τα Τεχνητά Νευρωνικά Δίκτυα δεν είναι προγραμματισμένα να εκτελούν μία δεδομένη διεργασία. Κάθε «νευρώνας» στο εσωτερικό του απαντά σύμφωνα με το άθροισμα της πληροφορίας που δέχεται. Ωστόσο, μπορούν να εκπαιδευτούν για να κάνουν έξυπνα πράγματα. Οι κανόνες της μάθησης που εκπαιδεύουν τα δίκτυα το επιτυγχάνουν αυτό τροποποιώντας την ισχύ των συνδέσεων ανάμεσα στους νευρώνες, και ένας κοινός κανόνας είναι η λήψη της πληροφορίας που μεταδίδεται από ένα δίκτυο ως απάντηση σε ένα ερέθισμα και η σύγκρισή του με το επιθυμητό ερέθισμα. Κάθε «λάθος» στη σύγκριση χρησιμοποιείται για να προσαρμοστεί το εύρος των συνδέσεων ώστε να επιτευχθεί ένα αποτέλεσμα όσο γίνεται πιο κοντά στο επιθυμητό. Το δίκτυο σταδιακά μειώνει τα λάθη στο ελάχιστο. Αυτή η διαδικασία είναι αποτελεσματική, αλλά αργή.

Το εξαιρετικό γνώρισμα των Τεχνητών Νευρωνικών Δικτύων βρίσκεται στην ικανότητά τους να γενικεύουν απαντώντας σε ερεθίσματα που δεν εκτέθηκαν ποτέ κατά την εκπαίδευσή τους. Βλέπουν σχέσεις, κάνουν συσχετίσεις και ανακαλύπτουν κανονικότητες στα μοτίβα. Ανέχονται δε τα λάθη, ακριβώς όπως ο πραγματικός εγκέφαλος. Μπορούν να ανακαλέσουν μία αποθηκευμένη πληροφορία ακόμη και όταν το εισερχόμενο ερέθισμα δεν είναι ξεκάθαρο ή πλήρες. Αυτές είναι πολύ σημαντικές ιδιότητες των βιολογικών εγκεφάλων και τα Τεχνητά Νευρωνικά Δίκτυα μπορούν επίσης να τα επιτύχουν.

# Κεφάλαιο 3

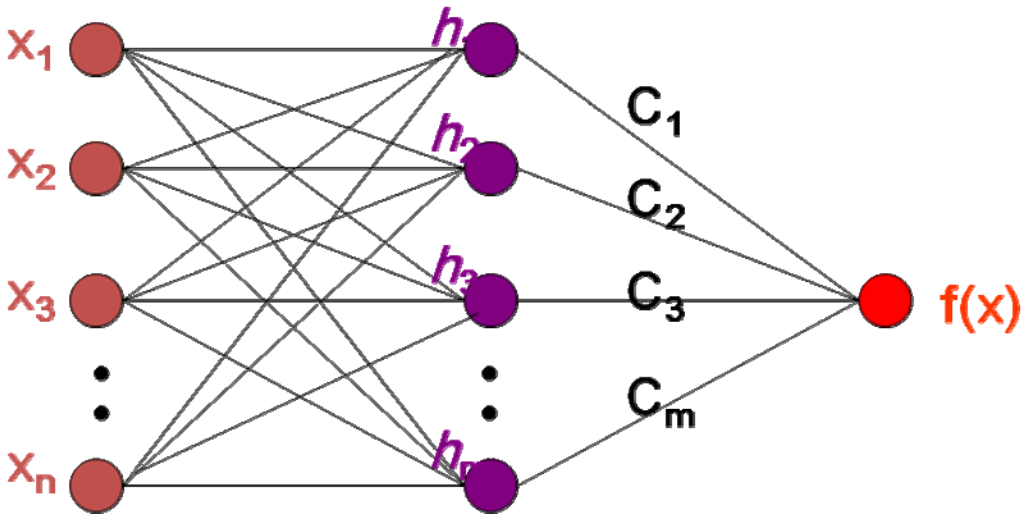
## Radial Basis Function Networks

3.1 Εισαγωγή στα RBF……...……... 11
3.2 Τρόπος λειτουργίας……...……..... 13
3.3 Εκπαίδευση με σταθερά κέντρα……...……. 14
3.4 Εκπαίδευση με μεταβλητά κέντρα……...……. 17

### 3.1 Radial Basis Function Networks – Εισαγωγή

Τα RBF ανήκουν στην κατηγορία της επιβλεπόμενης μάθησης αφού κατά την εκπαίδευση του δικτύου γνωρίζουμε και χρησιμοποιούμε το επιθυμητό αποτέλεσμα. Στα RBF υπάρχει ένα συνδεδεμένο δίκτυο με ένα επίπεδο εισόδου, ένα κρυφό επίπεδο και ένα επίπεδο εξόδου. Τα input δίνονται σε όλους τους κόμβους του κρυφού επιπέδου (οι οποίοι θα αναφέρονται σαν κέντρα), στο οποίο υπολογίζεται η ευκλείδεια απόσταση μεταξύ του input vector και του centre vector. Ακολούθως χρησιμοποιείται η Γκαουσιανή συνάρτηση για να υπολογιστεί η έξοδος που θα σταλεί στο output layer.
Η επιτυχία του σκοπού αυτού στηρίζεται στο θεώρημα του Cover για το διαχωρισμό των προτύπων. Σύμφωνα λοιπόν με τον Cover ένα πολύπλοκο πρόβλημα κατηγοριοποίησης προτύπων που αποτιμάται μη γραμμικά σε διανυσματικό χώρο μεγαλύτερης διαστατικότητας είναι πιο πιθανό να είναι γραμμικά διαχωρίσιμο παρά σε διανυσματικό χώρο μικρότερης διαστατικότητας. Η αποτίμηση αυτή σε διανυσματικό χώρο μεγαλύτερης διαστατικότητας και η μετατροπή των μη γραμμικά διαχωρίσιμων προβλημάτων σε γραμμικά διαχωρίσιμα επιτυγχάνεται με δύο τρόπους[5][6].

1 ) Περνώντας τα δεδομένα μας μέσα από τις Γκαουσιανές συναρτήσεις.
2 ) Αυξάνοντας τον αριθμό των νευρώνων στο κρυφό επίπεδο.




*Σχήμα 3.1Δομή RBF Δικτύου*

Στα RBF Networks χρησιμοποιούνται 2 μέθοδοι για την εκπαίδευση του δικτύου, με σταθερά και μεταβλητά κέντρα.

Στην εκπαίδευση με σταθερά κέντρα η διαδικασία έχει ως εξής:

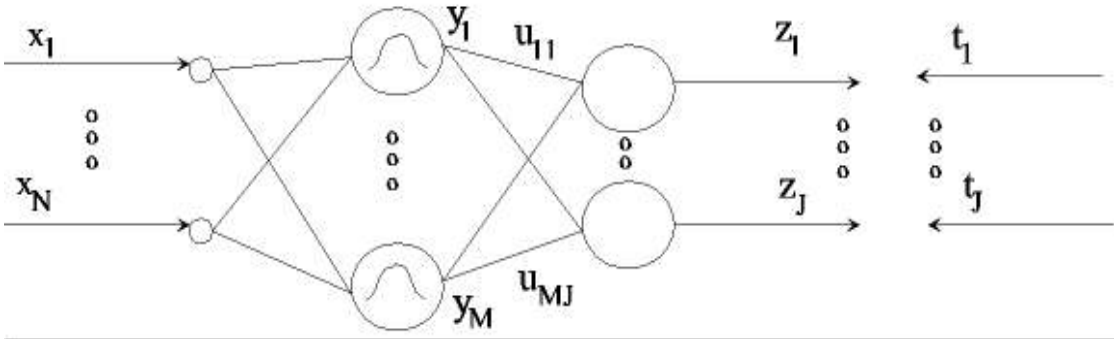
Κατ' αρχάς επιλέγουμε τα κέντρα. Τα κέντρα αυτά θα παραμείνουν σταθερά κατά την διάρκεια της εκπαίδευσης μας και θα μεταβάλλονται μόνο τα coefficients. Για κάθε ζεύγος δεδομένων (είσοδος, επιθυμητή έξοδος) θα δημιουργείται μια εξίσωση. Λύνοντας το σύστημα των εξισώσεων βρίσκουμε τα coefficients. Πολύ σημαντική είναι η επιλογή των κέντρων, του πλάτους της Γκαουσιανής συνάρτησης (σ), καθώς και η αρχικοποίηση των coefficients.

Η μέθοδος των σταθερών κέντρων χρησιμοποιείται όταν χρειάζεται ακρίβεια στο πρόβλημα μας. Αν για παράδειγμα σε κάποιο πρόβλημα χρειάζεται παρεμβολή τότε χρησιμοποιώ τη μέθοδο των σταθερών κέντρων με τον αριθμό των κέντρων να είναι ίσος με τον αριθμό των εισόδων.

Στην εκπαίδευση με μεταβλητά κέντρα η διαδικασία έχει ως εξής:

Η διαδικασία στηρίζεται στη μέθοδο καταβάσεως κλίσης (αλλάζουν κέντρα , διασπορές, βάρη). Και σε αυτή την περίπτωση επιλέγουμε τα κέντρα και την διασπορά. Στη συνέχεια αρχικοποιούμε τα βάρη με μικρές τυχαίες τιμές. Αφού βρούμε την πραγματική έξοδο, με την μέθοδο κατάβασης κλίσης βρίσκουμε τα νέα κέντρα, διασπορές, βάρη. Όταν το δίκτυο δείξει ικανοποιητική σύγκλιση, τότε σταματούμε. Η διαδικασία είναι πιο απλή από αυτή του Back Propagation γιατί σε αυτή την περίπτωση έχουμε μόνο ένα στρώμα από βάρη.

Η μέθοδος των μεταβλητών κέντρων χρησιμοποιείται όταν θέλω το δίκτυο μου να μάθει κατά προσέγγιση ένα πρόβλημα και όχι παρεμβολή. Ο αριθμός των κέντρων είναι συνήθως μικρότερος από τον αριθμό των εισόδων.




*Σχήμα 3.2Μαθηση με μεταβλητά κέντρα*

### 3.2 Radial Basis Function Networks - Τρόπος λειτουργίας

Η Radial Basis Function προσεγγίζει ένα πρόβλημα με μια λειτουργία προσέγγισης διαδικασίας (function approximation problem) [6]. Με κάποιες προκαθορισμένες συνθήκες, η Radial Basis Function είναι ικανή να προσεγγίσει καλά οποιαδήποτε εξίσωση. Η Radial Basis Function αποτελείται από τρία στρώματα :

- το στρώμα εισόδου (input layer) στο οποίο εισέρχονται τα δεδομένα
- το στρώμα εξόδου (output layer) στο οποίο "παράγονται" τα δεδομένα εξόδου
- το κρυφό στρώμα (hidden layer) στο οποίο εφαρμόζεται η RBF στα δεδομένα εισόδου.

Η αρχιτεκτονική αυτή είναι παρόμοια με την αρχιτεκτονική των Multilayer Perceptrons. Υπάρχουν όμως βασικές διαφορές που τις κάνουν να ξεχωρίζουν. Μια βασική διαφορά και ίσως η πιο σημαντική, είναι πως εάν επιλέξουμε κατάλληλα τις παραμέτρους της RBF τότε η διαδικασία εξισώνεται με την διαδικασία εκμάθησης ενός Simple Layer Perceptron η οποία είναι πολύ πιο γρήγορη απ' ότι η εκμάθηση ενός Multilayer Perceptron, και επομένως έχουμε γρηγορότερη σύγκλιση.

Η μόνη διαφορά μεταξύ του SLP με γραμμική εξίσωση εξόδου ενεργοποίησης και ενός RBF με την ίδια εξίσωση είναι ότι η έξοδος του **SLP** μπορεί να εκφραστεί ως

**yi(p) = w0 + w1x1(p) + w2x2(p) + ……..+ xK(p)**

ενώ η έξοδος του **RBF** μπορεί να εκφραστεί ως

**yi(p) = w0 + w1g1(x(p)) + w2g2(x(p)) + ……..+ gJ(x(p))**

όπου g παριστάνει μια εξίσωση μη γραμμικού μετασχηματισμού, σύμφωνα με την οποία τα δεδομένα εισόδου συνεχώς μεταβάλλονται καθώς διαδίδονται από το input στο hidden layer.

Το βασικό πλεονέκτημα των Radial Basis Function Networks είναι ότι ενώ έχουν παρόμοια δομή με τα Simple Layer Perceptrons, η οποία είναι απλή, ωστόσο λόγω της δομής input-hidden layer, μπορεί να επιλύσει προβλήματα τα οποία είναι άλυτα από τα Simple Layer Perceptrons. Τέτοια προβλήματα είναι προβλήματα στα οποία τα δεδομένα εισόδου είναι γραμμικά εξαρτημένα. Η μη γραμμικότητα λοιπόν στους κόμβους του hidden layer είναι εκείνη η οποία δίνει αυτή την δυνατότητα επίλυσης.

### 3.3 Radial Basis Function Networks – Μέθοδος σταθερών κέντρων

Όπως αναφέρθηκε και στην εισαγωγή, στα Radial Basis Function Networks υπάρχουν δύο μέθοδοι εκπαίδευσης. Στη μέθοδο των σταθερών κέντρων κατ' αρχάς επιλέγονται τα κέντρα. Τα κέντρα αυτά θα παραμείνουν σταθερά κατά την διάρκεια της εκπαίδευσης μας και θα μεταβάλλονται μόνο τα coefficients άρα η κατάλληλη επιλογή των κέντρων είναι πολύ σημαντική για την μετέπειτα λειτουργία του δικτύου. Επίσης πολύ σημαντικός είναι ο αριθμός των κέντρων. Αν και δεν υπάρχει συγκεκριμένη μέθοδος που να μας καθοδηγεί στο να βρούμε τον βέλτιστο αριθμό κέντρων για κάποιο συγκεκριμένο πρόβλημα, περιληπτικά θα αναφέρουμε ότι πολύ μικρός η πολύ μεγάλος αριθμός κέντρων είναι βέβαιο ότι θα οδηγήσει σε άσχημα αποτελέσματα, ως επακόλουθο της μη ορθής εκπαίδευσης του δικτύου [5].

Η έξοδος του δικτύου δίνεται από την εξίσωση :

$$y = \sum_{i=1}^{n} c_i \, \Phi(\|x - R_i\|)$$

όπου $c_i$ είναι το coefficient του κέντρου i

και $\Phi(\|x - R_i\|)$ είναι το αποτέλεσμα της μη γραμμικής συνάρτησης Φ (συνήθως χρησιμοποιείται η Γκαουσιανή) στη διαφορά του διανύσματος εισόδου με το διάνυσμα του κέντρου i (στην απόλυτη τιμή της).

Συνεπώς έχουμε ότι $y = c_1\Phi(x, R_1) + c_2\Phi(x, R_2) + \cdots + c_n\Phi(x, R_n)$

όπου $\Phi(x, R_i) = \Phi(\|x - R_i\|)$

Έτσι για κάθε ζεύγος δεδομένων εισόδου και επιθυμητής εξόδου από το Training Set δημιουργείται ένα ζεύγος τιμών για τα x και y. Έτσι, αφού συνήθως ο αριθμός των ζευγαριών αυτών που προκύπτουν είναι μεγαλύτερος από τον αριθμό των αγνώστων c, μπορούμε να λύσουμε ένα μαθηματικό σύστημα και να βρούμε τα βάρη c. Ο αριθμός των εξισώσεων που χρησιμοποιούμε είναι λίγο μεγαλύτερος από των αριθμό των κέντρων.

$$y_1 = c_1\Phi(x_1, R_1) + c_2\Phi(x_1, R_2) + \cdots + c_n\Phi(x_1, R_n)$$
$$y_2 = c_1\Phi(x_2, R_1) + c_2\Phi(x_2, R_2) + \cdots + c_n\Phi(x_2, R_n)$$
$$y_3 = c_1\Phi(x_3, R_1) + c_2\Phi(x_3, R_2) + \cdots + c_n\Phi(x_3, R_n)$$
$$\vdots$$
$$y_m = c_1\Phi(x_m, R_1) + c_2\Phi(x_m, R_2) + \cdots + c_n\Phi(x_m, R_n)$$

Όσον αφορά την μη γραμμική συνάρτηση Φ που θα χρησιμοποιήσουμε, όπως ανέφερα και προηγουμένως η πιο συνηθισμένη επιλογή είναι η Γκαουσιανή συνάρτηση. Για κάποια προβλήματα όμως ίσως να δίνει καλύτερα αποτελέσματα η χρήση κάποιας άλλης γραμμικής συνάρτησης. Μερικές από τις πιο συνηθισμένες επιλογές δίνονται πιο κάτω [5].

$\Phi(x) = x$ **linear function**

$\Phi(x) = x^3$ **cubic approximation**

$\Phi(x) = x^2 \ln x$ **thin-plate-spline function**

$\Phi(x) = \exp(-x^2 / \sigma^2)$ **Gaussian function**

$\Phi(x) = \sqrt{x^2 + \sigma^2}$ **multiquadratic function**

$\Phi(x) = \frac{1}{\sqrt{x^2 + \sigma^2}}$ **inverse multiquadratic function**

Τέλος, ας αναφέρουμε τα βήματα του αλγορίθμου μάθησης RBF με σταθερά κέντρα συγκεντρωμένα.

1) Διάλεξε τα κατάλληλα κέντρα **(R)**.

2) Διάλεξε την κατάλληλη διασπορά **(σ)**.

3) Αρχικοποίησε τα βάρη **(c)** με μικρές τυχαίες τιμές.

4) Υπολόγισε την έξοδο από την σχέση $y = c * \Phi(x_n, R)$

5) Βρες τα βέλτιστα βάρη χρησιμοποιώντας την $c = y * A^{-1}$

### 3.4 Radial Basis Function Networks – Μέθοδος μεταβλητών κέντρων

Στη μέθοδο μεταβλητών κέντρων χρησιμοποιούμε ένα στοχαστικό αλγόριθμο ταχυτέρας καθόδου [5]. Η διαδικασία στηρίζεται στη μέθοδο καταβάσεως κλίσης (αλλάζουν κέντρα , διασπορές, βάρη). Και σε αυτή την περίπτωση επιλέγουμε τα κέντρα και την διασπορά. Στη συνέχεια αρχικοποιούμε τα βάρη με μικρές τυχαίες τιμές. Αφού βρούμε την πραγματική έξοδο, με την μέθοδο κατάβασης κλίσης βρίσκουμε τα νέα κέντρα, διασπορές, βάρη. Όταν το δίκτυο δείξει ικανοποιητική σύγκλιση, τότε σταματούμε. Η διαδικασία είναι πιο απλή από αυτή του Back Propagation γιατί σε αυτή την περίπτωση έχουμε μόνο ένα στρώμα από βάρη.

Το στιγμιαίο συνολικό σφάλμα δίνεται από την έκφραση :

$$J[k] = \frac{1}{2}[e[k]]^2 = \frac{1}{2}\left\{d[k] - \sum_{k=1}^{n} c_k[k] * \Phi\,(x[k], R_k[k])\right\}^2$$

Όπου **:**

$d[k]$ είναι το επιθυμητό αποτέλεσμα

$\sum_{k=1}^{n} c_k[k] * \Phi\,(x[k], R_k[k])$ είναι το πραγματικό αποτέλεσμα

Αυτό που επιδιώκουμε είναι η ελαχιστοποίηση της J μεταβλητής, με τρείς όμως μεταβλητές αυτή την φορά αντί για μια όπως είχαμε στο back Propagation.Με δεδομένο λοιπόν ότι θα χρησιμοποιήσουμε την Γκαουσιανή συνάρτηση τότε η εξίσωση του στιγμιαίου συνολικού σφάλματος μετατρέπεται σε :

$$\mathbf{J[k]} = \frac{1}{2}\left\{d[k] - \sum_{k=1}^{n} c_k[k] * e^{-\left(\frac{\|x[k], R_k[k]\|^2}{\sigma_k^2[k]}\right)}\right\}^2$$

Τώρα, με την μέθοδο κατάβασης κλίσης θα επιδιώξουμε να βρούμε τα ολικά ελάχιστα. Για κάθε ένα από τα R, σ και c έχουμε τις αντίστοιχες εξισώσεις ταχυτέρας καθόδου :

$$c_k[k+1] = c_k[k] - n_c \frac{\delta J[k]}{\delta c_k}$$

$$R_k[k+1] = R_k[k] - n_R \frac{\delta J[k]}{\delta R_k}$$

$$\sigma_k[k+1] = \sigma_k[k] - n_\sigma \frac{\delta J[k]}{\delta \sigma_k}$$

Όπου :

$c_k[k+1], R_k[k+1], \sigma_k[k+1]$ είναι τα νέα βάρη, κέντρα, διασπορές

$c_k[k], R_k[k], \sigma_k[k]$ είναι τα υπάρχοντα βάρη, κέντρα, διασπορές

$n$ είναι ο ρυθμός μάθησης

$n_c \frac{\delta J[k]}{\delta c_k}, n_R \frac{\delta J[k]}{\delta R_k}, n_\sigma \frac{\delta J[k]}{\delta \sigma_k}$ είναι η μερική παράγωγος της συναρτήσεως του λάθους ως προς τα βάρη, κέντρα και διασπορά αντίστοιχα.

Παίρνουμε δηλαδή την μερική παράγωγο για κάθε μεταβλητή ως προς την ίδια την μεταβλητή κάθε φορά με σκοπό να βρούμε τις "βέλτιστες" παραμέτρους που θα μας ελαχιστοποιήσουν το σφάλμα.
Ας δούμε και πάλι τα βήματα του αλγορίθμου μάθησης RBF με μεταβλητά κέντρα συγκεντρωμένα :

1) Διάλεξε τα κατάλληλα κέντρα.

2) Διάλεξε τα κατάλληλα σ.

3) Αρχικοποίησε τα βάρη c με μικρές τυχαίες τιμές.

4) Δώσε στο δίκτυο ένα ζεύγος εισόδου – επιθυμητής εξόδου και βρες την έξοδο από την σχέση :

$$y[k] = \sum_{k=1}^{n} c_k \Phi(x[k], R_k, \sigma_k)$$

5) Βρες τις νέες τιμές των σ , R και c από τις εξισώσεις :

$$c[k+1] = c[k] + n_c * e[k] * \Psi[k]$$

$$R[k+1] = R_k[k] + n_R * \frac{e[k] * c_k[k]}{\sigma_k^2[k]} * \Phi(x[k], R_k, \sigma_k)\,[x[k] - R_k[k]]$$

$$\sigma[k+1] = \sigma_k[k] + n_\sigma * \frac{e[k] * c_k[k]}{\sigma_k^3[k]} * \Phi(x[k], R_k, \sigma_k)\,\|x[k] - R_k[k]\|^2$$

6) Αν το δίκτυο έδειξε ικανοποιητική σύγκλιση σταμάτα. Αν όχι τότε πήγαινε στο βήμα 4.

# Κεφάλαιο 4

## Support Vector Machines (Μηχανές διανυσμάτων υποστήριξης)


4.1 Εισαγωγή στις Μηχανές διανυσμάτων υποστήριξης…… 20

4.2 Τρόπος λειτουργίας Μηχανών διανυσμάτων υποστήριξης……..……..... 22

4.2.1 Μέθοδος Κυρτών πολυγώνων……..... 23

4.2.2 Μέθοδος μεγίστου διαστήματος μεταξύ παράλληλων επιπέδων…….. 24

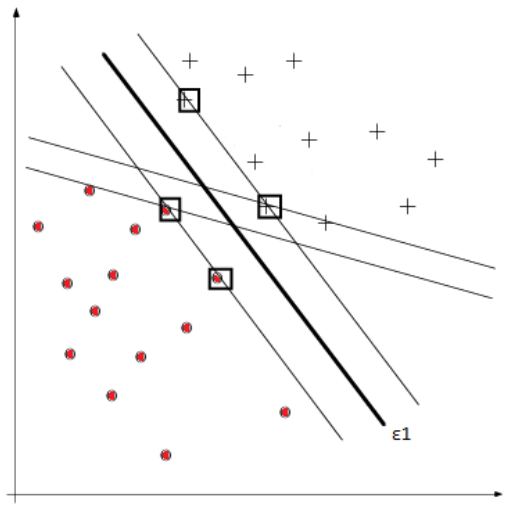
4.2.3 Μη γραμμικά διαχωρίσιμα πρότυπα…..……...……..... 25

4.2.4 Τα Support Vector Machines στην έρευνα μου……..... 26


### 4.1 Εισαγωγή στις Μηχανές διανυσμάτων υποστήριξης

Οι μηχανές διανυσμάτων υποστήριξης (Support Vector Machines, SVMs) αποτελούν αλγορίθμους ταξινόμησης προτύπων. Οι αλγόριθμοι αυτοί βασίστηκαν στη στατιστική θεωρία μάθησης και αναπτύχθηκαν από τον Vapnik το 1963. [12] [13]

Για την ταξινόμηση, οι μηχανές διανυσμάτων υποστήριξης αναζητούν ένα υπερεπίπεδο (hyperplane) στο χώρο των πιθανών εισόδων. Το υπερεπίπεδο αυτό διαχωρίζει τα θετικά από τα αρνητικά παραδείγματα προσπαθώντας να επιτύχει την μέγιστη απόσταση μεταξύ του πιο κοντινού θετικού και αρνητικού παραδείγματος.




*Σχήμα 4.1Διαχωρισμός μεγίστου περιθωρίου*

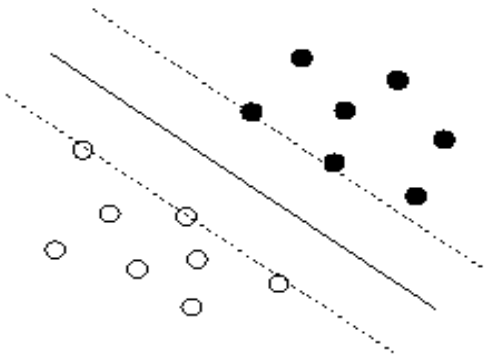
Όπως φαίνεται στο σχήμα, υπάρχει μεγάλος αριθμός υπερεπιπέδων που μπορούν να διαχωρίσουν τα θετικά και τα αρνητικά παραδείγματα [10]. Ο αλγόριθμος όμως θα επιλέξει το υπερεπίπεδο εκείνο με το μεγαλύτερο περιθώριο μεταξύ του κοντινότερου θετικού και αρνητικού παραδείγματος, το οποίο στη περίπτωσή μας είναι το ε1.

Οι μηχανές διανυσμάτων υποστήριξης ανήκουν στην κατηγορία των αλγορίθμων επιβλεπόμενης μάθησης και εκτός της χρήσης τους για ταξινόμηση έχουν την ικανότητα να προσεγγίσουν και συναρτήσεις. Αποτελούν και αυτοί μία μέθοδο ταξινόμησης προτύπων και δεδομένων όπως είναι και τα νευρωνικά δίκτυα. Η διαφορά των SVM από τις υπόλοιπες μεθόδους είναι ότι ελαχιστοποιείται το εμπειρικό λάθος (empirical risk) ενώ ταυτόχρονα μεγιστοποιείται η απόσταση διαχωρισμού (margin).

Για τον λόγο αυτό μπορεί να τους συναντήσουμε κάποιες φορές και σαν ταξινομητές μεγίστου περιθωρίου (Maximum Margin Classifier). Οι ιδιαίτερες ικανότητες των μηχανών αυτών χρησιμοποιούνται όταν χρειάζεται να ταξινομηθούν δεδομένα τα οποία ανήκουν σε επίπεδο μεγαλύτερο των δύο διαστάσεων. Σε αυτή την περίπτωση ο διαχωρισμός αυτών των δεδομένων γίνεται με τη βοήθεια ενός υπερεπιπέδου n-1 διαστάσεων, με τη χρήση δηλαδή ενός γραμμικού ταξινομητή.

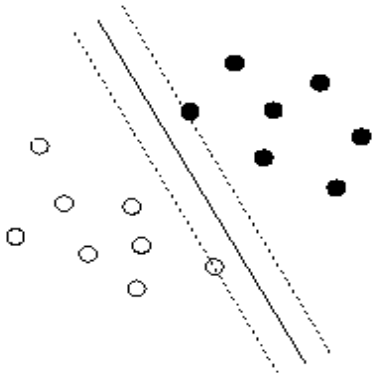
Κάτι τέτοιο μπορεί να γίνει εφικτό με την χρήση και άλλων ταξινομητών, όπως για παράδειγμα είναι η γραμμική παλινδρόμηση με χρήση perceptron [11]. Η μέθοδος αυτή χρησιμοποιείται σε δεδομένα που μπορούν να διαχωριστούν τέλεια σε ομάδες (linearly seperable), με αποτέλεσμα ο αλγόριθμος αυτός να εκτελείται σε πεπερασμένο αριθμό επαναλήψεων. Άρα και σε αυτή την περίπτωση βρίσκεται ένα υπερεπίπεδο για το διαχωρισμό των δεδομένων όπως και σε άλλους γραμμικούς ταξινομητές. Στον αλγόριθμο SVM όμως, αυτό που τον κάνει να διαφέρει από τους άλλους είναι το γεγονός ότι επιτυγχάνεται ο μέγιστος δυνατός διαχωρισμός μεταξύ των σημείων δεδομένων, δηλαδή υπολογίζεται το μέγιστο υπερεπίπεδο. Γι' αυτό το λόγο το επίπεδο ονομάζεται επίπεδο μέγιστου διαχωρισμού και αντίστοιχα ο ταξινομητής ονομάζεται ταξινομητής μεγίστου περιθωρίου.

**A)**



*Σχήμα 4.2Διαχωρισμός με SVM*

**B)**



*Σχήμα 4.3Διαχωρισμός με άλλο ταξινομητή*

Όπως φαίνεται στο σχήμα οι 2 γραμμικοί ταξινομητές διαχωρίζουν τα σημεία στο επίπεδο. Στη πρώτη περίπτωση, στην οποία χρησιμοποιείται ο αλγόριθμος SVM, επιτυγχάνεται μέγιστος διαχωρισμός (μέγιστο περιθώριο), ενώ στη δεύτερη στην οποία γίνεται χρήση κάποιου άλλου ταξινομητή όχι.

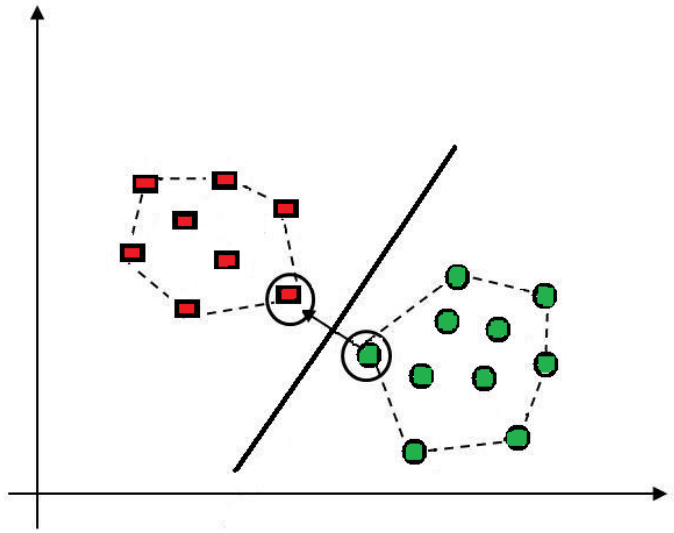
### 4.2 Τρόπος Λειτουργίας Μηχανών διανυσμάτων υποστήριξης

Όπως ανέφερα και προηγουμένως σκοπός της ταξινόμησης είναι να βρεθεί ένας κανόνας, ο οποίος να κατατάσσει ένα αντικείμενο σε μία από τις πιθανές κλάσεις. Έστω ότι υπάρχουν μόνο δύο διαφορετικές κλάσεις και τα δεδομένα μπορούν να χωριστούν τέλεια σε ομάδες (η απλούστερη περίπτωση). Το επίπεδο το οποίο κατηγοριοποιεί σωστά τις δύο κλάσεις διαχωρίζοντας μεταξύ τους τα σημεία στο επίπεδο μπορεί να βρεθεί πολύ εύκολα [8].

Τέτοια επίπεδα όμως υπάρχουν άπειρα όπως είπαμε και προηγουμένως, αφού μπορούν να χαραχθούν άπειρες ευθείες γραμμές που διαχωρίζουν τα δεδομένα του προηγούμενου σχήματος. Αυτό όμως που ζητείται είναι η εύρεση του επιπέδου με τη μέγιστη απόσταση από τις δύο κλάσεις αφού με τον τρόπο αυτό οι κλάσεις είναι καλύτερα διαχωρισμένες και σύμφωνα με τη στατιστική θεωρία μάθησης αυτό θα οδηγήσει στο μέγιστο βαθμό γενικότητας κατά την κατάταξη αταξινόμητων παραδειγμάτων. Για την κατασκευή αυτού του επιπέδου υπάρχουν δυο τρόποι. Ο πρώτος τρόπος είναι η μέθοδος των κυρτών πολυγώνων ενώ ο δεύτερος τρόπος στοχεύει στο να βρει το μέγιστο διάστημα μεταξύ δύο συγκεκριμένων παράλληλων επιπέδων. Πιο κάτω θα προσπαθήσω να περιγράψω σε συντομία τις δύο μεθόδους.

### 4.2.1 Μέθοδος Κυρτών Πολυγώνων

Εδώ ο σκοπός μας είναι να δημιουργήσουμε κυρτά πολύγωνα τα οποία να είναι τα μικρότερα από ένα σύνολο πολυγώνων που έχουν στο σύνορο είτε στο εσωτερικό τους τα αντικείμενα των κλάσεων. Μετά τον υπολογισμό αυτών των πολυγώνων προσπαθούμε να εντοπίσουμε δύο σημεία τους, ένα από το ένα και ένα από το άλλο, που έχουν την μικρότερη απόσταση. Στη συνέχεια θα διχοτομήσουμε τα δύο αυτά σημεία και θα βρούμε το επίπεδο ταξινόμησης [8].

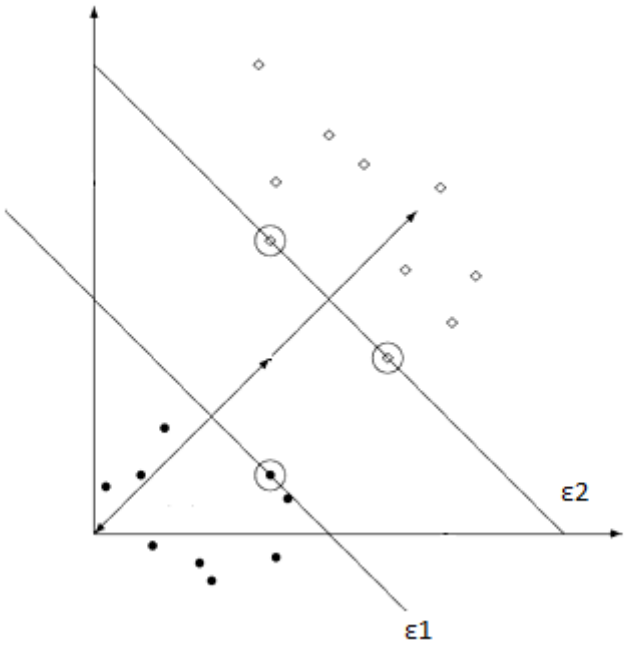


*Σχήμα 4.4Μέθοδος κυρτών πολυγώνων*

Όπως βλέπουμε στο σχήμα πιο πάνω έχουν δημιουργηθεί δύο κυρτά πολύγωνα για τα δύο γραμμικά διαχωρίσιμα σύνολα των κλάσεων του προβλήματος μας. Τα κυκλωμένα σημεία αποτελούν τα πιο κοντινά σημεία των περιφερειών των δύο πολυγώνων. Διχοτομώντας τα 2 αυτά σημεία, προκύπτει το επίπεδο που μας ενδιαφέρει.

### 4.2.2 Μέθοδος μεγίστου διαστήματος μεταξύ παράλληλων επιπέδων

Σε αυτή τη μέθοδο σκοπός μας είναι να βρεθεί το μέγιστο διάστημα μεταξύ δύο συγκεκριμένων παράλληλων επιπέδων. Τα επίπεδα αυτά είναι υποστηριχτικά και η βασική προδιαγραφή τους είναι ότι αφήνουν όλα τα σημεία του συνόλου που υποστηρίζουν από τη μία πλευρά του επιπέδου. Τα στοιχεία (patterns) που βρίσκονται πάνω στα υποστηριχτικά επίπεδα ονομάζονται διανύσματα στήριξης [10].




*Σχήμα 4.5Μέθοδος μεγίστου διαστήματος μεταξύ παράλληλων επιπέδων*

Όπως βλέπουμε στο σχήμα επιλέγονται δύο επίπεδα τα οποία αφήνουν όλα τα σημεία του συνόλου που υποστηρίζουν από την μία πλευρά. Το ένα αφήνει από τα αριστερά το σύνολο με τις τελείες, ενώ το άλλο αφήνει από τα δεξιά το σύνολο με τα τετραγωνάκια. Τα κυκλωμένα είναι τα διανύσματα στήριξης. Σκοπός μας σε αυτή την περίπτωση είναι με την βοήθεια των υποστηρικτικών επιπέδων ε1 και ε2, να βρούμε το υπερεπίπεδο το οποίο μεγιστοποιεί το περιθώριο. Το πρόβλημα αυτό είναι ένα συνηθισμένο πρόβλημα βελτιστοποίησης και μπορεί να λυθεί πολύ εύκολα χρησιμοποιώντας τη μέθοδο των πολλαπλασιαστών Lagrange. Έτσι προκύπτει το υπερεπίπεδο που μας ενδιαφέρει.

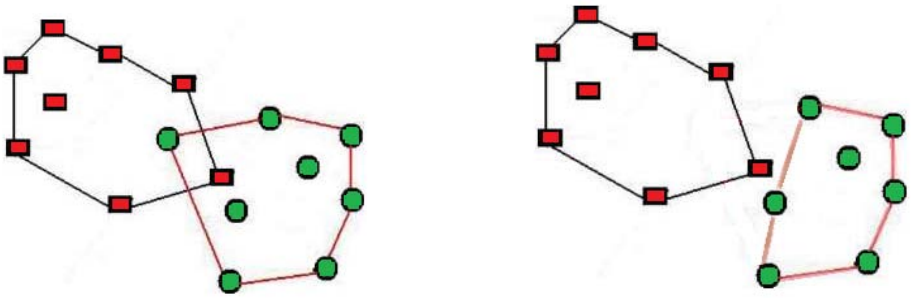
### 4.2.3 Μη γραμμικά διαχωρίσιμα πρότυπα

Η περίπτωση των γραμμικά διαχωρίσιμων συνόλων που εξετάσαμε πιο πριν, είναι η εύκολη – ιδανική περίπτωση. Μερικές φορές ο πληθυσμός από τον οποίο προέρχονται τα δεδομένα εκπαίδευσης, δεν είναι γραμμικά διαχωρίσιμος και κατ' επέκταση δεν υπάρχει υπερεπίπεδο (γραμμικό) που να διαχωρίζει τα θετικά και τα αρνητικά πρότυπα. Σε αυτή την περίπτωση υπάρχουν και πάλι δυο μέθοδοι που μπορούμε να ακολουθήσουμε για να πετύχουμε το στόχο μας [8] [10].

Η πρώτη λύση είναι να μετατρέψουμε τα δεδομένα μας σε διανύσματα ενός χώρου με μεγαλύτερες διαστάσεις, έτσι ώστε στο νέο χώρο το πρόβλημα μας να είναι πλέον γραμμικά διαχωρίσιμο και να μπορέσουμε να βρούμε το γραμμικό υπερεπίπεδο που διαχωρίζει πλήρως τα θετικά από τα αρνητικά παραδείγματα. Το μεγάλο μειονέκτημα της λύσης αυτής είναι το μεγάλο υπολογιστικό κόστος που προκύπτει από την μετατροπή των δεδομένων μας σε διανύσματα ενός χώρου με μεγαλύτερες διαστάσεις.

Η δεύτερη λύση είναι να αφαιρεθούν τα σημεία τα οποία καθιστούν το πρόβλημά μας μη γραμμικά διαχωρίσιμο.

**A)** **B)**



*Σχήμα 4.6 Μη γραμμικά διαχωρίσιμα σύνολα* *Σχήμα 4.7 Γραμμικά διαχωρίσιμα σύνολα*

Όπως βλέπουμε πιο πάνω, στο σχήμα Α τα σύνολα δεν είναι γραμμικά διαχωρίσιμα. Δηλαδή δεν υπάρχει υπερεπίπεδο που διαχωρίζει πλήρως τα θετικά από τα αρνητικά παραδείγματα. Γι' αυτό τον λόγο, τα πολύγωνα που κατασκευάζονται τέμνονται μεταξύ τους. Για να λυθεί το πρόβλημα αρκεί η αφαίρεση του ενός κύκλου που βρίσκεται μέσα στο πολύγωνο των τετραγώνων. Τώρα όπως βλέπουμε στο σχήμα Β έχουμε πολύγωνα τα οποία δεν τέμνονται και μπορούμε να δουλέψουμε όπως προηγουμένως και να βρούμε το γραμμικό υπερεπίπεδο που διαχωρίζει πλήρως τα θετικά από τα αρνητικά παραδείγματα.

### 4.2.4 Τα Support Vector Machines στην έρευνα μου

Τα Support Vector Machines θεωρητικά έχουν μεγαλύτερες δυνατότητες από τις υπόλοιπες μεθόδους που αφορούν ή στηρίζονται στα νευρωνικά δίκτυα. Σκοπός μου σε αυτό το μέρος της εργασίας θα είναι η εξακρίβωση αυτής της υπόθεσης στο δικό μας πρόβλημα, αν δηλαδή η χρησιμοποίηση των SVM's μπορεί να προσφέρει στην πρόβλεψη σημείων στο ποδοσφαιρικό στοίχημα με ποσοστά επιτυχίας μεγαλύτερα από αυτά που έχουν επιτευχθεί με τις υπόλοιπες μεθόδους που έχουν υλοποιηθεί. Για αυτό τον σκοπό θα χρησιμοποιήσω ένα simulator ο οποίος θα χρησιμοποιεί Support Vector Machine για την επίλυση του ίδιου προβλήματος.

Ο προσομοιωτής που αυτός λέγεται SVM Classifer. Έχει εκδοθεί το Δεκέμβριο του 2006, από τους Mehdi Pirooznia και Yooping Deng στο University of Southern Mississippi. Περισσότερα για τον τρόπο λειτουργίας του και τα αποτελέσματα που προέκυψαν θα αναφερθούν σε επόμενο στάδιο της έρευνας. Σε αυτό το σημείο θα ήθελα απλώς να αναφέρω ότι ο σκοπός μου είναι να συγκριθούν τα αποτελέσματα του προγράμματος μου με αυτά των προαναφερθέντων καθώς και αυτά που θα προκύψουν με την χρήση του SVM simulator έτσι ώστε να μπορέσουν να εξαχθούν συμπεράσματα για τις δυνατότητες και τα περιθώρια βελτίωσης που υπάρχουν στη συγκεκριμένη έρευνα.

# Κεφάλαιο 5

## Δεδομένα Εισόδου

5.1 Δεδομένα εισόδου που χρησιμοποιεί το δίκτυο……..... 27
5.2 Το Πρωτάθλημα της Championship……....……..... 33
5.3 Επεξεργασία Δεδομένων……...……..... 35
5.4 Κανονικοποίηση Τιμών……...……..... 38

### 5.1 Δεδομένα εισόδου που χρησιμοποιεί το δίκτυο

Σε αυτό το σημείο της διπλωματικής μου εργασίας θα αναλύσω το σύνολο των δεδομένων εισόδου που χρησιμοποιώ στο Radial Basis Function δίκτυο μου για την πρόβλεψη του αριθμού των τερμάτων που θα επιτευχθούν σε ένα ποδοσφαιρικό αγώνα. Τα ίδια δεδομένα επίσης θα χρησιμοποιηθούν και για την εκπαίδευση του SVM simulator.

Θα πρέπει να αναφέρω ότι τα Data Set που χρησιμοποιώ είναι αυτά που έχει προτείνει ο Θεοφάνης Νεοκλέους στη δική του διπλωματική εργασία, η οποία ολοκληρώθηκε το 2007 και είχε το ίδιο στόχο με την δική μου, χρησιμοποιώντας φυσικά εντελώς διαφορετικές μεθόδους.

Πόσα τέρματα θα σημειωθούν σε ένα συγκεκριμένο ποδοσφαιρικό παιχνίδι λοιπόν. Οι επιλογές για τα δεδομένα εισόδου που μπορούμε να χρησιμοποιήσουμε είναι αρκετές. Αποδεδειγμένα όμως, από τις προηγούμενες προσπάθειες που έγιναν τόσο από την Turner [3] όσο και από τον Νεοκλέους [4], αυτά που πραγματικά έχουν σημασία στη προσπάθεια για πρόβλεψη του αριθμού των τερμάτων που θα επιτευχθούν σε κάποιο παιχνίδι είναι τα δεδομένα που αναφέρονται σε αριθμούς και ποσοστά που έχουν άμεση σχέση με τα τέρματα που σκόραραν ή δέχτηκαν οι δυο ομάδες που θα αγωνιστούν στο παιχνίδι αυτό. Δεν θα χρησιμοποιηθούν δηλαδή δεδομένα που δεν έχουν άμεση σχέση με τα τέρματα που δέχονται και σκοράρουν οι δυο αντίπαλες ομάδες (όπως είναι η προϊστορία των δυο ομάδων, απουσίες παιχτών που μπορεί να έχουν, βαθμολογικό κίνητρο, βαθμολογική θέση, καιρικές συνθήκες κτλ). Ίσως σε κάποιο να ακούγεται

λογική και απαραίτητη η χρησιμοποίηση κάποιου από τα πιο πάνω στοιχεία αλλά οι έρευνες των 2 προαναφερθέντων έχουν αποδείξει το αντίθετο αφού τα νευρωνικά δίκτυα που κατασκεύασαν απομόνωσαν τα δεδομένα αυτά και στο τέλος όχι μόνο δεν βοηθούσαν στην εκπαίδευση του δικτύου αλλά την καθιστούσαν ακόμη πιο δύσκολη. Πιο κάτω θα αναφέρω ένα προς ένα τα δεδομένα που αποτελούν το διάνυσμα εισόδου που δίνεται στο Radial Basis Function δίκτυο μου για κάθε ποδοσφαιρικό παιχνίδι για το οποίο προβλέπει τον αριθμό τερμάτων που θα σημειωθούν σε αυτό.

1) **Μέσος όρος τερμάτων που πέτυχε η γηπεδούχος ομάδα στην έδρα της**

Υπολογισμός του μέσου όρου των τερμάτων που κατάφερε να πετύχει η γηπεδούχος ομάδα σε όλα τα παιχνίδια που αγωνίστηκε στην έδρα της μέχρι στιγμής στο πρωτάθλημα.

2) **Ποσοστό των παιχνιδιών που σκόραρε η γηπεδούχος ομάδα στην έδρα της**

Υπολογισμός του ποσοστού των παιχνιδιών που κατάφερε να πετύχει έστω και ένα τέρμα η γηπεδούχος ομάδα σε όλα τα παιχνίδια που αγωνίστηκε στην έδρα της μέχρι στιγμής στο πρωτάθλημα.

3) **Μέσος όρος τερμάτων που δέχτηκε η γηπεδούχος ομάδα στην έδρα της**

Υπολογισμός του μέσου όρου των τερμάτων που δέχτηκε η γηπεδούχος ομάδα σε όλα τα παιχνίδια που αγωνίστηκε στην έδρα της μέχρι στιγμής στο πρωτάθλημα.

4) **Ποσοστό των παιχνιδιών που δέχτηκε τέρμα η γηπεδούχος ομάδα στην έδρα της**

Υπολογισμός του ποσοστού των παιχνιδιών που δέχτηκε έστω και ένα τέρμα η γηπεδούχος ομάδα σε όλα τα παιχνίδια που αγωνίστηκε στην έδρα της μέχρι στιγμής στο πρωτάθλημα.

5) **Μέσος όρος τερμάτων που πέτυχε η φιλοξενούμενη ομάδα εκτός έδρας**

Υπολογισμός του μέσου όρου των τερμάτων που κατάφερε να πετύχει η φιλοξενούμενη ομάδα σε όλα τα παιχνίδια που αγωνίστηκε εκτός έδρας μέχρι στιγμής στο πρωτάθλημα.

6) **Ποσοστό των παιχνιδιών που σκόραρε η φιλοξενούμενη ομάδα εκτός έδρας**

Υπολογισμός του ποσοστού των παιχνιδιών που κατάφερε να πετύχει έστω και ένα τέρμα η φιλοξενούμενη ομάδα σε όλα τα παιχνίδια που αγωνίστηκε εκτός έδρας μέχρι στιγμής στο πρωτάθλημα.

7) **Μέσος όρος τερμάτων που δέχτηκε η φιλοξενούμενη ομάδα εκτός έδρας**

Υπολογισμός του μέσου όρου των τερμάτων που δέχτηκε η φιλοξενούμενη ομάδα σε όλα τα παιχνίδια που αγωνίστηκε εκτός έδρας μέχρι στιγμής στο πρωτάθλημα.

8) **Ποσοστό των παιχνιδιών που δέχτηκε τέρμα η φιλοξενούμενη ομάδα εκτός έδρας**

Υπολογισμός του ποσοστού των παιχνιδιών που δέχτηκε έστω και ένα τέρμα η φιλοξενούμενη ομάδα σε όλα τα παιχνίδια που αγωνίστηκε εκτός έδρας μέχρι στιγμής στο πρωτάθλημα.

9) **Ποσοστό των παιχνιδιών της γηπεδούχου ομάδας στην έδρα της στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα**

Υπολογισμός του ποσοστού των παιχνιδιών της γηπεδούχου στην έδρα της στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα και από τις δυο ομάδες.

10) **Ποσοστό των παιχνιδιών της φιλοξενούμενης ομάδας εκτός έδρας στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα**

Υπολογισμός του ποσοστού των παιχνιδιών της φιλοξενούμενης ομάδας εκτός έδρας στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα και από τις δυο ομάδες.

11) **Μέσος όρος τερμάτων που πέτυχε η γηπεδούχος ομάδα στην έδρα της στους τελευταίους έξι αγώνες**

Υπολογισμός του μέσου όρου των τερμάτων που κατάφερε να πετύχει η γηπεδούχος ομάδα σε όσα παιχνίδια αγωνίστηκε στην έδρα της τους τελευταίους έξι αγώνες.

**12) <u>Ποσοστό των παιχνιδιών που σκόραρε η γηπεδούχος ομάδα στην έδρα της τους τελευταίους έξι αγώνες</u>**

Υπολογισμός του ποσοστού των παιχνιδιών που κατάφερε να πετύχει έστω και ένα τέρμα η γηπεδούχος ομάδα σε όσα παιχνίδια αγωνίστηκε στην έδρα της τους τελευταίους έξι αγώνες.

**13) <u>Μέσος όρος τερμάτων που δέχτηκε η γηπεδούχος ομάδα στην έδρα της στους τελευταίους έξι αγώνες</u>**

Υπολογισμός του μέσου όρου των τερμάτων που δέχτηκε η γηπεδούχος ομάδα σε όσα παιχνίδια αγωνίστηκε στην έδρα της τους τελευταίους έξι αγώνες.

**14) <u>Ποσοστό των παιχνιδιών που δέχτηκε τέρμα η γηπεδούχος ομάδα στην έδρα της τους τελευταίους έξι αγώνες</u>**

Υπολογισμός του ποσοστού των παιχνιδιών που δέχτηκε έστω και ένα τέρμα η γηπεδούχος ομάδα σε όσα παιχνίδια αγωνίστηκε στην έδρα της τους τελευταίους έξι αγώνες.

**15) <u>Μέσος όρος τερμάτων που πέτυχε η φιλοξενούμενη ομάδα εκτός έδρας στους τελευταίους έξι αγώνες</u>**

Υπολογισμός του μέσου όρου των τερμάτων που κατάφερε να πετύχει η φιλοξενούμενη ομάδα σε όσα παιχνίδια αγωνίστηκε εκτός έδρας τους τελευταίους έξι αγώνες.

**16) <u>Ποσοστό των παιχνιδιών που σκόραρε η φιλοξενούμενη ομάδα εκτός έδρας τους τελευταίους έξι αγώνες</u>**

Υπολογισμός του ποσοστού των παιχνιδιών που κατάφερε να πετύχει έστω και ένα τέρμα η φιλοξενούμενη ομάδα σε όσα παιχνίδια αγωνίστηκε εκτός έδρας τους τελευταίους έξι αγώνες.

**17) <u>Μέσος όρος τερμάτων που δέχτηκε η φιλοξενούμενη ομάδα εκτός έδρας στους τελευταίους έξι αγώνες</u>**

Υπολογισμός του μέσου όρου των τερμάτων που δέχτηκε η φιλοξενούμενη ομάδα σε όσα παιχνίδια αγωνίστηκε εκτός έδρας τους τελευταίους έξι αγώνες.

18) **Ποσοστό των παιχνιδιών που δέχτηκε τέρμα η φιλοξενούμενη ομάδα εκτός έδρας τους τελευταίους έξι αγώνες**

Υπολογισμός του ποσοστού των παιχνιδιών που δέχτηκε έστω και ένα τέρμα η φιλοξενούμενη ομάδα σε όσα παιχνίδια αγωνίστηκε εκτός έδρας τους τελευταίους έξι αγώνες.

19) **Ποσοστό των παιχνιδιών της γηπεδούχου ομάδας στην έδρα της στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα στους τελευταίους έξι αγώνες**

Υπολογισμός του ποσοστού των παιχνιδιών της γηπεδούχου στην έδρα της στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα και από τις δυο ομάδες στους τελευταίους έξι αγώνες.

20) **Ποσοστό των παιχνιδιών της φιλοξενούμενης ομάδας εκτός έδρας στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα στους τελευταίους έξι αγώνες**

Υπολογισμός του ποσοστού των παιχνιδιών της φιλοξενούμενης ομάδας εκτός έδρας στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα και από τις δυο ομάδες στους τελευταίους έξι αγώνες.

21) **Μέσος όρος τερμάτων που επιτεύχθηκαν στα παιχνίδια που αγωνίστηκε η γηπεδούχος ομάδα.**

Υπολογισμός του μέσου όρου των τερμάτων που επιτεύχθηκαν σε όλα τα παιχνίδια που αγωνίστηκε η γηπεδούχος ομάδα τόσο εντός όσο και εκτός έδρας.

22) **Μέσος όρος τερμάτων που επιτεύχθηκαν στα παιχνίδια που αγωνίστηκε η φιλοξενούμενη ομάδα.**

Υπολογισμός του μέσου όρου των τερμάτων που επιτεύχθηκαν σε όλα τα παιχνίδια που αγωνίστηκε η φιλοξενούμενη ομάδα τόσο εντός όσο και εκτός έδρας.

23) **Ποσοστό των παιχνιδιών της γηπεδούχου ομάδας στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα**

Υπολογισμός του ποσοστού των παιχνιδιών στα οποία αγωνίστηκε η γηπεδούχος ομάδα τόσο εντός όσο και εκτός έδρας και στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα και από τις δυο ομάδες.

**24) <u>Ποσοστό των παιχνιδιών της φιλοξενούμενης ομάδας στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα</u>**

Υπολογισμός του ποσοστού των παιχνιδιών στα οποία αγωνίστηκε η φιλοξενούμενη ομάδα τόσο εντός όσο και εκτός έδρας και στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα και από τις δυο ομάδες.

**25) <u>Ποσοστό των παιχνιδιών της γηπεδούχου ομάδας στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα στους τελευταίους έξι αγώνες της</u>**

Υπολογισμός του ποσοστού των παιχνιδιών στα οποία αγωνίστηκε η γηπεδούχος ομάδα τόσο εντός όσο και εκτός έδρας τους τελευταίους έξι αγώνες και στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα και από τις δυο ομάδες.

**26) <u>Ποσοστό των παιχνιδιών της φιλοξενούμενης ομάδας στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα στους τελευταίους έξι αγώνες της</u>**

Υπολογισμός του ποσοστού των παιχνιδιών στα οποία αγωνίστηκε η φιλοξενούμενη ομάδα τόσο εντός όσο και εκτός έδρας τους τελευταίους έξι αγώνες και στα οποία σημειώθηκαν περισσότερα από 2.5 τέρματα και από τις δυο ομάδες.

## 5.2 Το πρωτάθλημα της Championship

Όπως ανέφερα στον πρόλογο μου, για την εκπαίδευση του δικτύου μου χρησιμοποίησα δεδομένα και αποτελέσματα της Championship, της δεύτερης εν τη τάξη ποδοσφαιρικής κατηγορίας στην Αγγλία.

Η κατηγορία αυτή αποτελείται από 24 ομάδες. Σε κάθε γύρο, η κάθε ομάδα αγωνίζεται με όλες τις άλλες δηλαδή δίνει 23 αγώνες. Το πρωτάθλημα έχει δυο γύρους άρα 46 αγωνιστικές. Συνολικά στο πρωτάθλημα διεξάγονται 552 παιχνίδια, αριθμός που θεωρείται αρκετά μεγάλος και σπάνια μπορείς να τον συναντήσεις σε άλλα πρωταθλήματα. Αυτός είναι ο πιο σημαντικός λόγος για τον οποίο επέλεξα το συγκεκριμένο πρωτάθλημα για την εκπαίδευση του δικτύου μου.

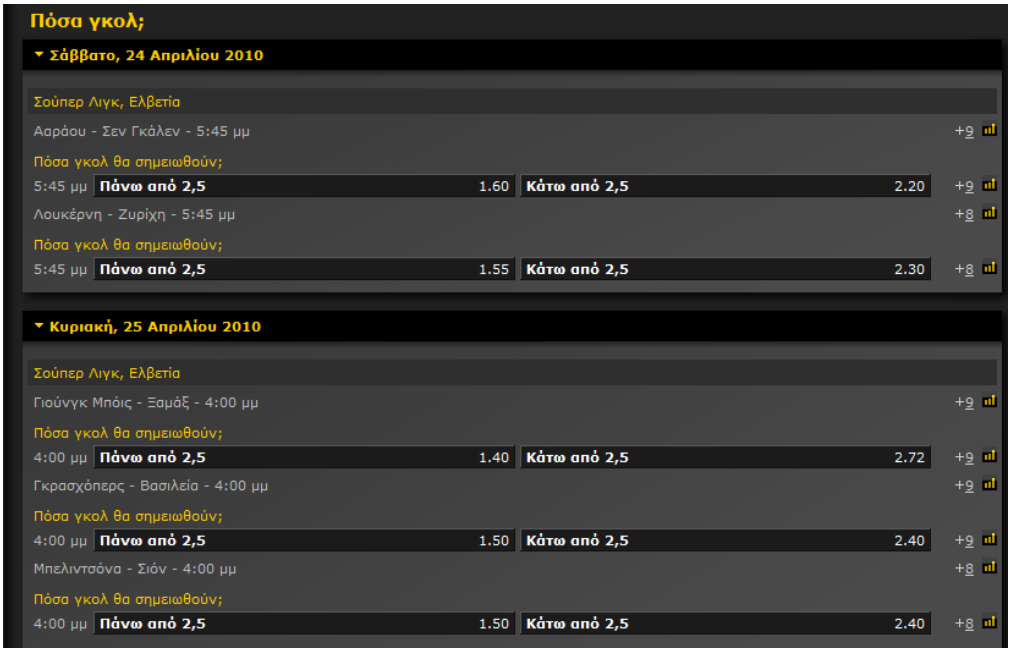
ENGLAND - CHAMPIONSHIP 2009/2010

| | | | HOME | | | | | AWAY | | | | | TOTAL | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P. | TEAM | PL | W | D | L | F | A | W | D | L | F | A | W | D | L | F | A | GD | +/- | Pts |
| 1 | NEWCASTLE | 46 | 18 | 4 | 0 | 54 | 11 | 11 | 7 | 4 | 33 | 22 | 29 | 11 | 4 | 87 | 33 | +54 | | 98 |
| 2 | WEST BROM | 46 | 16 | 2 | 4 | 47 | 20 | 10 | 9 | 3 | 40 | 26 | 26 | 11 | 7 | 87 | 46 | +41 | | 89 |
| 3 | NOTTM FOREST | 46 | 17 | 2 | 3 | 42 | 13 | 4 | 10 | 8 | 18 | 25 | 21 | 12 | 11 | 60 | 38 | +22 | | 75 |
| 4 | CARDIFF | 46 | 11 | 6 | 5 | 34 | 18 | 10 | 4 | 8 | 36 | 32 | 21 | 10 | 13 | 70 | 50 | +20 | | 73 |
| 5 | LEICESTER | 46 | 12 | 6 | 4 | 38 | 18 | 7 | 8 | 8 | 20 | 27 | 19 | 14 | 12 | 58 | 45 | +13 | | 70 |
| 6 | SWANSEA | 46 | 10 | 9 | 3 | 21 | 12 | 7 | 8 | 7 | 19 | 23 | 17 | 17 | 10 | 40 | 35 | +5 | | 68 |
| 7 | BLACKPOOL | 46 | 13 | 5 | 4 | 45 | 21 | 5 | 7 | 10 | 27 | 36 | 18 | 12 | 14 | 72 | 57 | +15 | | 66 |
| 8 | MIDDLESBROUGH | 44 | 9 | 7 | 6 | 24 | 20 | 7 | 6 | 9 | 33 | 27 | 16 | 13 | 15 | 57 | 47 | +10 | | 61 |
| 9 | READING | 43 | 9 | 7 | 6 | 35 | 21 | 7 | 4 | 10 | 27 | 36 | 16 | 11 | 16 | 62 | 57 | +5 | | 59 |
| 10 | SHEFFIELD UTD | 46 | 11 | 8 | 3 | 35 | 20 | 4 | 6 | 12 | 22 | 35 | 15 | 14 | 15 | 57 | 55 | +2 | | 59 |
| 11 | BRISTOL CITY | 44 | 9 | 10 | 3 | 36 | 33 | 5 | 7 | 10 | 17 | 30 | 14 | 17 | 13 | 53 | 63 | -10 | | 59 |
| 12 | DONCASTER | 44 | 8 | 7 | 7 | 28 | 26 | 6 | 7 | 9 | 27 | 29 | 14 | 14 | 16 | 55 | 55 | +0 | | 56 |
| 13 | IPSWICH | 44 | 8 | 11 | 3 | 24 | 20 | 4 | 8 | 10 | 24 | 36 | 12 | 19 | 13 | 48 | 56 | -8 | | 55 |
| 14 | PRESTON | 44 | 9 | 10 | 3 | 35 | 25 | 4 | 5 | 13 | 22 | 43 | 13 | 15 | 16 | 57 | 68 | -11 | | 54 |
| 15 | DERBY | 46 | 11 | 13 | 8 | 35 | 32 | 3 | 8 | 11 | 15 | 29 | 14 | 11 | 19 | 50 | 61 | -11 | | 53 |
| 16 | COVENTRY | 44 | 8 | 9 | 5 | 27 | 25 | 5 | 5 | 12 | 19 | 34 | 13 | 14 | 17 | 46 | 59 | -13 | | 53 |
| 17 | BARNSLEY | 44 | 8 | 7 | 7 | 25 | 28 | 6 | 4 | 12 | 27 | 39 | 14 | 11 | 19 | 52 | 67 | -15 | | 53 |
| 18 | Q.P.R | 43 | 7 | 9 | 5 | 35 | 27 | 5 | 6 | 11 | 21 | 37 | 12 | 15 | 16 | 56 | 64 | -8 | | 51 |
| 19 | SCUNTHORPE | 43 | 10 | 5 | 6 | 36 | 28 | 4 | 3 | 15 | 19 | 48 | 14 | 8 | 21 | 55 | 76 | -21 | | 50 |
| 20 | WATFORD | 43 | 9 | 6 | 7 | 33 | 26 | 3 | 6 | 12 | 22 | 41 | 12 | 12 | 19 | 54 | 67 | -13 | | 48 |
| 21 | CRYSTAL PALACE | 44 | 8 | 4 | 10 | 23 | 26 | 6 | 6 | 11 | 5 | 15 | 14 | 10 | 21 | 47 | 50 | -3 | -10 | 47 |
| 22 | SHEFFIELD WED | 44 | 8 | 5 | 9 | 28 | 29 | 3 | 8 | 11 | 17 | 35 | 11 | 13 | 20 | 45 | 64 | -19 | | 46 |
| 23 | PLYMOUTH | 44 | 5 | 6 | 11 | 19 | 28 | 6 | 2 | 14 | 23 | 35 | 11 | 8 | 25 | 42 | 63 | -21 | | 41 |
| 24 | PETERBOROUGH | 44 | 6 | 5 | 11 | 32 | 36 | 1 | 5 | 16 | 12 | 42 | 7 | 10 | 27 | 44 | 78 | -34 | | 31 |

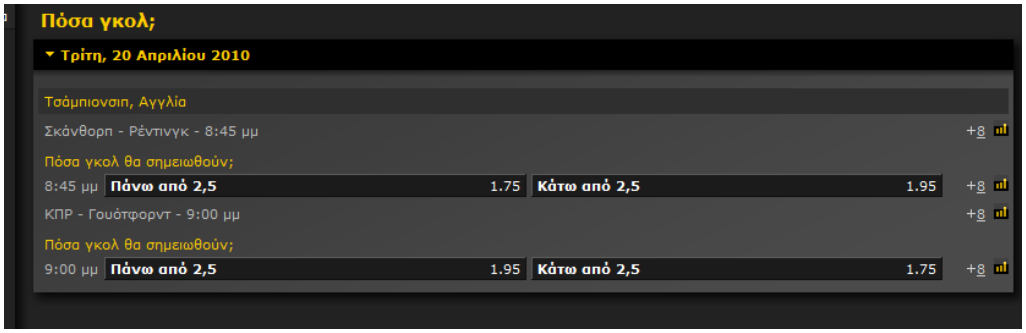
*Σχήμα 5.1Παράδειγμα βαθμολογίας στη Championship*

Αν λάβουμε υπόψη ότι οι πρώτες 6 αγωνιστικές κάθε περιόδου δεν μπορούν να χρησιμοποιηθούν ως δεδομένα εκπαίδευσης του δικτύου (μπορούμε να χρησιμοποιήσουμε παιχνίδια από την $7^{η}$ αγωνιστική και μετά λόγω του ότι κάποια από τα δεδομένα εισόδου στηρίζονται στο τι έγινε τα τελευταία 6 παιχνίδια), τότε απομένουν 480 αγώνες στους οποίους μπορεί να εκπαιδευτεί το δίκτυο. Ο αριθμός αυτός εξακολουθεί να είναι αρκετά μεγάλος και ικανοποιεί πλήρως το κριτήριο μας.

Ακόμη τα παιχνίδια της Championship προσφέρονται προς στοιχηματισμό από όλες τις μεγάλες εταιρίες στοιχημάτων στο διαδίκτυο και οι αποδόσεις είναι λογικές αφού δεν παρουσιάζονται "παράξενα" αποτελέσματα ή κοινές συμπεριφορές ομάδων που θα αναγκάσουν τις εταιρίες να μειώσουν τις αποδόσεις ή ακόμη να αποσύρουν τα στοιχήματα [9]. Για παράδειγμα στα πρωταθλήματα της Αυστρίας και της Ελβετίας οι ομάδες σκοράρουν με απίστευτους ρυθμούς. Αυτό έχει ως αποτέλεσμα οι αποδόσεις του over 2.5 να είναι πολύ χαμηλές και να κυμαίνονται από 1.30 μέχρι 1.65 στην καλύτερη περίπτωση. Πιο κάτω βλέπουμε μια τυπική αγωνιστική περίοδο του ελβετικού πρωταθλήματος. Στη συνέχεια βλέπουμε μια τυπική αγωνιστική του πρωταθλήματος της Championship.




*Σχήμα 5.2 : τυπική αγωνιστική περίοδος του ελβετικού πρωταθλήματος*

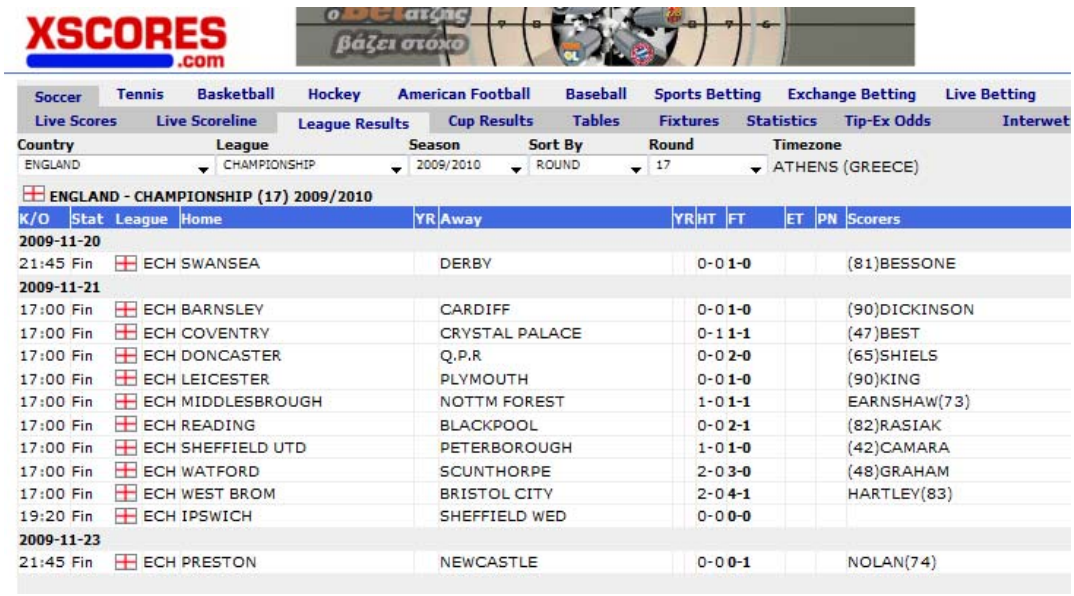



*Σχήμα 5.3 : τυπική αγωνιστική περίοδος του πρωταθλήματος της Championship*

### 5.3 Επεξεργασία δεδομένων

Η εκπαίδευση του δικτύου μου βασίστηκε σε δεδομένα του πρωταθλήματος της Championship διαφόρων αγωνιστικών περιόδων και συγκεκριμένα από το 2000 μέχρι το 2010. Δοκίμασα να χρησιμοποιήσω διαφορετικό αριθμό περιόδων για τα δίκτυα που εκπαίδευσα και τα καλύτερα αποτελέσματα προέκυψαν σε αυτά που εκπαιδεύτηκαν για 8 συνεχόμενες αγωνιστικές περιόδους.(2000/2001 – 2008/2009). Όσον αφορά τα δεδομένα ελέγχου αυτά είναι η αγωνιστική περίοδος 2009/2010, τόσο για το RBF network όσο και για το SVM.

Τα δεδομένα μου τα πήρα από την ιστοσελίδα http://www.xscores.com. Για να έχεις πρόσβαση στα αποτελέσματα που σε ενδιαφέρουν, αφού πατήσεις το σύνδεσμο που σε μεταφέρει στα ποδοσφαιρικά αποτελέσματα επιλέγεις το "league results" και στη συνέχεια συμπληρώνεις τις πληροφορίες του πρωταθλήματος που σε ενδιαφέρει.




*Σχήμα 5.4 : Παράδειγμα εξαγωγής αποτελεσμάτων από www.xscores.com*

Όπως βλέπουμε πιο πάνω παρέχονται όλες οι απαραίτητες πληροφορίες που χρειάζονται στη δική μας περίπτωση. Μπορούμε να δούμε τα παιχνίδια ταξινομημένα κατά αγωνιστική ή και με την χρονολογική σειρά που εξελίχθηκαν. Παρέχεται επίσης το αποτέλεσμα του αγώνα στο ημίχρονο, καθώς και οι ποδοσφαιριστές που πέτυχαν τα τέρματα των ομάδων τους.

Για την επεξεργασία των δεδομένων αυτών δημιούργησα την κλάση FileEditor.java η οποία δεχόταν ένα text αρχείο με αυτά τα δεδομένα (σε διαφορετική μορφή μετά από τροποποιήσεις που έκανα πάνω τους) και δημιουργούσε ένα file για κάθε ομάδα που λάμβανε μέρος στο πρωτάθλημα με τα αποτελέσματα της σε κάθε αγωνιστική από την αρχή μέχρι το τέλος. Στη συνέχεια το πρόγραμμα μέσω της MainClass.java διάβαζε τα αρχεία αυτά και δημιουργούσε ένα δυσδιάστατο πίνακα 46 x 12 στον οποίο περιέχονταν όλα τα αποτελέσματα όλων των αγωνιστικών στο πρωτάθλημα. Περισσότερα όμως για τον τρόπο που υλοποιήθηκε η σκέψη αυτή θα δούμε στο επόμενο κεφάλαιο.

MIDDLESBROUGH SHEFFIELDUTD 0 0
CARDIFF SCUNTHORPE 4 0
CRYSTALPALACE PLYMOUTH 1 1
DERBY PETERBOROUGH 2 1
LEICESTER SWANSEA 2 1
PRESTON BRISTOLCITY 2 2
Q.P.R BLACKPOOL 1 1
READING NOTTMFOREST 0 0
SHEFFIELDWED BARNSLEY 2 2
WATFORD DONCASTER 1 1
WESTBROM NEWCASTLE 1 1
COVENTRY IPSWICH 2 1
BARNSLEY COVENTRY 0 2
BLACKPOOL CARDIFF 1 1
BRISTOLCITY CRYSTALPALACE 1 0
DONCASTER PRESTON 1 1
IPSWICH LEICESTER 0 0
NOTTMFOREST WESTBROM 0 1
PETERBOROUGH SHEFFIELDWED 1 1
PLYMOUTH Q.P.R 1 1

*(Παράδειγμα του αρχείου tempSeason.txt από το οποίο πήρε τις πληροφορίες η κλάση FileEditor.java για την δημιουργία του αρχείου της SWANSEA όπως φαίνεται πιο κάτω.)*

LEICESTER SWANSEA 2 1
SWANSEA MIDDLESBROUGH 0 3
SWANSEA READING 0 0
COVENTRY SWANSEA 0 1
SWANSEA WATFORD 1 1
PRESTON SWANSEA 2 0
SWANSEA BRISTOLCITY 0 0
BARNSLEY SWANSEA 0 0
SWANSEA SHEFFIELDUTD 2 1
DONCASTER SWANSEA 0 0
SWANSEA Q.P.R 2 0
IPSWICH SWANSEA 1 1
WESTBROM SWANSEA 0 1
SWANSEA BLACKPOOL 0 0
SCUNTHORPE SWANSEA 0 2
SWANSEA CARDIFF 3 2
SWANSEA DERBY 1 0

*(Ένα μέρος του αρχείου της SWANSEA όπως προέκυψε μετά από την επεξεργασία του αρχείου tempSeason από την FileEditor.java)*

### 5.4 Κανονικοποίηση Τιμών

Όπως ανέφερα προηγουμένως (κεφ. 4.1) αρκετές από τις διαστάσεις του διανύσματος το οποίο δίνεται σαν είσοδος στο πρόγραμμα κατά την εκπαίδευση του αποτελούν ποσοστά. Κατ' επέκταση οι τιμές τους κυμαίνονται από 0 μέχρι 1. Κάποιες άλλες τιμές όμως, όπως για παράδειγμα ο μέσος όρος τερμάτων που επιτεύχθηκαν στα παιχνίδια που αγωνίστηκε συγκεκριμένη ομάδα μπορεί να είναι ένας αριθμός που ξεπερνάει κατά πολύ το 1. Αυτό είναι ένα φαινόμενο που θέλησα να αποφύγω, αφού τέτοιες μεγάλες διαφορές ανάμεσα στις τιμές των διαστάσεων του διανύσματος μπορούν να προκαλέσουν ανισορροπία στο δίκτυο και μη επιθυμητά αποτελέσματα. Έτσι χρησιμοποίησα την μέθοδο της κανονικοποίησης τιμών και την πιο κάτω εξίσωση :

$$y = \frac{X - x_{min}}{x_{max} - x_{min}}$$

*όπου*

X : η αρχική τιμή πριν την κανονικοποίηση

$x_{max}$ : η μεγαλύτερη πιθανή τιμή για την συγκεκριμένη μεταβλητή

$x_{min}$ : η μικρότερη πιθανή τιμή για την συγκεκριμένη μεταβλητή

$y$ : η νέα τιμή της μεταβλητής μετά την κανονικοποίηση

Έτσι με αυτό τον τρόπο όλες οι διαστάσεις του διανύσματος το οποίο δίνεται σαν είσοδος στο πρόγραμμα τόσο κατά την εκπαίδευση όσο και κατά την πρόβλεψη θα έχουν τιμές από 0 μέχρι 1 χωρίς να έχουν τις τεράστιες διαφορές μεταξύ τους που πιθανόν να δημιουργήσουν προβλήματα, αλλά επίσης χωρίς να χάνουν την σημασία τους και το νόημα τους.

# Κεφάλαιο 6

## Σχεδιασμός και υλοποίηση

6.1 Σχεδιασμός και υλοποίηση RBF Network……..……..... 39
6.1.1 Κλάση FileEditor.java.……..……..... 40
6.1.2 Κλάση Match.java.……..……..……..... 41
6.1.3 Κλάση Season.java.……..……..……..... 41
6.1.4 Κλάση Team.java ……..……..……..……..... 43
6.1.5 Κλάση HidenLayerNode.java.……..……..... 45
6.1.6 Κλάση OutputNode.java.……..……..... 47
6.1.7 Κλάση TrainingProcedure.java.……..……..……..... 48
6.1.8 Κλάση QuickLauncher.java.……..……..……..... 49
6.1.9 Δομή Δικτύου……..……..……..……..... 50
6.2 Υλοποίηση με τον SVM Classifier……..……..……..... 51

### 6.1 Σχεδιασμός και υλοποίηση RBF Network

Σε αυτό το υποκεφάλαιο θα αναφερθώ αναλυτικά στον τρόπο με τον οποίο σχεδιάστηκε και υλοποιήθηκε το Radial Basis Function νευρωνικό δίκτυο μου. Όπως ανέφερα και στον πρόλογο της εργασίας, επέλεξα η υλοποίηση του δικτύου να γίνει με την Java σαν γλώσσα προγραμματισμού, άρα ο σχεδιασμός έπρεπε να γίνει με βάση τους κανόνες του αντικειμενοστρεφούς προγραμματισμού. Πιο κάτω θα περιγράψω τις κλάσεις που χρησιμοποίησα, τις δομές δεδομένων που διαχειρίζεται η κάθε μια και πως αλληλεπιδρά με τις υπόλοιπες κλάσεις έτσι ώστε να συνθέτουν την σωστή λειτουργία του νευρωνικού δικτύου.

### 6.1.1 Κλάση FileEditor.java

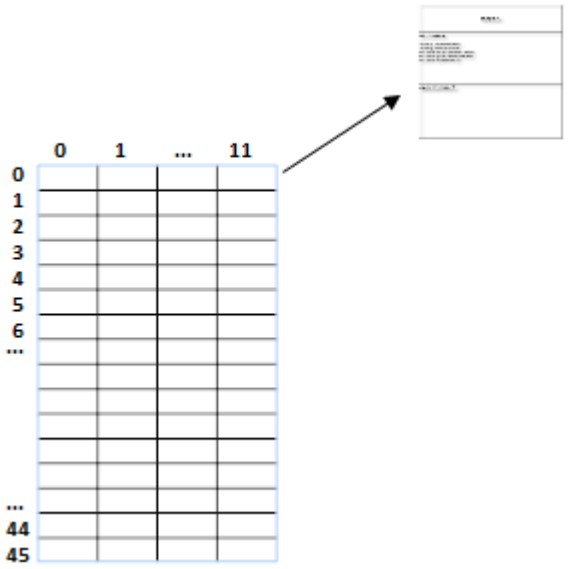
Η κλάση αυτή κατασκευάστηκε για την δημιουργία των αρχείων τα οποία περιέχουν τα αποτελέσματα της κάθε μιας από τις 24 ομάδες του πρωταθλήματος. Η κλάση αυτή διαβάζει ένα αρχείο το οποίο περιέχει τα αποτελέσματα όλων των αγωνιστικών της χρονιάς όπως αυτά εξάχθηκαν από το www.xscores.com. Ο χρήστης δίνει το όνομα της ομάδας για την οποία θέλει να δημιουργήσει αρχείο με τα αποτελέσματα της και η κλάση αυτή αντιγράφει όλα τα σχετικά παιχνίδια στα οποία συμμετείχε η ομάδα αυτή σε ένα αρχείο με το όνομα της το οποίο δημιουργεί. Έτσι έχουμε 24 αρχεία (ένα για κάθε ομάδα) στα οποία υπάρχουν συγκεντρωμένα τα αποτελέσματα της κάθε ομάδας στο πρωτάθλημα και με την σωστή χρονολογική σειρά με την οποία διεξάχθηκαν.

### 6.1.2 Κλάση Match.java

Η κλάση αυτή κατασκευάστηκε για να υλοποιεί αντικείμενα τύπου ποδοσφαιρικός αγώνας. Στα αντικείμενα τύπου match δηλαδή θα αποθηκεύονται όλες οι απαραίτητες πληροφορίες που χρειάζεται το δίκτυο μας για κάθε αγώνα. Οι πληροφορίες αυτές είναι το όνομα της γηπεδούχου ομάδας, το όνομα της φιλοξενούμενης ομάδας και το σκορ, πόσα τέρματα δηλαδή πέτυχε η γηπεδούχος και πόσα η φιλοξενούμενη ομάδα. Η κλάση αυτή δεν έχει μεθόδους αφού δεν έχει καθόλου συμπεριφορές. Είναι βοηθητική κλάση και χρησιμοποιείται σαν Data Struct από τις υπόλοιπες κλάσεις.


### 6.1.3 Κλάση Season.java

Η κλάση αυτή υλοποιεί αντικείμενα τύπου ποδοσφαιρική περίοδος. Έχει σαν πεδία ένα πίνακα από Teams (**LeaqueTable**) στον οποίο αποθηκεύονται όλες οι ομάδες που λαμβάνουν μέρος στο πρωτάθλημα την συγκεκριμένη περίοδο, τον αριθμό των αγωνιστικών του πρωταθλήματος, τον αριθμό των παιχνιδιών ανά αγωνιστική και την χρονολογία την οποία τελειώνει η περίοδος στην οποία αναφερόμαστε. Ακόμη περιέχει ένα πινάκα δύο διαστάσεων τύπου Match (**Games**) ο οποίος στη περίπτωση μας δημιουργείται με 46 γραμμές (όσες είναι οι αγωνιστικές) και 12 στήλες (όσοι είναι οι αγώνες ανά αγωνιστική). Όλες οι μέθοδοι της κλάσης αποσκοπούν στο να αρχικοποιήσουν τους δυο αυτούς πίνακες και να τους ενημερώνουν κατάλληλα κατά την λειτουργία του δικτύου.

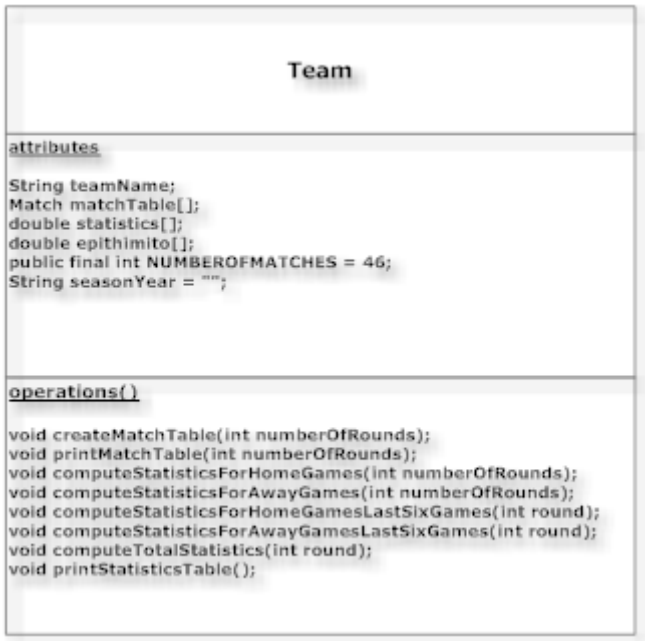




*Πίνακας* ***Games*** *(46 x 12) τύπου* ***Match*** *στον οποίο περιέχονται πληροφορίες για όλους τους αγώνες της χρονιάς.*

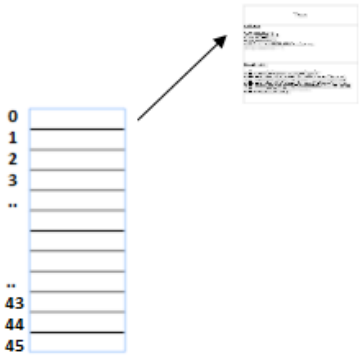

*Πίνακας* ***LeaqueTable*** *24 θέσεων τύπου* ***Team*** *στον οποίο αποθηκεύονται 24 αντικείμενα, ένα για κάθε μια από τις ομάδες που συμμετέχουν στο πρωτάθλημα.*

### 6.1.4 Κλάση Team.java

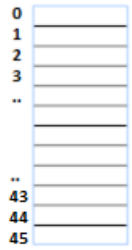
Η κλάση αυτή υλοποιεί αντικείμενα τύπου ομάδα. Τα χαρακτηριστικά της είναι το όνομα της ομάδας, ο αριθμός των παιχνιδιών που θα δώσει συνολικά σε όλη την διάρκεια της περιόδου και μια συμβολοσειρά που αντιπροσωπεύει την χρονολογία την οποία τελειώνει η περίοδος στην οποία αναφερόμαστε. Ακόμη, η κλάση περιέχει 3 πίνακες στους οποίους αποθηκεύονται οι πληροφορίες που χρειάζεται να ξέρει το δίκτυο μας για την κάθε ομάδα. Ο πρώτος είναι ένας πίνακας 46 θέσεων τύπου Match (**matchTable**) στον οποίο αποθηκεύονται τα 46 Match που θα δώσει η ομάδα καθ' όλη την διάρκεια της περιόδου. Ο δεύτερος είναι ένας πίνακας 23 θέσεων τύπου double (**statistics**) στον οποίο αποθηκεύονται τα στατιστικά κάθε ομάδας για την συγκεκριμένη αγωνιστική με τη σειρά που τα περιέγραψα στο κεφάλαιο 4.1. Ο πίνακας αυτός ανανεώνεται κάθε αγωνιστική. Ο τρίτος είναι ένας πίνακας 46 θέσεων τύπου double (**epithimito**) στον οποίο αποθηκεύονται τα επιθυμητά αποτελέσματα κάθε αγώνα της συγκεκριμένης ομάδας για τα αντίστοιχα παιχνίδια του πρώτου πίνακα και ο οποίος χρησιμοποιείται μόνο κατά την εκπαίδευση του δικτύου μας.



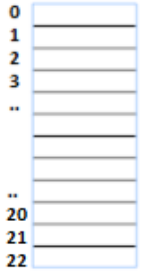

Επίσης όπως βλέπουμε η κλάση περιέχει μεθόδους που αποσκοπούν στο να δημιουργήσουν τους πίνακες αυτούς και να τους ενημερώνουν κατάλληλα κατά την λειτουργία του δικτύου. Ακόμη περιέχονται μέθοδοι οι οποίες τυπώνουν τις πληροφορίες αυτές στην οθόνη.




*Πίνακας* ***matchTable*** *(46 θέσεων) τύπου* ***Match*** *στον οποίο περιέχονται 46 αντικείμενα τύπου Match για την κάθε ομάδα.*



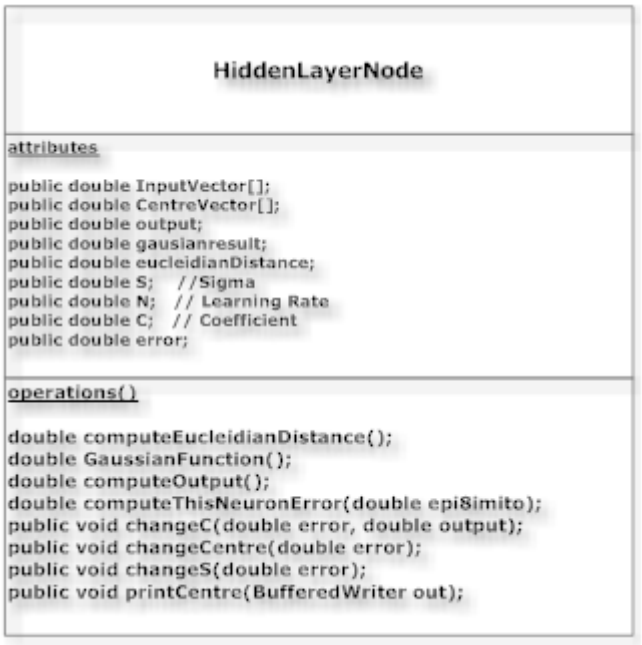

*Πίνακας* ***epithimito*** *(46 θέσεων) τύπου* ***double*** *στον οποίο περιέχονται τα 46 επιθυμητά αποτελέσματα που αντιστοιχούν στους αγώνες του πίνακα* ***matchTable.***




*Πίνακας* ***statistics*** *(24 θέσεων) τύπου* ***double*** *στον οποίο περιέχονται τα* στατιστικά κάθε ομάδας για την συγκεκριμένη αγωνιστική με τη σειρά που τα περιέγραψα στο κεφάλαιο **4.1**.

### 6.1.5 Κλάση HiddenLayerNode.java

Η κλάση αυτή υλοποιεί το Hidden Layer. Τα χαρακτηριστικά της είναι το input vector (vector 46 θέσεων τύπου double) στο οποίο *περιέχονται τα* στατιστικά των 2 ομάδων οι οποίες αγωνίζονται μεταξύ τους σε συγκεκριμένο αγώνα (23 θέσεις για την μία και 23 για την άλλη), το centre vector το οποίο είναι ένα αντίστοιχο vector για το συγκεκριμένο κέντρο, το output που παράγει ο συγκεκριμένος κρυφός νευρώνας, το αποτέλεσμα της γκαουσιανής συνάρτησης όπως υπολογίζεται στον συγκεκριμένο κρυφό νευρώνα, η ευκλείδεια απόσταση μεταξύ του input vector και του centre vector, η διασπορά του κρυφού νευρώνα, το βάρος και ο ρυθμός μάθησης του κρυφού νευρώνα καθώς και το error του κρυφού νευρώνα.




Επίσης οι μέθοδοι της κλάσης υπολογίζουν τα χαρακτηριστικά που περιέγραψα πιο πάνω δηλαδή την ευκλείδεια απόσταση, το αποτέλεσμα της Γκαουσιανής συνάρτησης, το output του νευρώνα και το error του νευρώνα. Επίσης υπάρχουν οι 3 συναρτήσεις οι οποίες αλλάζουν τα βάρη τις διασπορές και τα κέντρα κάθε νευρώνα προκαλώντας την εκπαίδευση του δικτύου.

Στη συνέχεια θα δούμε με ποιο τρόπο αλλάζουν οι συντελεστές αυτοί, ο κάθε ένας ξεχωριστά.

**Υπολογισμός Γκαουσιανής συνάρτησης:**

$$\varphi(x) = exp\left(\frac{-x^2}{\sigma^2}\right)$$

**Υπολογισμός output:**

$$y[k] = c[k] * exp\left(\frac{-x^2}{\sigma^2}\right)$$

**Υπολογισμός error:**

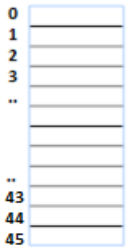
$$e[k] = d[k] - y[k]$$

**Αλλαγή Μεταβλητών:**

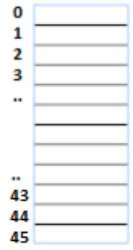
$$c[k+1] = c[k] + n_c * e[k] * \eta'[k]$$

$$R[k+1] = R_k[k] + n_R * \frac{e[k] * c_k[k]}{\sigma_k^2[k]} * \Phi\{x[k], R_k, \sigma_k\}\left[x[k] - R_k[k]\right]$$

$$\sigma[k+1] = \sigma_k[k] + n_\sigma * \frac{e[k] * c_k[k]}{\sigma_k^3[k]} * \Phi\{x[k], R_k, \sigma_k\} \|x[k] \quad R_k[k]\|^2$$



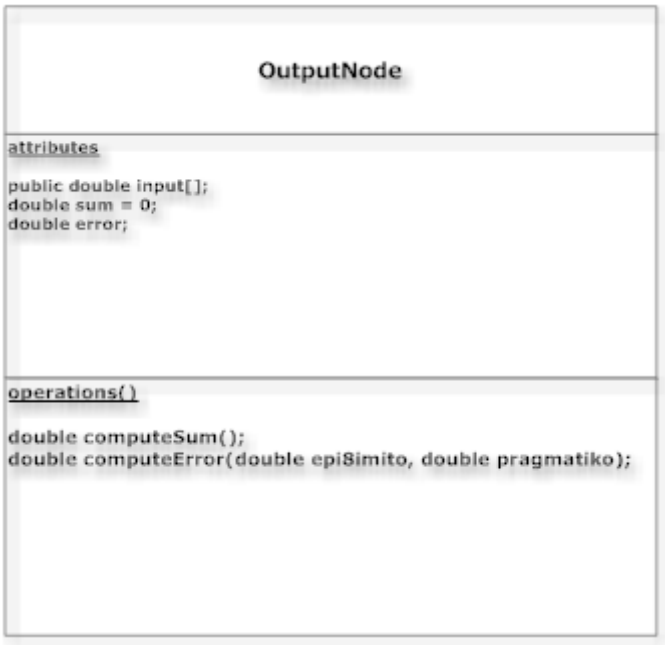

*Πίνακας* ***input vector*** *(46 θέσεων) τύπου* ***double*** *στον* οποίο *περιέχονται τα* στατιστικά των 2 ομάδων οι οποίες αγωνίζονται μεταξύ τους σε συγκεκριμένο αγώνα (23 θέσεις για την μία και 23 για την άλλη).




*Πίνακας* ***centre vector*** *(46 θέσεων) τύπου* ***double*** *στον* οποίο *περιέχεται το vector του κέντρου για τον συγκεκριμένο νευρώνα.*

### 6.1.6 Κλάση OutputNode.java

Η κλάση αυτή υλοποιεί το Output Layer. Έχει ως χαρακτηριστικά ένα πίνακα input στον οποίο έρχονται σαν είσοδος όλα τα αποτελέσματα των κρυφών νευρώνων. Ο πίνακας αυτός έχει μέγεθος ίσο με τον αριθμό των κρυφών νευρώνων στο δίκτυο μας. Ακόμη έχει το sum στο οποίο αποθηκεύεται το αποτέλεσμα που θα παράξει καθώς και το error στο οποίο αποθηκεύεται το λάθος που θα υπολογίσει.




Επίσης οι μέθοδοι της κλάσης υπολογίζουν το αποτέλεσμα του δικτύου καθώς επίσης και το λάθος αν το δίκτυο μας βρίσκεται σε κατάσταση εκπαίδευσης.

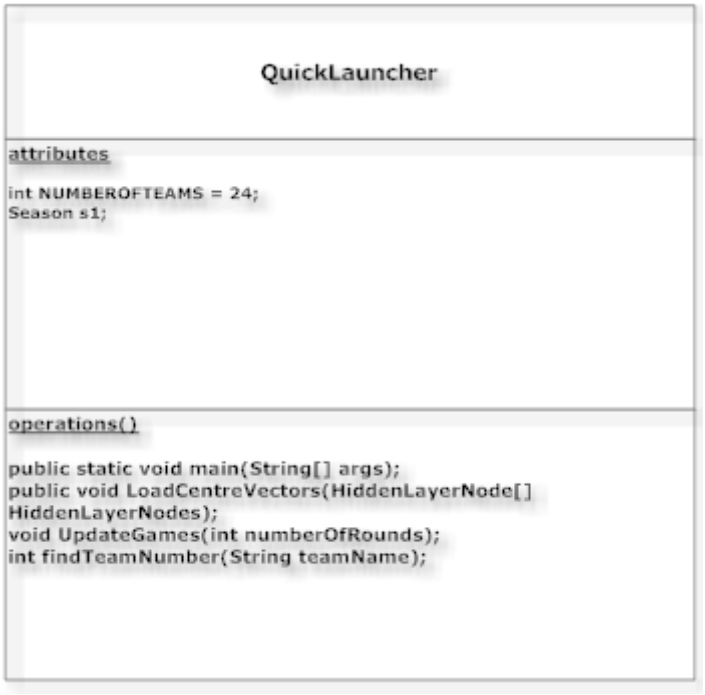
### 6.1.7 Κλάση TrainingProcedure.java

Η κλάση αυτή υλοποιεί την διαδικασία εκπαίδευσης του δικτύου. Εδώ θα δημιουργηθούν όλα τα αντικείμενα που χρειάζονται για την λειτουργία του νευρωνικού μας δικτύου όπως οι κρυφοί νευρώνες, ο νευρώνας εξόδου και οι πίνακες με τα στατιστικά και τα δεδομένα. Θα διαβαστούν τα αρχεία με τα δεδομένα που χρειάζεται το δίκτυο για να λειτουργήσει και αφού δημιουργηθούν οι κατάλληλες δομές, θα καλεστούν οι κατάλληλες μέθοδοι για να προσομοιωθεί η διαδικασία εκπαίδευσης ενός Radial Basis Function νευρωνικού δικτύου.

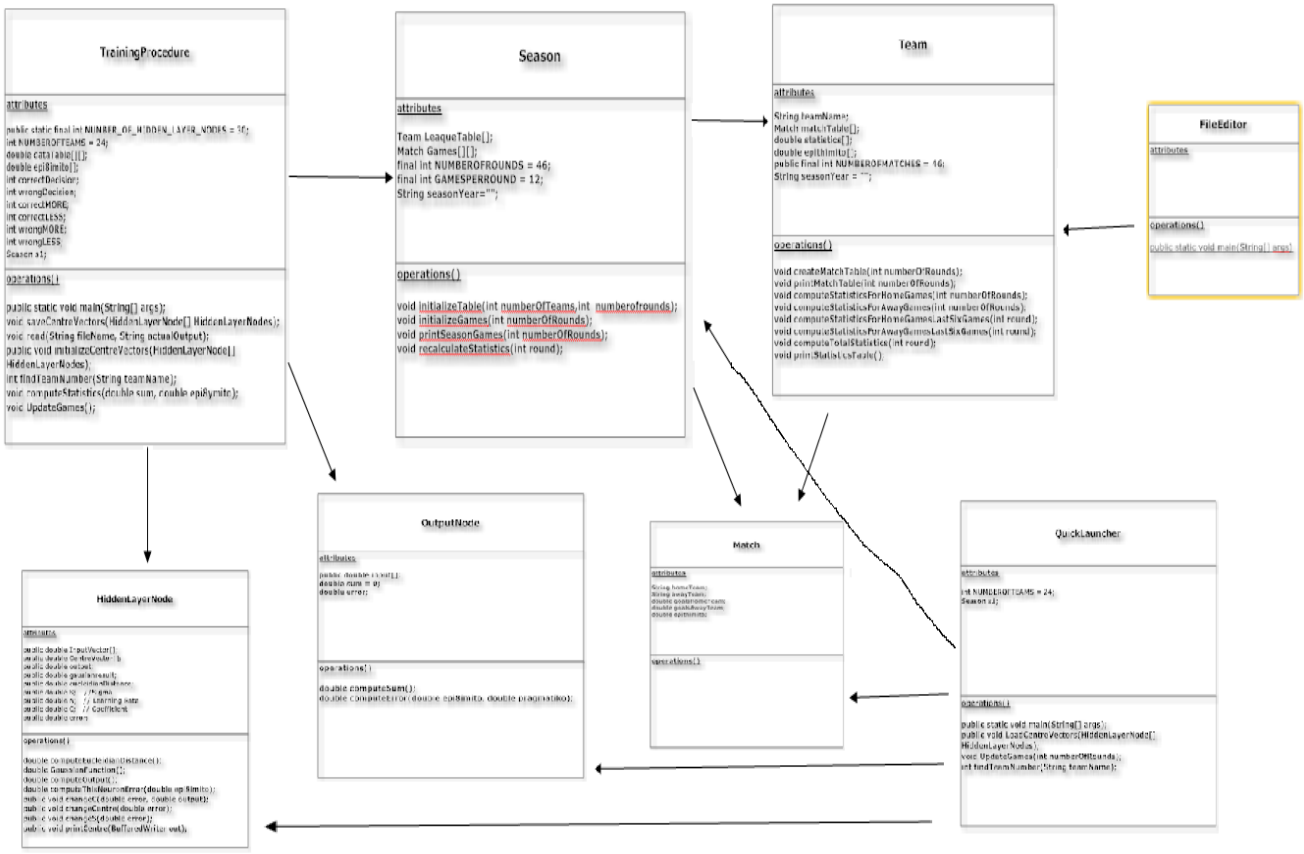

Οι μέθοδοι της κλάσης αυτής είναι υπεύθυνες για την δημιουργία και αρχικοποίηση των αντικειμένων που αποτελούν το νευρωνικό μου δίκτυο καθώς επίσης και για την αποθήκευση σε αρχεία της δομής του τελικού δικτύου όπως αυτή προκύπτει μετά από την ολοκλήρωση της εκπαίδευσης. Επίσης υπάρχουν βοηθητικές μέθοδοι που εκτελούν δευτερεύοντες λειτουργίες όπως ο υπολογισμός στατιστικών και η εμφάνιση των τρεχόντων δεδομένων συγκεκριμένης ομάδας.

### 6.1.8 Κλάση QuickLauncher.java

Η κλάση αυτή υλοποιεί την διαδικασία πρόβλεψης των ποδοσφαιρικών αποτελεσμάτων της επόμενης αγωνιστικής. Εδώ δημιουργείται ξανά το δίκτυο μας και αντιγράφει στους κρυφούς νευρώνες την δομή του εκπαιδευμένου δικτύου από το αρχείο NetworkStructure.txt το οποίο δημιούργησε η TrainingProcedure.java. Στη συνέχεια αφού διαβάσει τα αναβαθμισμένα αρχεία με τις νέες πληροφορίες του πρωταθλήματος και των ομάδων, υπολογίζει τα αποτελέσματα των αγώνων που δεν έχουν πραγματοποιηθεί ακόμη και δίνει τις προβλέψεις της στο αρχείο predictions.txt. Ο χρήστης το μόνο που πρέπει να κάνει είναι να εισάξει στην κλάση τον αριθμό της αγωνιστικής της οποίας θέλει να πάρει πρόβλεψη αποτελεσμάτων αφού φυσικά έχει ενημερώσει τα αρχεία των ομάδων για τα αποτελέσματα της προηγούμενης αγωνιστικής.



### 6.1.9 Δομή Δικτύου



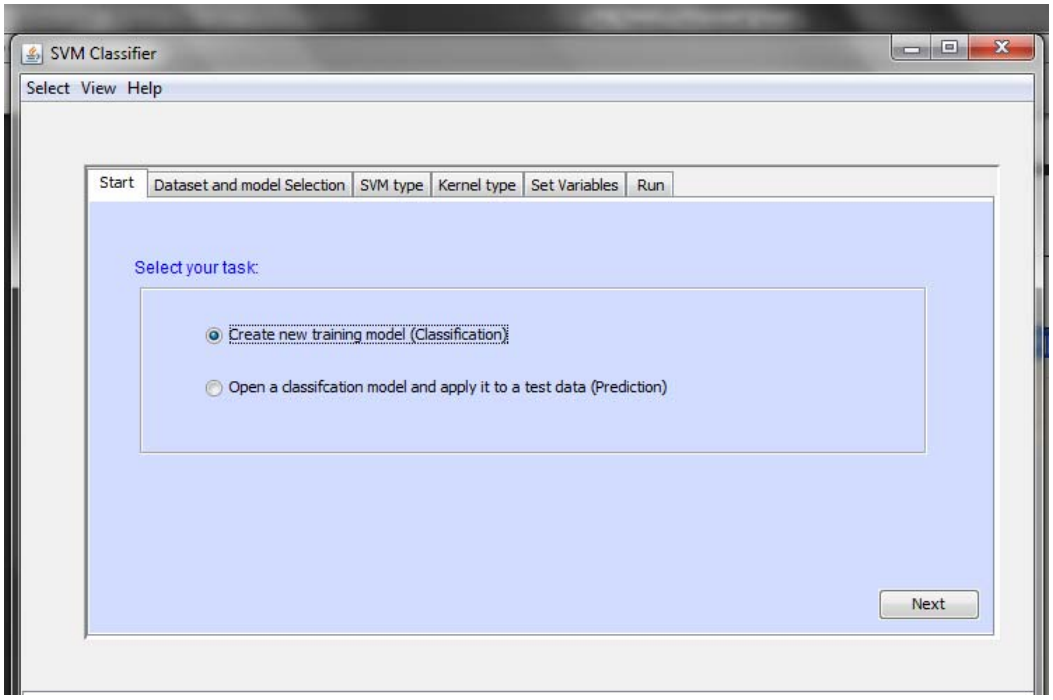

Πιο πάνω βλέπουμε την δομή του δικτύου μας, με ποίο τρόπο επικοινωνούν δηλαδή οι κλάσεις μας και ποιες χρησιμοποιούν ποιες για την σωστή λειτουργία του προγράμματος. Όπως παρατηρούμε, οι δυο κλάσεις που περιέχουν μέθοδο main και μπορούν να τρέξουν από μόνες τους είναι η TrainingProcedure και η QuickLauncher. Οι υπόλοιπες χρησιμοποιούνται σαν βοηθητικές κλάσεις αναπαριστώντας είτε στοιχεία του δικτύου μας ( HiddenLayerNode, OutputLayerNode), είτε του πρωταθλήματος μας(Season ,Match, Team).

## 6.2 Υλοποίηση με τον SVM Classifier

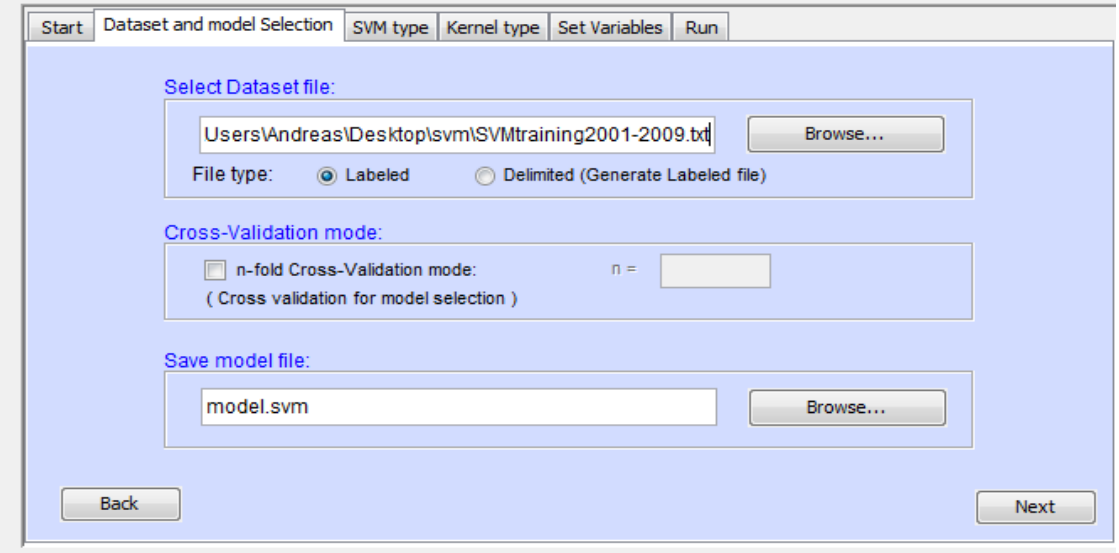
Όπως ανέφερα και στην εισαγωγή μου, για την εξαγωγή αποτελεσμάτων με την χρήση των Support Vector Machines θα χρησιμοποιηθεί ο προσομοιωτής SVM Classifer ο οποίος έχει εκδοθεί το Δεκέμβριο του 2006, από τους Mehdi Pirooznia και Yooping Deng στο University of Southern Mississippi. Ο προσομοιωτής αυτός είναι διαθέσιμος στο διαδίκτυο στην ιστοσελίδα http://mcbc.usm.edu/svm.

Ο SVM Classifer είναι ένα java interface το οποίο με τη χρήση support vector machines πετυχαίνει την κατηγοριοποίηση δεδομένων. Η χρήση του περιλαμβάνει δύο βασικές κατηγορίες, την εκπαίδευση του μέσω της προσπάθειας ταξινόμησης – κατηγοριοποίησης των δεδομένων εκπαίδευσης και την διαδικασία πρόβλεψης που επιχειρείται στα δεδομένα ελέγχου μέσω του μοντέλου το οποίο ήδη δημιουργήθηκε την πρώτη φάση.

Πιο κάτω θα προσπαθήσω να εξηγήσω περιληπτικά την χρήση του simulator αυτού και τι σημαίνει για την δική μας περίπτωση η κάθε επιλογή που σου δίνει η εφαρμογή.




Μόλις ανοίξει η εφαρμογή σου δίνει 2 επιλογές. Η μια αφορά την πρώτη φάση που αναφέραμε πιο πάνω, αυτή δηλαδή της εκπαίδευσης του μοντέλου που θα δημιουργήσουμε και η δεύτερη αφορά την δεύτερη φάση, αυτή της πρόβλεψης για τα δεδομένα ελέγχου. Για αρχή λοιπόν επιλέγουμε το πρώτο.




Στην αρχή ζητείται από τον χρήστη να επιλέξει το αρχείο με τα δεδομένα εκπαίδευσης. Ο προσομοιωτής αναγνωρίζει τα δεδομένα αυτά μόνο αν έχουν τροποποιηθεί ώστε να έχουν την μορφή μιας εκ των δύο που γνωρίζει. Οι δύο λοιπόν επιλογές του χρήστη είναι να τροποποιήσει τα δεδομένα εισόδου που θα δώσει στην εφαρμογή έτσι ώστε να έχουν την μορφή "labeled" είτε "delimited". Αυτό ισχύει επίσης και για τα δεδομένα ελέγχου [7].

Στην δική μας περίπτωση έχω δημιουργήσει μια μέθοδο σε java η οποία τροποποιεί τα δεδομένα που παράγει το πρόγραμμα μου έτσι ώστε να εμφανίζονται όπως τα ζητά ο προσομοιωτής σε μορφή labeled.

**Labeled μορφή**

**label index1 : value1 index2 : value2 κτλ…**

Το label είναι η επιθυμητή τιμή των δεδομένων εκπαίδευσης. Στη περίπτωση της κατηγοριοποίησης, είναι μια οποιαδήποτε ακέραια τιμή η οποία αναπαριστά μια κατηγορία. Το index είναι ένας αύξων αριθμός που αντιπροσωπεύει τον αριθμό του στοιχείου που περιγράφει το πρότυπο. Στην δική μας περίπτωση αντιπροσωπεύει τον αριθμό του κάθε ενός από τα 46 στοιχεία του διανύσματος εισόδου ενώ το value αντιπροσωπεύει την τιμή του στοιχείου αυτού. Πιο κάτω φαίνεται ένα παράδειγμα ενός προτύπου, ενός παιχνιδιού δηλαδή μιας συγκεκριμένης αγωνιστικής (από την 6η και μετά όπως και στο RBF). Το label του είναι -1 δηλαδή το επιθυμητό αποτέλεσμα είναι LESS (Η επιλογή μου αυτή ήταν τυχαία, -1 για τα LESS και 1 για τα MORE). Στη

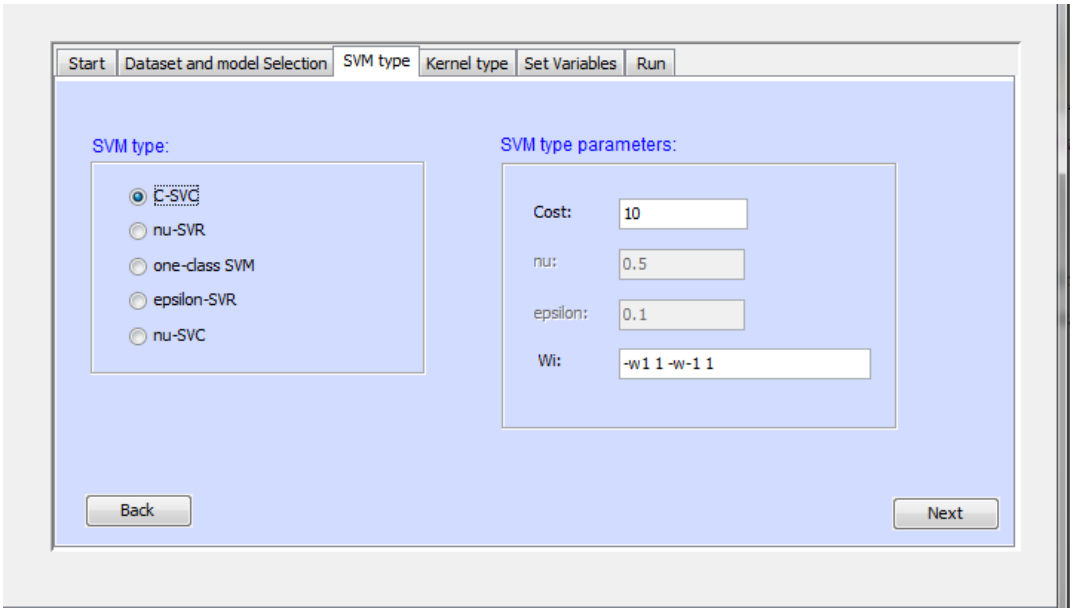
συνέχεια ακολουθούν οι 46 τιμές που καθορίζουν το διάνυσμα εισόδου του συγκεκριμένου παιχνιδιού.

```
-1    1:0.25     2:1.0      3:0.3333333333333333
      4:0.6666666666666666 5:0.2222222222222222
      6:0.6666666666666666 7:0.4166666666666667    8:1.0
      9:0.3333333333333333 10:0.3333333333333333   11:0.2
      12:1.0     13:0.25    14:0.6666666666666666
      15:0.16666666666666666     16:0.6666666666666666
      17:0.4166666666666667  18:1.0   19:0.3333333333333333
      20:0.3333333333333333  21:0.28571428571428575
      22:0.2857142857142857  23:0.3333333333333333    24:0.5
      25:1.0     26:0.0     27:0.0     28:0.08333333333333333
      29:0.25    30:0.5     31:0.75    32:0.0     33:0.5
      34:0.4     35:1.0     36:0.0     37:0.0     38:0.0625
      39:0.25    40:0.5     41:0.75    42:0.0     43:0.5
      44:0.28571428571428575     45:0.2857142857142857
      46:0.3333333333333333
```

Για κάθε συγκεκριμένο παιχνίδι θα δημιουργηθεί μια τέτοια ομάδα δεδομένων, 12 από αυτές δηλαδή σε κάθε αγωνιστική και 480 για μια ολόκληρη ποδοσφαιρική περίοδο. Με τις ομάδες αυτές δεδομένων θα εκπαιδευτεί ο προσομοιωτής.

Όσον αφορά τα delimited αρχεία, θεωρήστε ότι περιέχουν ακριβώς τις ίδιες πληροφορίες αλλά τα δεδομένα παρουσιάζονται διαφορετικά (κάθετα αντί οριζόντια). Στην περίπτωσή μας θα χρησιμοποιούμε πάντοτε labeled αρχεία.

Αφού λοιπόν επιλεγεί το αρχείο με τα δεδομένα εκπαίδευσης και το όνομα με το οποίο θέλουμε να αποθηκευτεί το μοντέλο που θα δημιουργηθεί, στην συνέχεια πρέπει να επιλέξουμε τον τύπο του SVM που θέλουμε να χρησιμοποιήσουμε καθώς και τις παραμέτρους του.




Οι επιλογές είναι: **C-SVC , nu-SVR , one-class SVM , epsilon-SVR , nu-SVC**

Στο δικό μας πρόβλημα που είναι πρόβλημα κατηγοριοποίησης, δεν μπορούν να χρησιμοποιηθούν οι **nu-SVR** και **epsilon-SVR** αφού οι τύποι αυτοί χρησιμοποιούνται για regression και όχι classification. Έτσι όπως θα δούμε αργότερα στο κεφάλαιο με τα αποτελέσματα με απασχόλησαν μόνο οι : **C-SVC** και **nu-SVC.**

## C-SVC: C-Support Vector Classification

Έστω ότι **{(xi, di)}i=1..n** είναι το δεδομένο εκπαίδευσης

**xi** είναι το πρότυπο εισόδου για το i-οστό παράδειγμα
και
**di** είναι το επιθυμητό αποτέλεσμα με τιμές {+1, -1}.

Με δεδομένο το σύνολο εκπαίδευσης **{(xi, di)} i = 1..N,** πρέπει να βρεθούν οι βέλτιστες τιμές του διανύσματος βαρών w και κατωφλίου b τέτοιες ώστε να ικανοποιούν τον περιορισμό

**di (wT xi + b) ≥ 1 - ξi για i = 1,2,..N**
**ξi ≥ 0** για όλα τα i

καθώς επίσης το διάνυσμα βαρών w και οι μεταβλητές ξi να ελαχιστοποιούν τη συνάρτηση κόστους **Φ(w, ξ) = ½ wT w + C Σ i=1..N ξi**
όπου **C** είναι μια θετική παράμετρος την οποία μπορεί να καθορίσει ο χρήστης. Στον προσομοιωτή μας η παράμετρος αυτή αναφέρεται σαν Cost.

Η συνάρτηση απόφασης είναι **sgn( Σ i=1..N di αi K(xi,x) + b).**

**nu-SVC: ν-Support Vector Classification**

Έστω ότι **{(xi, di)}i=1..n** είναι το δεδομένο εκπαίδευσης

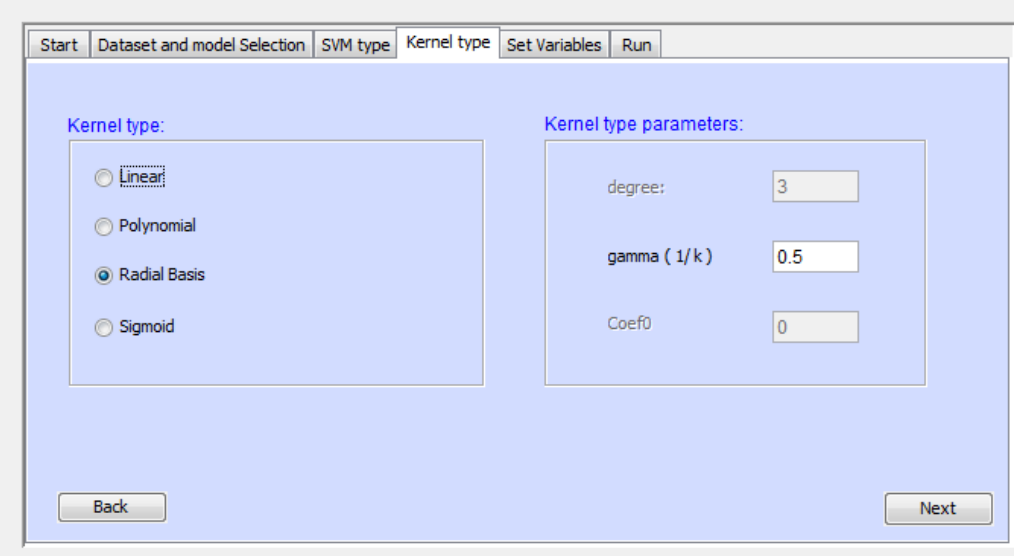
Η παράμετρος ν Є (0,1) είναι άνω όριο στο κλάσμα των λαθών εκπαίδευσης και κάτω όριο στο κλάσμα των support vectors.

Σκοπός είναι να βρεθούν οι βέλτιστες τιμές του διανύσματος βαρών **w** και κατωφλίου **b** οι οποίες να ικανοποιούν τον περιορισμό:

**$d_i$ ($w^T$ $x_i$ + b) ≥ ρ - $ξ_i$ για i = 1,2,..N**
**$ξ_i$ ≥ 0, ρ ≥ 0** για όλα τα **i**

καθώς επίσης το διάνυσμα βαρών **w** και οι μεταβλητές $ξ_i$ να ελαχιστοποιούν τη συνάρτηση κόστους $Φ(w, ξ) = ½ w^T w – θρ + (1/N) Σ_{i=1..N} ξ_i$

Η συνάρτηση απόφασης είναι η $sgn( Σ_{i=1..N} d_i αi K(x_i,x) + b)$.




Ακολούθως ο χρήστης επιλέγει τον τύπο του πυρήνα που θέλει να χρησιμοποιήσει. Οι τέσσερις επιλογές που δίνει ο προσομοιωτής είναι **ο linear**, **ο polynomial, ο radial basis και ο sigmoid.** Στην δική μας περίπτωση επέλεξα να χρησιμοποιήσω τον linear, τον Radial Basis και τον Sigmoid.

1. **Linear**: $K(x_i, x_j) = x_i^T x_j$

2. **Polynomial**: Ο πολυωνυμικός πυρήνας δύναμης d είναι της μορφής
$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$

3. **RBF**: Ο γκαουσιανός πυρήνας (Gaussian) επίσης γνωστός και ως radial basis function, είναι της μορφής

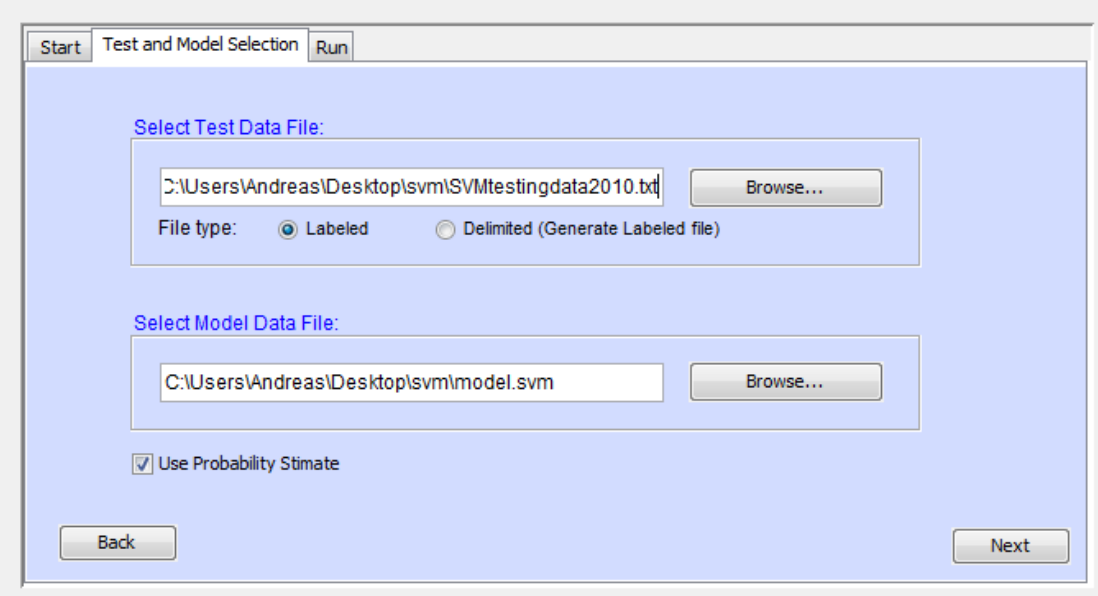
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0.$$

4. **Sigmoid**: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$
Όταν χρησιμοποιείται sigmoid kernel με το SVM μπορεί να θεωρηθεί ως two layer network.

Επίσης, επιλέγει και τροποποιεί αν θέλει τις παραμέτρους που έχει την δυνατότητα, ανάλογα με το kernel type που έχει επιλέξει.

Στη συνέχεια, επιλέγει κάποιες άλλες μεταβλητές του προσομοιωτή όπως για παράδειγμα το κριτήριο τερματισμού και τέλος επιλέγοντας run ξεκινάει η εκπαίδευση.

Αφού λοιπόν εκπαιδευτεί το μοντέλο του classifier στη συνέχεια πρέπει να επιστρέψουμε στην αρχή έτσι ώστε να πραγματοποιήσουμε πρόβλεψη, να δοκιμάσουμε δηλαδή το μοντέλο το οποίο ήδη έχουμε εκπαιδεύσει σε δεδομένα ελέγχου που δεν έχει συναντήσει μέχρι αυτή την στιγμή.




Επιλέγουμε λοιπόν την δεύτερη επιλογή στην αρχική φόρμα και εμφανίζεται η πιο πάνω στην οποία επιλέγουμε το αρχείο που περιέχει τα δεδομένα ελέγχου αλλά και το κατάλληλο μοντέλο με το οποίο θέλουμε να γίνει η κατηγοριοποίηση. Και σε αυτή την περίπτωση τα αρχεία με τα δεδομένα μου πρέπει να είναι σε μια εκ των δυο μορφών που αναγνωρίζονται από τον προσομοιωτή. Όπως στην προηγούμενη περίπτωση, έτσι και τώρα επέλεξα να χρησιμοποιήσω την μορφή labeled. Τα δεδομένα δημιουργήθηκαν και τροποποιήθηκαν έτσι ώστε να έχουν αυτή την μορφή από μια μέθοδο σε java την οποία έγραψα για αυτό το σκοπό.

Στη συνέχεια προχωρώντας στην επόμενη φόρμα και επιλέγοντας run θα αρχίσει η διαδικασία πρόβλεψης για τα δεδομένα ελέγχου. Αν όλα εξελιχθούν ομαλά, θα

δημιουργηθεί ένα αρχείο του οποίου έχουμε επιλέξει το όνομα, στο οποίο θα εμφανίζονται οι προβλέψεις για κάθε ένα από τα πρότυπα που υπήρχαν στο testing set. Επίσης μετά το πέρας της προσομοίωσης θα ξέρουμε τι ποσοστό των δεδομένων ελέγχου κατηγοριοποιήθηκε σωστά, αν φυσικά υπάρχει στο αρχείο δεδομένων το επιθυμητό αποτέλεσμα.

Παράδειγμα

Accuracy = 54.37100213219617% (255/469) (classification)

Mean squared error = 1.8251599147121536 (regression)

Squared correlation coefficient = 7.37041102786221E-4 (regression)

# Κεφάλαιο 7

## Αποτελέσματα και συζήτηση

7.1 Αποτελέσματα RBF Network……..……..... 59
7.2 Αποτελέσματα SVM Simulator……...……..... 84

### 7.1 Αποτελέσματα RBF Network

Σε αυτό το κεφάλαιο θα αναλύσουμε τα αποτελέσματα που προέκυψαν κατά την διάρκεια της εκπαίδευσης του δικτύου μας τα οποία ήταν πολλά και με εντελώς διαφορετικά ποσοστά επιτυχίας. Φυσικά δεν μπορούμε να αναφερθούμε σε όλα γιατί από τις δοκιμές που έγιναν τους μήνες που προηγήθηκαν, ο όγκος των αποτελεσμάτων τα οποία μπορούμε να σχολιάσουμε είναι απαγορευτικός. Έτσι επέλεξα ο ίδιος κάποιες ομάδες από αποτελέσματα, δεδομένα εκπαίδευσης, δομές και παραμέτρους δικτύων και αριθμό επαναλήψεων εκπαίδευσης από τα οποία πιστεύω ότι μπορούν να προκύψουν σημαντικά συμπεράσματα και να δώσουν τροφή προς συζήτηση.
Κατ' αρχάς τα πιο κάτω αποτελέσματα καταγράφηκαν ως αποτέλεσμα της χρήσης του Radial Basis Function Network που περιγράφηκε στο κεφάλαιο 5. Ως επί το πλείστον τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση του ήταν τα αποτελέσματα της Championship, της δεύτερης εν τη τάξη ποδοσφαιρικής κατηγορίας στην Αγγλία για τις αγωνιστικές περιόδους 2000-2001 μέχρι 2008-2009. Σε κάποιες περιπτώσεις επιχείρησα να μειώσω τον αριθμό των περιόδων αυτών με τα αποτελέσματα στην καλύτερη περίπτωση να παραμένουν στα ίδια επίπεδα όπως θα δούμε αναλυτικά στη συνέχεια. Όσον αφορά τα δεδομένα ελέγχου αυτά είναι η αγωνιστική περίοδος 2009/2010, τόσο για το RBF network όσο και για το SVM. Επίσης όσον αφορά την δομή του δικτύου μου τα περισσότερα αποτελέσματα πάρθηκαν με την χρήση RBF δικτύου με 30 ή 45 κρυφούς νευρώνες και διάνυσμα εισόδου 46 θέσεων (23 για την κάθε ομάδα). Από τα 26 τα στατιστικά τα οποία αναφέρονται στην περιγραφή μου στο προηγούμενο κεφάλαιο εξαιρούνται τα 3 που καθορίζουν την συμπεριφορά μιας ομάδας σε όλη την

διάρκεια της περιόδου (και όχι σε αγώνες οι οποίοι έγιναν στην έδρα της ή έκτος έδρας) αφού είναι ο ίδιος αριθμός (πχ το 21-22). Έτσι αυτά υπολογίζονται μία φορά γιατί αλλιώς θα υπολόγιζα το ίδιο πράγμα 2 φορές. Στην περιγραφή αναφέρεται για να δείξω ότι το κάνω και για τον γηπεδούχο και για τον φιλοξενούμενο. Έτσι απομένουν 23 στατιστικά.

Και σε αυτή την περίπτωση, δοκίμασα να μεταβάλω τις παραμέτρους αυτές καθώς και τις υπόλοιπες που καθορίζουν την λεπτομέρεια που κάνει την διαφορά σε ένα νευρωνικό δίκτυο όπως η διασπορά, τα βάρη και ο ρυθμός μάθησης.

Ο ρυθμός μάθησης είναι ο ίδιος για την αλλαγή και των τριών αυτών παραμέτρων.

Θα πρέπει επίσης να αναφέρω ότι από τα αποτελέσματα που θα σας παρουσιαστούν εξαιρούνται οι έξι πρώτοι αγώνες κάθε αγωνιστικής περιόδου, αφού όπως εξήγησα και σε προηγούμενο κεφάλαιο κάθε χρονιά πρέπει να δημιουργηθεί ένα απαραίτητο αντιπροσωπευτικό δείγμα για το πώς συμπεριφέρεται η κάθε ομάδα την τρέχουσα περίοδο. Το ελάχιστο αυτό δείγμα αποφασίσαμε ότι πρέπει να προκύπτει από έξι τουλάχιστο παιχνίδια.

Ακόμη, θα σημειώσω ότι ο έλεγχος θα γίνει τόσο με δεδομένα εκπαίδευσης αλλά και με δεδομένα ελέγχου, δηλαδή για μια αγωνιστική περίοδο την οποία δεν γνώρισε το δίκτυο αφού δεν χρησιμοποιήθηκαν τα αποτελέσματα της κατά την εκπαίδευση του.

**<u>Σύνολο Δεδομένων 1</u>**

**Περίοδοι εκπαίδευσης : 2004/2005 – 2008/2009**

**Αριθμός κρυφών νευρώνων: 30**

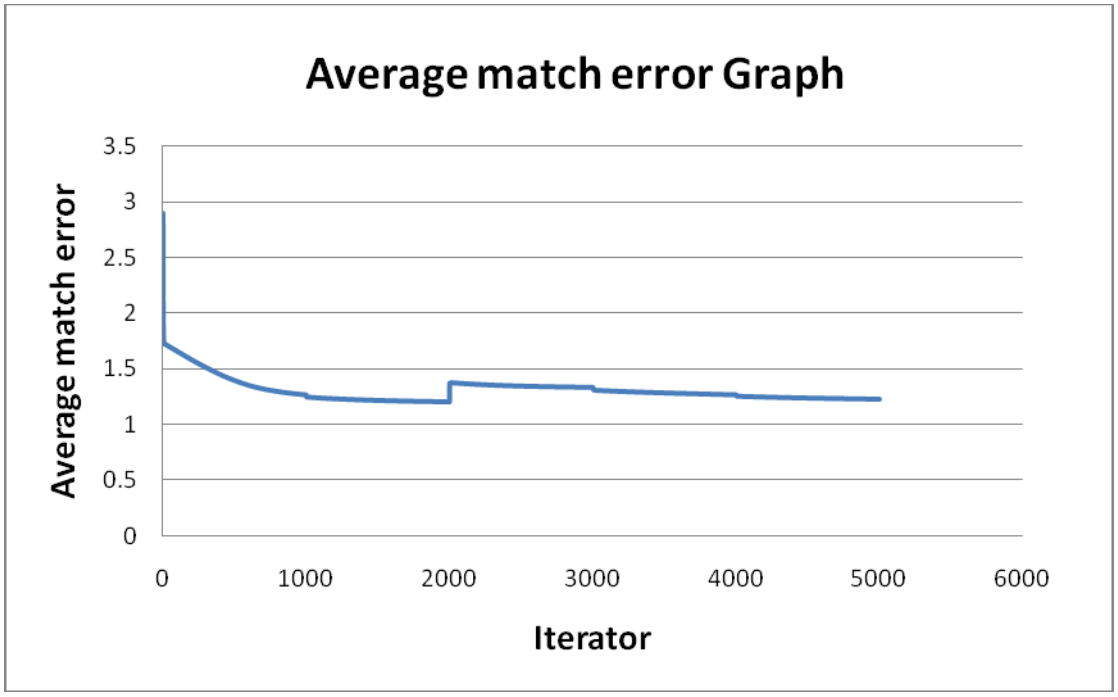
**Διάνυσμα εισόδου : 46 στοιχεία**

**Ρυθμός μάθησης : 0.0002**

**Αριθμός εποχών: 1000 x 5 = 5000**

Πιο κάτω βλέπουμε την γραφική παράσταση του λάθους κατά τις περιόδους εκπαίδευσης του δικτύου. Στον άξονα x είναι οι 5000 επαναλήψεις που έτρεξε το δίκτυο κατά την διάρκεια της εκπαίδευσης του (1000 εποχές για κάθε χρονιά). Στον άξονα y έχουμε το μέσο όρο του λάθους σε κάθε παιχνίδι στη συγκεκριμένη επανάληψη. Δηλαδή σε κάθε παιχνίδι που προβλέπεται αποτέλεσμα, υπολογίζεται το λάθος και προστίθεται (η απόλυτη τιμή του). Στο τέλος της επανάληψης ο αριθμός αυτός διαιρείται με 480 που είναι ο αριθμός των παιχνιδιών στα οποία εκπαιδεύεται το

δίκτυο για κάθε αγωνιστική περίοδο και προκύπτει ο μέσος όρος του λάθους κάθε παιχνιδιού για κάθε επανάληψη.




**correct decisions : 266**

**wrong decisions : 214**

**correct MORE : 105**

**correct LESS : 161**

**wrong MORE : 100**

**wrong LESS : 114**

Όπως βλέπουμε, σε σύνολο 480 αγώνων έχουμε 266 επιτυχίες, δηλαδή ποσοστό επιτυχίας **55,41%.**

**<u>Σύνολο Δεδομένων 2</u>**

**Περίοδοι εκπαίδευσης : 2004/2005 – 2008/2009**

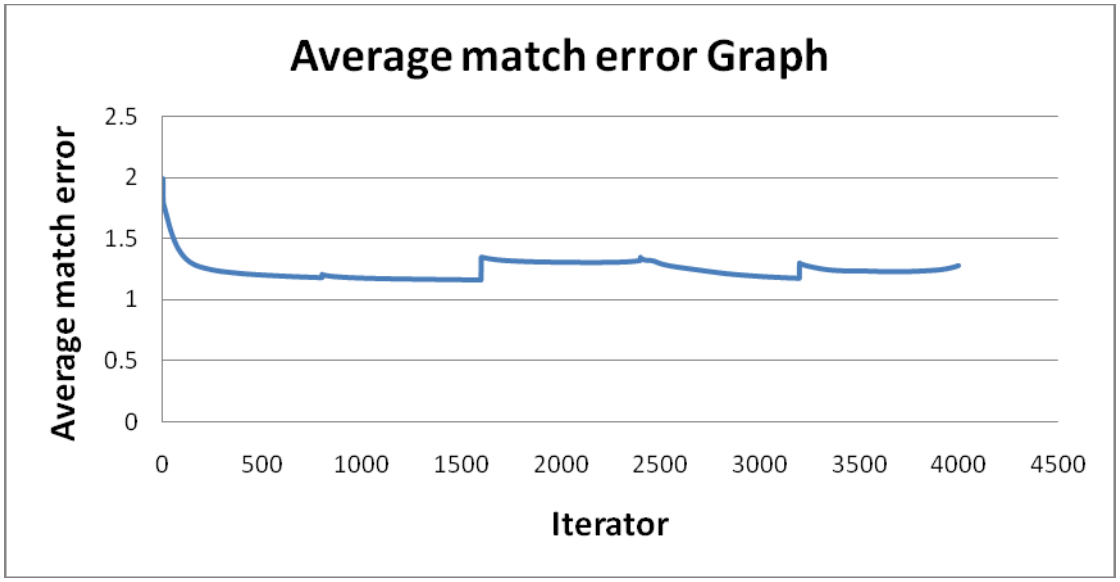
**Αριθμός κρυφών νευρώνων: 30**

**Διάνυσμα εισόδου : 46 στοιχεία**

**Ρυθμός μάθησης : 0.008**

**Αριθμός εποχών : 800 x 5 = 4000**

Σε αυτό το σύνολο δεδομένων έχουμε κάπως αυξημένο ρυθμό μάθησης και λίγο μειωμένο τον αριθμό των εποχών για κάθε περίοδο.




**correct decisions :269**

**wrong decisions :211**

**correct MORE :113**

**correct LESS :156**

**wrong MORE :105**

**wrong LESS :106**

Δοκιμές που έγιναν με μεγαλύτερο ρυθμό μάθησης δεν απέφεραν καλά αποτελέσματα. Μετά από προσεκτική μελέτη της τροπής που έπαιρνε η εκπαίδευση ανακάλυψα ότι το δίκτυο "χαλούσε" εντελώς σε περιπτώσεις που προέκυπτε κάποιο "παράξενο" αποτέλεσμα. Για παράδειγμα σε συγκεκριμένο παιχνίδι το αποτέλεσμα ήταν 6-5 υπέρ της γηπεδούχου ομάδας. Μόλις το δίκτυο συναντούσε το συγκεκριμένο παιχνίδι και λόγω του ψηλού ρυθμού μάθησης, οι παράμετροι του μεταβάλλονταν σε σημείο που το καθιστούσαν ανήμπορο να επανέλθει στα φυσιολογικά επίπεδα. Αυτό συνέβαινε γιατί μεγάλος ρυθμό μάθησης ισοδυναμεί με απότομες μεταβολές κάτι που σε ένα τόσο ευαίσθητο δίκτυο σαν το δικό μας δεν θέλουμε. Εδώ, σε σύνολο 480 αγώνων έχουμε 269 επιτυχίες, δηλαδή ποσοστό επιτυχίας **56,04%.**

**<u>Σύνολο Δεδομένων 3</u>**

**Περίοδοι εκπαίδευσης : 2004/2005 – 2008/2009**

**Αριθμός κρυφών νευρώνων: 30**

**Διάνυσμα εισόδου : 30 στοιχεία**

**Ρυθμός μάθησης : 0.0005**

**Αριθμός εποχών : 900 x 5 = 4500**

Σε αυτό το σύνολο δεδομένων έχουμε μειωμένο μέγεθος διανύσματος εισόδου, δηλαδή για κάθε παιχνίδι χρησιμοποιούσε πιο λίγες πληροφορίες σχετικά με τις ομάδες που αγωνίζονταν. Τα αποτελέσματα δεν ήταν καθόλου ικανοποιητικά και αποφάσισα να συνεχίσω με το διάνυσμα των 46 στοιχείων που αποδείχτηκε ότι εκπλήρωνε τις προσδοκίες μου καλύτερα από κάθε άλλο.

**<u>Σύνολο Δεδομένων 4</u>**

**Περίοδοι εκπαίδευσης : 2004/2005 – 2008/2009**

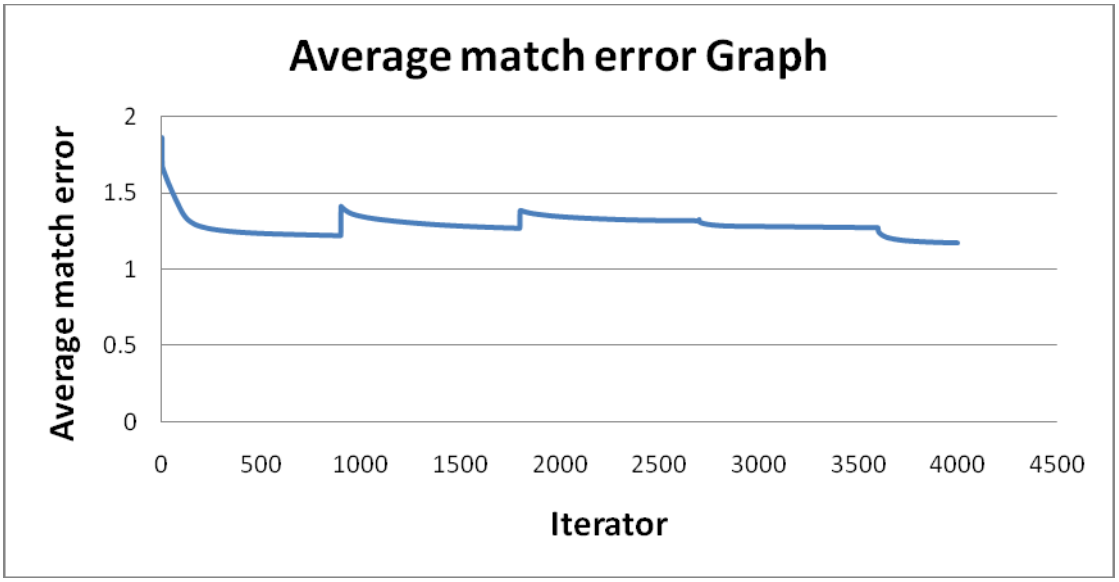
**Αριθμός κρυφών νευρώνων: 45**

**Διάνυσμα εισόδου : 46 στοιχεία**

**Ρυθμός μάθησης : 0.002**

**Αριθμός εποχών : 900 x 5 = 4500**

Σε αυτό το σύνολο δεδομένων αύξησα τον αριθμό των κρυφών νευρώνων σε 45.



**correct decisions :295**

**wrong decisions :185**

Όπως βλέπουμε, σε σύνολο 480 αγώνων έχουμε 295 επιτυχίες, δηλαδή ποσοστό επιτυχίας **61,45%.** Το average error σε αυτή την περίπτωση έφτασε το **1,17** ανά αγώνα.

**<u>Σύνολο Δεδομένων 5</u>**

**Περίοδοι εκπαίδευσης : 2000/2001 – 2008/2009**

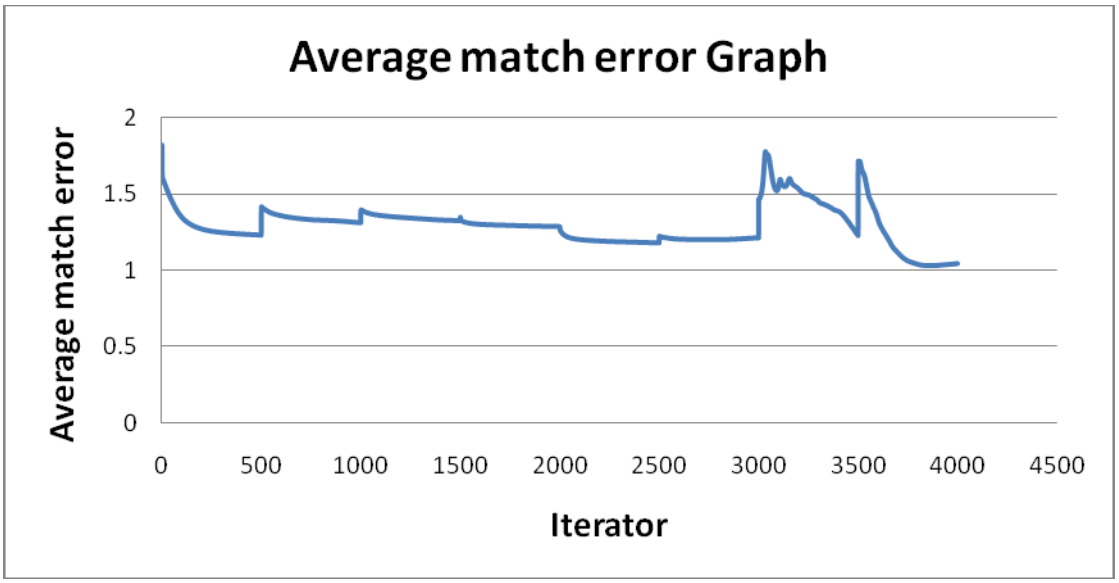
**Αριθμός κρυφών νευρώνων: 45**

**Διάνυσμα εισόδου : 46 στοιχεία**

**Ρυθμός μάθησης : 0.002**

**Αριθμός εποχών : 500 x 9 = 4500**

Σε αυτό το σύνολο δεδομένων αύξησα τον αριθμό των περιόδων εκπαίδευσης σε 9 αρχίζοντας από την περίοδο 2000/2001 μέχρι την περίοδο 2008/2009.
Σε αυτή την περίπτωση περίμενα ότι τα αποτελέσματα δεν θα ήταν αρκετά ικανοποιητικά αφού από το 2000 μέχρι το 2009 το ποδόσφαιρο άλλαξε αρκετά, το ίδιο και το πρωτάθλημα της Championship το οποίο μας ενδιαφέρει. Πίστευα ότι το δίκτυο μας θα προσπαθούσε να "μάθει" συμπεριφορές ομάδων που στην ουσία αγωνίζονταν σε διαφορετικό πρωτάθλημα, αφού η πάροδος τόσων πολλών χρόνων και η εξέλιξη του ποδοσφαίρου όλο αυτό το διάστημα προκάλεσαν αυτή την αλλαγή. Όπως φαίνεται από την γραφική παράσταση όμως και από το ποσοστό των σωστών προβλέψεων τα αποτελέσματα ήταν πολύ καλύτερα από αυτά που ανέμενα.



**correct decisions :315**

**wrong decisions :165**

**correct MORE :126**

**correct LESS :189**

**wrong MORE :72**

**wrong LESS :93**

Όπως βλέπουμε, σε σύνολο 480 αγώνων έχουμε 315 επιτυχίες, δηλαδή ποσοστό επιτυχίας **65,62%.** Το average error σε αυτή την περίπτωση έφτασε το 1,08 ανά αγώνα. Η γραφική παράσταση μετά την 3000η εποχή κινείται κάπως περίεργα αφού το error ανεβαίνει απότομα. Αυτό συμβαίνει γιατί εκεί αλλάζει αγωνιστική περίοδος και οι ομάδες συμπεριφέρονται διαφορετικά. Έτσι χρειάζεται κάποιος αριθμός εποχών για να προσαρμοστεί το δίκτυο μας και στη συνέχεια συμβαίνει το ίδιο με την επόμενη περίοδο. Αυτό είναι αναπόφευκτο και έχει να κάνει με τα δεδομένα μας. Κατά συνέπεια δεν μπορούμε να κάνουμε κάτι γι' αυτό παρά μόνο να το αναφέρουμε. Η περίοδος αυτή είναι η χρονιά 2006/2007. Αν προσέξουμε, και στις άλλες γραφικές παραστάσεις όταν άρχιζε η συγκεκριμένη περίοδος το error αυξανόταν προς στιγμής σε μικρότερο φυσικά βαθμό.

**<u>Σύνολο Δεδομένων 6</u>**

**Περίοδοι εκπαίδευσης : 2000/2001 – 2008/2009**

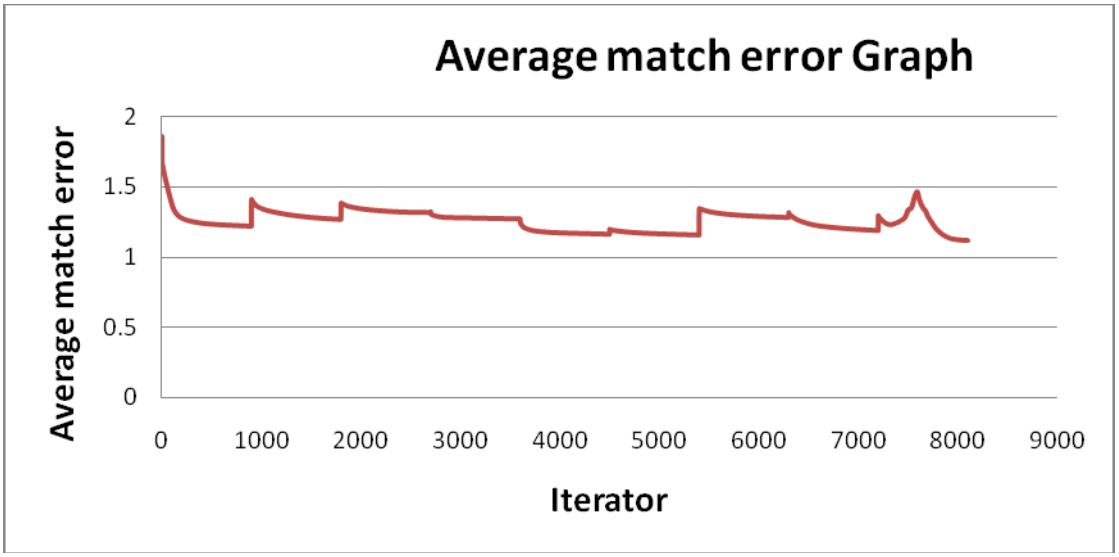
**Αριθμός κρυφών νευρώνων: 45**

**Διάνυσμα εισόδου : 46 στοιχεία**

**Ρυθμός μάθησης : 0.002**

**Αριθμός εποχών : 900 x 9 = 8100**

Σε αυτό το σύνολο δεδομένων διατήρησα τον αριθμό των περιόδων εκπαίδευσης σε 9 αρχίζοντας από την περίοδο 2000/2001 μέχρι την περίοδο 2008/2009 αλλά αύξησα των αριθμό των εποχών ανά περίοδο. Τα αποτελέσματα ήταν ακόμη καλύτερα.




**correct decisions :321**

**wrong decisions :159**

**correct MORE :124**

**correct LESS :197**

**wrong MORE :64**

**wrong LESS :95**

Σε σύνολο 480 αγώνων έχουμε 321 επιτυχίες, δηλαδή ποσοστό επιτυχίας **66,87%.** Το average error σε αυτή την περίπτωση έφτασε το **1,12** ανά αγώνα.

**<u>Σύνολο Δεδομένων 7</u>**

**Περίοδοι εκπαίδευσης : 2000/2001 – 2008/2009**

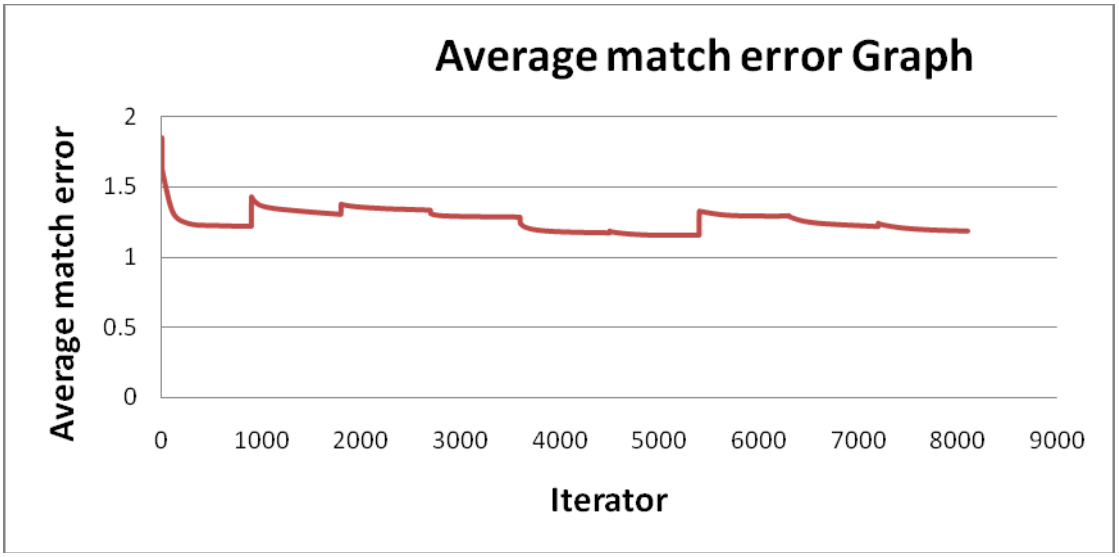
**Αριθμός κρυφών νευρώνων: 60**

**Διάνυσμα εισόδου : 46 στοιχεία**

**Ρυθμός μάθησης : 0.002**

**Αριθμός εποχών : 900 x 9 = 8100**

Σε αυτό το σύνολο δεδομένων διατήρησα τον αριθμό των περιόδων εκπαίδευσης σε 9 αρχίζοντας από την περίοδο 2000/2001 μέχρι την περίοδο 2008/2009 αλλά και τον αριθμό των εποχών ανά περίοδο. Αυτό που άλλαξα ήταν να αυξήσω τον αριθμό των κρυφών νευρώνων ακόμη περισσότερο.




**correct decisions : 290**

**wrong decisions : 190**

**correct MORE : 103**

**correct LESS : 187**

**wrong MORE : 74**

**wrong LESS : 116**

Αυτή την φορά τα αποτελέσματα δεν ήταν τόσο καλά. Σε σύνολο 480 αγώνων είχαμε 290 σωστές προβλέψεις δηλαδή ποσοστό επιτυχίας **60,41%,** ενώ το average error σε αυτή την περίπτωση έφτασε το **1,18** ανά αγώνα.

**<u>Σύνολο Δεδομένων 8</u>**

**Περίοδοι εκπαίδευσης : 2000/2001 – 2008/2009**

**Αριθμός κρυφών νευρώνων: 60**

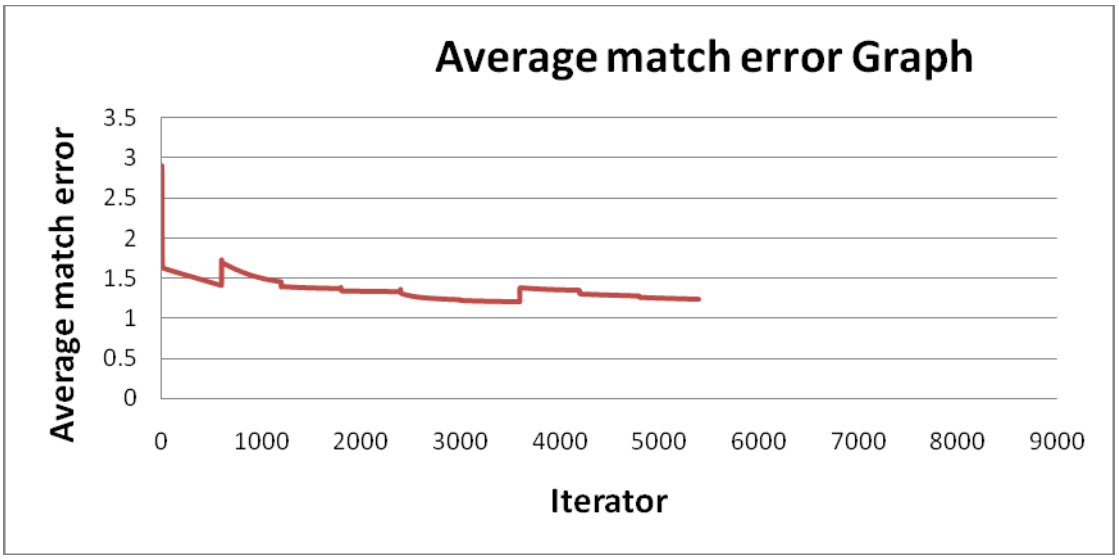
**Διάνυσμα εισόδου : 46 στοιχεία**

**Ρυθμός μάθησης : 0.0002**

**Αριθμός εποχών : 600 x 9 = 5400**

Σε αυτό το σύνολο δεδομένων διατήρησα τον αριθμό των περιόδων εκπαίδευσης σε 9 αρχίζοντας από την περίοδο 2000/2001 μέχρι την περίοδο 2008/2009 αλλά και τον αριθμό των κρυφών νευρώνων. Αυτό που άλλαξα ήταν να χρησιμοποιήσω πολύ μικρό ρυθμό μάθησης και μικρότερο αριθμό εποχών αφού όσο αυξάνεται η πολυπλοκότητα

του δικτύου (περισσότεροι κρυφοί νευρώνες), ο χρόνος που χρειάζεται το δίκτυο μας για να εκπαιδευτεί πολλαπλασιάζεται.




**correct decisions :266**

**wrong decisions :214**

**correct MORE :102**

**correct LESS :164**

**wrong MORE :97**

**wrong LESS :117**

Αυτή την φορά τα αποτελέσματα δεν ήταν καθόλου καλά. Είχαμε 290 σωστές προβλέψεις δηλαδή ποσοστό επιτυχίας **60,41%** ενώ το average error κατάφερε να μειωθεί μέχρι το **1,24** ανά αγώνα.

**<u>Σύνολο Δεδομένων 9 – Καλύτερα αποτελέσματα</u>**

**Περίοδοι εκπαίδευσης : 2000/2001 – 2008/2009**

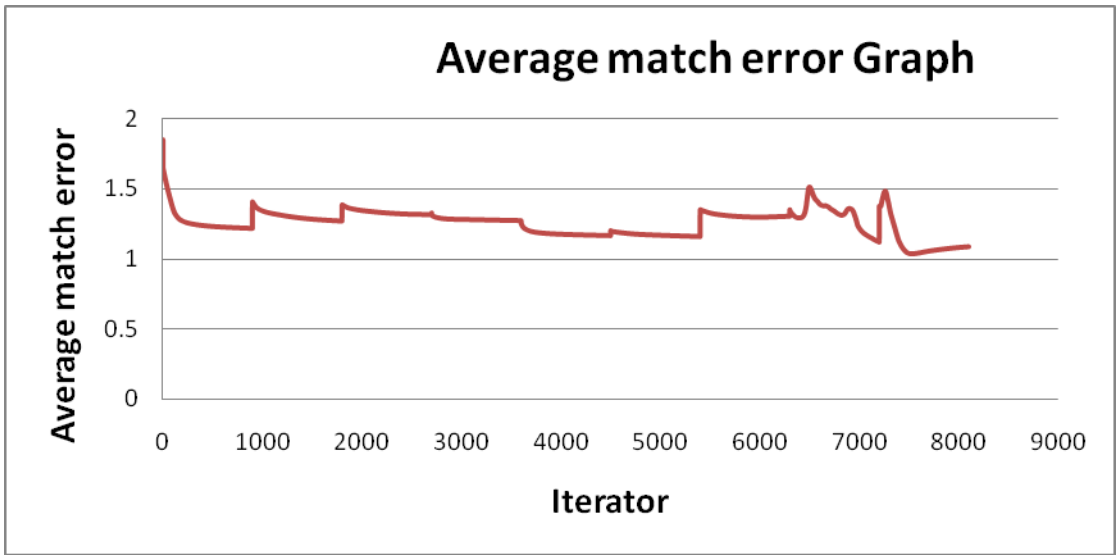
**Αριθμός κρυφών νευρώνων: 45**

**Διάνυσμα εισόδου : 46 στοιχεία**

**Ρυθμός μάθησης : 0.002**

**Αριθμός εποχών : 900 x 9 = 8100**

Σε αυτό το σύνολο δεδομένων φαίνονται τα καλύτερα αποτελέσματα που έχουν προκύψει. Τα στοιχεία του δικτύου είναι τα ίδια με το σύνολο δεδομένων 6, αλλά τα αποτελέσματα είναι ελαφρώς καλύτερα. Αυτό συμβαίνει λόγω της αρχικοποίησης των τιμών των κέντρων στην αρχή του προγράμματος, πριν την έναρξη της εκπαίδευσης του δικτύου. Τα vector λοιπόν αυτά, αρχικοποιούνται με τυχαίες τιμές μεταξύ 0-1. Αυτή είναι η μόνη διαφοροποίηση που προκύπτει μεταξύ των δυο εκπαιδεύσεων και όπως βλέπουμε προκαλεί μια αξιοσημείωτη διαφορά.




**correct decisions :328**
**wrong decisions :152**
**correct MORE :142**
**correct LESS :186**
**wrong MORE :75**
**wrong LESS :77**

Σε αυτή την περίπτωση, είχαμε 328 σωστές προβλέψεις δηλαδή ποσοστό επιτυχίας **68,33%** ενώ το average error έφτασε το **1,08** ανά αγώνα. Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **26,41%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

**Αποτελέσματα με μια παράμετρο να μεταβάλλεται**

Σίγμα (s) = [1..8]

Coefficient (c) = [0,10..0,50]

Learning Rate (n) = 0,002

Περίοδοι εκπαίδευσης : 2000/2001 – 2004/2005

Αριθμός κρυφών νευρώνων: 30

Διάνυσμα εισόδου : 46 στοιχεία

| **Επαναλήψεις** | **500** | **800** | **1200** | **1800** | **3000** |
|---|---|---|---|---|---|
| **Επιτυχίες** | 304 | 306 | 306 | 306 | 306 |
| **Αποτυχίες** | 176 | 174 | 174 | 174 | 174 |
| **Επιτυχημένα MORE** | 137 | 136 | 136 | 135 | 135 |
| **Επιτυχημένα LESS** | 167 | 170 | 170 | 171 | 171 |
| **Αποτυχημένα MORE** | 105 | 102 | 102 | 101 | 101 |
| **Αποτυχημένα LESS** | 71 | 72 | 72 | 73 | 73 |

Σε αυτή την περίπτωση επιχείρησα να κρατήσω σταθερές όλες τις παραμέτρους του δικτύου μου και να μεταβάλλω μόνο τον αριθμό των εποχών. Όπως φαίνεται στον πιο πάνω πίνακα τα αποτελέσματα δεν αλλάζουν δραματικά. Από ένα σημείο και μετά μάλιστα, οι αλλαγές που προκύπτουν αυξάνοντας τον αριθμό των επαναλήψεων είναι από ελάχιστες έως μηδαμινές.

Ας δούμε τώρα τον τρόπο που αλλάζουν τα 5 πρώτα αποτελέσματα στα οποία εκπαιδεύεται το δίκτυο μας την πρώτη του αγωνιστική περίοδο. Τα αποτελέσματα πάρθηκαν από το σύνολο δεδομένων 6.

| Επαναλήψεις<br>Επιθυμητό | 1 | 100 | 200 | 300 | 500 |
|---|---|---|---|---|---|
| **0** | 6,64 | 1,9 | 1,71 | 1,65 | 1,57 |
| **2** | 3,09 | 1,27 | 1,33 | 1,41 | 1,5 |
| **1** | 7,7 | 2,36 | 2,17 | 2,13 | 2,05 |
| **4** | 13,11 | 3,19 | 2,72 | 2,62 | 2,55 |
| **2** | 1,95 | 0,81 | 0,84 | 0,89 | 0,91 |
| **2** | 11,55 | 2,25 | 1,94 | 1,87 | 1,8 |

| **Επανάληψη 0** | **Επιθυμητό** | **Αποτέλεσμα** | **Λάθος** |
|---|---|---|---|
| WOLVES - SHEFFIELDUTD | 0 | 6,64 | 6,64 |
| HUDDERSFIELD - AFCWIMBLEDON | 2 | 3,09 | 1,09 |
| SHEFFIELDWED - NOTTMFOREST | 1 | 7,70 | 6,70 |
| Q.P.R - GILLINGHAM | 4 | 13,11 | 9,11 |
| SHEFFIELDUTD – BLACKBURN | 2 | 1,95 | 0,05 |
| AFCWIMBLEDON - WOLVES | 2 | 11,55 | 9,55 |

| **Επανάληψη 100** | **Επιθυμητό** | **Αποτέλεσμα** | **Λάθος** |
|---|---|---|---|
| WOLVES - SHEFFIELDUTD | 0 | 1,90 | 1,90 |
| HUDDERSFIELD - AFCWIMBLEDON | 2 | 1,27 | 0,72 |
| SHEFFIELDWED - NOTTMFOREST | 1 | 2,36 | 1,36 |
| Q.P.R - GILLINGHAM | 4 | 3,19 | 0,80 |
| SHEFFIELDUTD - BLACKBURN | 2 | 0,81 | 1,19 |
| AFCWIMBLEDON - WOLVES | 2 | 2,25 | 0,25 |

| **Επανάληψη 200** | **Επιθυμητό** | **Αποτέλεσμα** | **Λάθος** |
|---|---|---|---|
| WOLVES - SHEFFIELDUTD | 0 | 1,71 | 1,71 |
| HUDDERSFIELD - AFCWIMBLEDON | 2 | 1,33 | 0,67 |
| SHEFFIELDWED - NOTTMFOREST | 1 | 2,17 | 1,17 |
| Q.P.R - GILLINGHAM | 4 | 2,72 | 1,28 |
| SHEFFIELDUTD - BLACKBURN | 2 | 0,84 | 1,16 |
| AFCWIMBLEDON - WOLVES | 2 | 1,94 | 0,06 |

| **Επανάληψη 300** | **Επιθυμητό** | **Αποτέλεσμα** | **Λάθος** |
|---|---|---|---|
| WOLVES - SHEFFIELDUTD | 0 | 1,65 | 1,65 |
| HUDDERSFIELD - AFCWIMBLEDON | 2 | 1,41 | 0,59 |
| SHEFFIELDWED - NOTTMFOREST | 1 | 2,13 | 1,13 |
| Q.P.R - GILLINGHAM | 4 | 2,62 | 1,38 |
| SHEFFIELDUTD - BLACKBURN | 2 | 0,89 | 1,11 |
| AFCWIMBLEDON - WOLVES | 2 | 1,87 | 0,13 |

| **Επανάληψη 500** | **Επιθυμητό** | **Αποτέλεσμα** | **Λάθος** |
|---|---|---|---|
| WOLVES - SHEFFIELDUTD | 0 | 1,57 | 1,57 |
| HUDDERSFIELD - AFCWIMBLEDON | 2 | 1,50 | 0,50 |
| SHEFFIELDWED - NOTTMFOREST | 1 | 2,05 | 1,05 |
| Q.P.R - GILLINGHAM | 4 | 2,55 | 1,45 |
| SHEFFIELDUTD - BLACKBURN | 2 | 0,91 | 1,09 |
| AFCWIMBLEDON - WOLVES | 2 | 1,80 | 0,20 |

Επίσης, πιο κάτω φαίνεται το πώς "εξελίσσεται" το δίκτυο μας δια μέσου της εκπαίδευσης του. Επειδή τα vector των κέντρων έχουν 46 θέσεις και είναι πολλές για να τις σημειώσουμε όλες, θα επικεντρωθούμε στις πρώτες 5 θέσεις του κάθε vector. Φαίνεται επίσης το Coefficient και το Sigma του κάθε κέντρου. Τα στοιχεία του δικτύου πάρθηκαν από το σύνολο δεδομένων 9, στο πρώτο παιχνίδι των επαναλήψεων 0, 100, 200 και 500. Χρησιμοποιήθηκαν πληροφορίες από τους 3 πρώτους νευρώνες, και πάλι λόγω μεγάλης ποσότητας δεδομένων.

**Επανάληψη 0**

**Κρυφός Νευρώνας 0**

**C** = 4.917896248531558 **S** = 2.50997350335249

**Centre**

**[0]**= 0.13645663360842958 **[1]**= 0.5663954543479727

**[2]**= 0.3606050360592566 **[3]**= 0.657903690748723

**Κρυφός Νευρώνας 1**

**C** = 5.917896248531558 **S** = 1.5899999886198473

**Centre**

**[0]**= 0.08037345639628457 **[1]**= 0.9728395504103537

**[2]**= 0.26096643833676514 **[3]**= 0.5184637562209763

**Κρυφός Νευρώνας 2**

**C** = 2.917896248531558 **S** = 1.7699999950145051

**Centre**

**[0]**= 0.6213166939567516 **[1]**= 0.23330618808242454

**[2]**= 0.080647776229901 **[3]**= 0.06435019210387724

**Επανάληψη 100**

**Κρυφός Νευρώνας 0**

**C** = 1.087183630753447 **S** = 2.5015799144494766

**Centre**

**[0]**= -0.012485876437618034 **[1]**= 0.5379223149892024

**[2]**= 0.2948882863535896 **[3]**= 0.6768897434314389

**Κρυφός Νευρώνας 1**

**C** = 2.0871836307534073 **S** = 1.5896550761351018

**Centre**

**[0]**= 0.03425565402448357 **[1]**= 0.9840618896831893

**[2]**= 0.23526588064934165 **[3]**= 0.516557530860462

**Κρυφός Νευρώνας 2**

**C** = 3.087183630753409 **S** = 1.2199985587138038

**Centre**

**[0]**= 0.6564168569892209 **[1]**= 0.7117447497131311

**[2]**= 0.6043866357489736 **[3]**= 0.8541105087873622

**Επανάληψη 200**

**Κρυφός Νευρώνας 0**

**C** = 1.3998820629959658 **S** = 2.500275667282734

**Centre**

**[0]**= -0.08629899537994848 **[1]**= 0.5242671232547111

**[2]**= 0.26047034993691326 **[3]**= 0.6828714107800552

**Κρυφός Νευρώνας 1**

**C** = 2.399882062995891 **S** = 1.5895570293605643

**Centre**

**[0]**= 0.014425205671321677 **[1]**= 0.9890807950854598

**[2]**= 0.22395620467678373 **[3]**= 0.515761342792605

**Κρυφός Νευρώνας 2**

**C** = 3.399882062995893 **S** = 1.2199976435500712

**Centre**

**[0]**= 0.6560148258109375 **[1]**= 0.7116813172758463

**[2]**= 0.6041539992948234 **[3]**= 0.8543991343377675

**Επανάληψη 500**

**Κρυφός Νευρώνας 0**

**C** = 2.148034951643177 **S** = 2.49812424015971

**Centre**

**[0]**= -0.2134167150108806 **[1]**= 0.5000811852920514

**[2]**= 0.2013103883201993 **[3]**= 0.6930720488882763

**Κρυφός Νευρώνας 1**

**C** = 3.1480349516431425 **S** = 1.589346145283959

**Centre**

**[0]**= -0.02465063870459963 **[1]**= 0.9997320249292012

**[2]**= 0.20184837815018047 **[3]**= 0.5146350201014782

**Κρυφός Νευρώνας 2**

**C** = 4.148034951643163 **S** = 1.2199944072864084

**Centre**

**[0]**= 0.6549288668065906 **[1]**= 0.7115041283681471

**[2]**= 0.6034998102412362 **[3]**= 0.8552126806445364

Όπως παρατηρούμε, οι μεταβολές στη δομή του δικτύου μας γίνονται με μικρούς ρυθμούς. Αυτό οφείλεται στο συντελεστή μάθησης, τον οποίο επιλέξαμε να είναι αρκετά μικρός έτσι ώστε να αποφύγουμε τις απότομες μεταβολές που αποδεδειγμένα επέφεραν άσχημα αποτελέσματα. Ακόμη, όπως φαίνεται πιο πάνω έχω επιτρέψει οι τιμές των vector των κρυφών μου νευρώνων να μπορούν να πάρουν και αρνητικές τιμές. Αυτή η απόφαση προέκυψε και πάλι εμπειρικά, μετά από δοκιμές που έκανα για την συγκεκριμένη λεπτομέρεια και είδα ότι απέφερε καλύτερα αποτελέσματα έτσι.

**Στοιχηματικό κέρδος με πραγματικές αποδόσεις**

Μετά την ολοκλήρωση του δικτύου, εκτός από την σύγκριση των αποτελεσμάτων που προέκυψαν με τα δεδομένα εκπαίδευσης και ελέγχου, προσπάθησα να πάρω κάποια αποτελέσματα από τα παιχνίδια που ήταν προγραμματισμένα να εξελιχθούν το διάστημα που ακολούθησε και για τα οποία είχα πραγματικές αποδόσεις. Έτσι μπόρεσα να βγάλω αποτελέσματα όσον αφορά το στοιχηματικό κέρδος που μπορεί να προκύψει από την χρήση του Νευρωνικού μου Δικτύου.
Όπως βλέπουμε σε κάθε αγωνιστική στοιχημάτιζα το ποσό του ενός ευρώ για κάθε παιχνίδι το οποίο πρόβλεπε το δίκτυο μου. Ανάλογα με το αν το αποτέλεσμα ήταν επιτυχημένο ή όχι, επιστρεφόταν είτε η απόδοση που προσφερόταν για την συγκεκριμένη πρόβλεψη είτε χανόταν το ευρώ που στοιχηματίσαμε. Οι αποδόσεις πάρθηκαν από την εταιρεία στοιχημάτων BWIN που θεωρείται η μεγαλύτερη και η πιο αξιόπιστη στον κόσμο.

| **Αγώνας** | **Less** | **More** | **Πρόβλεψη** | **Επιστροφή** |
|---|---|---|---|---|
| CRYSTALPALACE CARDIFF | 1,65 | 1,95 | less | 0 |
| BARNSLEY DONCASTER | 1,75 | 1,95 | less | 1,75 |
| COVENTRY SHEFFIELDWED | 1,7 | 2,02 | less | 1,7 |
| DERBY LEICESTER | 1,65 | 2,1 | more | 0 |
| PETERBOROUGH BRISTOLCITY | 1,8 | 1,9 | more | 0 |
| PLYMOUTH BLACKPOOL | 1,85 | 1,85 | less | 1,85 |
| PRESTON Q.P.R | 1,9 | 1,8 | less | 0 |
| READING WESTBROM | 2,1 | 1,65 | more | 0 |
| SWANSEA IPSWICH | 1,6 | 2,2 | less | 1,6 |
| WATFORD MIDDLESBROUGH | 1,8 | 1,9 | less | 1,8 |
| SHEFFIELDUTD SCUNTHORPE | 1,8 | 1,9 | more | 0 |
| NEWCASTLE NOTTMFOREST | 1,8 | 1,9 | less | 1,8 |

**Συνολικό ποσό στοιχηματισμού: €12**

**Επιστροφές : €10,5**

| Αγώνας | Less | More | Πρόβλεψη | Επιστροφή |
|---|---|---|---|---|
| SCUNTHORPE BLACKPOOL | 1,75 | 1,95 | more | 1,95 |
| WESTBROM LEICESTER | 1,95 | 1,75 | more | 1,75 |
| BRISTOLCITY NOTTMFOREST | 1,7 | 2,02 | less | 1,7 |
| COVENTRY DERBY | 1,7 | 2,02 | less | 1,7 |
| DONCASTER PLYMOUTH | 1,7 | 2,02 | more | 2,02 |
| CARDIFF SWANSEA | 1,6 | 2,2 | less | 0 |
| IPSWICH READING | 1,75 | 1,95 | more | 1,95 |
| MIDDLESBROUGH CRYSTALPALACE | 1,7 | 2,02 | less | 1,7 |
| PETERBOROUGH NEWCASTLE | 1,85 | 1,85 | more | 1,85 |
| PRESTON WATFORD | 1,85 | 1,85 | less | 1,85 |
| Q.P.R SHEFFIELDWED | 1,7 | 2,02 | more | 0 |
| SHEFFIELDUTD BARNSLEY | 1,72 | 2 | more | 0 |

**Συνολικό ποσό στοιχηματισμού: €12**

**Επιστροφές : €16,47**

| Αγώνας | Less | More | Πρόβλεψη | Επιστροφή |
|---|---|---|---|---|
| BRISTOLCITY SWANSEA | 1,55 | 2,3 | less | 1,55 |
| BARNSLEY DERBY | 1,75 | 1,95 | less | 1,75 |
| CARDIFF READING | 1,8 | 1,9 | more | 0 |
| CRYSTALPALACE Q.P.R | 1,65 | 2,1 | less | 1,65 |
| DONCASTER WESTBROM | 1,75 | 1,95 | more | 1,95 |
| MIDDLESBROUGH SHEFFIELDWED | 1,75 | 1,95 | less | 1,75 |
| NEWCASTLE BLACKPOOL | 1,95 | 1,75 | more | 1,75 |
| NOTTMFOREST IPSWICH | 1,65 | 2,1 | more | 2,1 |
| PETERBOROUGH LEICESTER | 1,75 | 1,95 | less | 0 |
| PRESTON SCUNTHORPE | 1,85 | 1,85 | more | 1,85 |
| SHEFFIELDUTD COVENTRY | 1,65 | 2,1 | less | 1,65 |
| WATFORD PLYMOUTH | 1,7 | 2,02 | more | 0 |

**Συνολικό ποσό στοιχηματισμού: €12**

**Επιστροφές : €16**

| Αγώνας | Less | More | Πρόβλεψη | Επιστροφή |
|---|---|---|---|---|
| BLACKPOOL NOTTMFOREST | 1,65 | 2,1 | less | 0 |
| COVENTRY PRESTON | 1,75 | 1,95 | less | 1,75 |
| DERBY CRYSTALPALACE | 1,65 | 2,1 | less | 1,65 |
| IPSWICH DONCASTER | 1,7 | 2,02 | less | 1,7 |
| LEICESTER WATFORD | 1,7 | 2,02 | less | 0 |
| Q.P.R CARDIFF | 1,85 | 1,85 | more | 0 |
| READING PETERBOROUGH | 1,8 | 1,9 | less | 0 |
| SCUNTHORPE BRISTOLCITY | 1,75 | 1,95 | more | 1,95 |
| SWANSEA BARNSLEY | 1,65 | 2,1 | less | 0 |
| WESTBROM MIDDLESBROUGH | 1,85 | 1,85 | more | 0 |
| SHEFFIELDWED SHEFFIELDUTD | 1,7 | 2,02 | more | 0 |
| PLYMOUTH NEWCASTLE | 1,7 | 2,02 | less | 1,7 |

**Συνολικό ποσό στοιχηματισμού: € 12**

**Επιστροφές : €8,75**

| Αγώνας | Less | More | Πρόβλεψη | Επιστροφή |
|---|---|---|---|---|
| BARNSLEY Q.P.R | 1,7 | 2,02 | less | 1,7 |
| BRISTOLCITY DERBY | 1,65 | 2,1 | more | 2,1 |
| CARDIFF SHEFFIELDWED | 1,7 | 2,02 | more | 2,02 |
| DONCASTER SCUNTHORPE | 1,85 | 1,85 | more | 1,85 |
| MIDDLESBROUGH COVENTRY | 1,68 | 2,02 | more | 0 |
| NEWCASTLE IPSWICH | 1,95 | 1,75 | more | 1,75 |
| NOTTMFOREST PLYMOUTH | 1,7 | 2,02 | more | 2,02 |
| PETERBOROUGH BLACKPOOL | 2,02 | 1,7 | less | 2,02 |
| PRESTON LEICESTER | 1,68 | 2,05 | less | 1,68 |
| SHEFFIELDUTD SWANSEA | 1,6 | 2,2 | less | 1,6 |
| WATFORD READING | 1,9 | 1,8 | more | 1,8 |
| CRYSTALPALACE WESTBROM | 1,7 | 2,02 | less | 1,7 |

**Συνολικό ποσό στοιχηματισμού: €12**

**Επιστροφές : €20,24**

| Αγώνας | Less | More | Πρόβλεψη | Επιστροφή |
|---|---|---|---|---|
| BLACKPOOL BRISTOLCITY | 1,95 | 1,75 | more | 0 |
| COVENTRY WATFORD | 1,75 | 1,95 | more | 1,95 |
| DERBY CARDIFF | 1,7 | 2,02 | less | 1,7 |
| IPSWICH SHEFFIELDUTD | 1,7 | 2,02 | less | 0 |
| LEICESTER MIDDLESBROUGH | 1,75 | 1,95 | less | 1,75 |
| PLYMOUTH PETERBOROUGH | 1,85 | 1,85 | less | 0 |
| Q.P.R NEWCASTLE | 1,85 | 1,85 | more | 0 |
| READING PRESTON | 2 | 1,72 | less | 0 |
| SCUNTHORPE NOTTMFOREST | 1,75 | 1,95 | less | 0 |
| SHEFFIELDWED CRYSTALPALACE | 1,75 | 1,95 | more | 1,95 |
| SWANSEA DONCASTER | 1,75 | 1,95 | less | 1,75 |
| WESTBROM BARNSLEY | 2,02 | 1,7 | more | 0 |

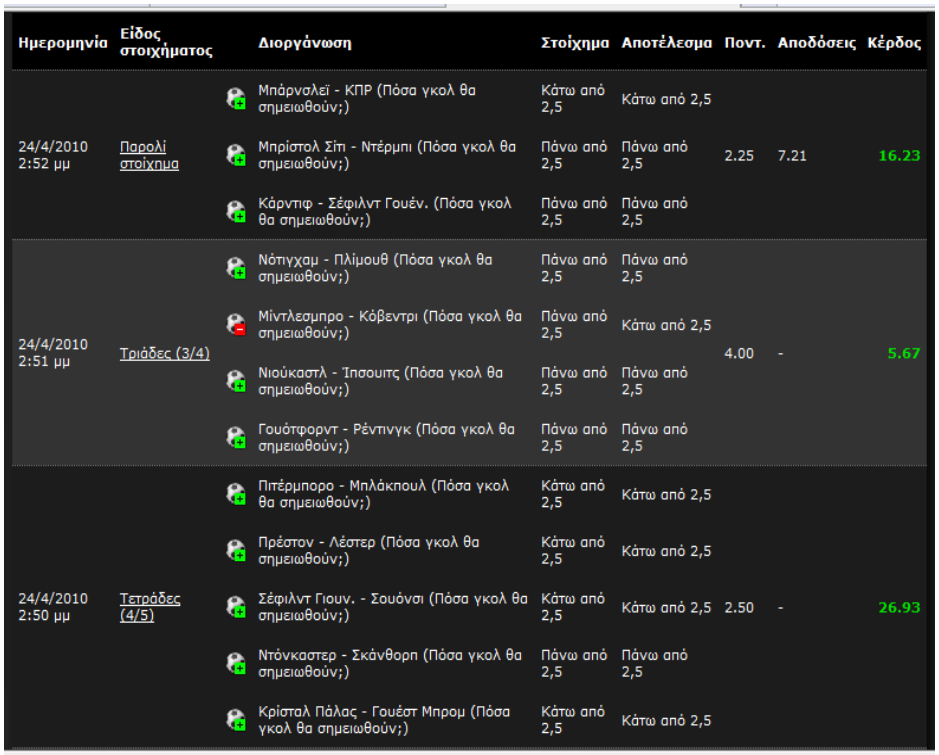
**Συνολικό ποσό στοιχηματισμού: €12**

**Επιστροφές : €9,1**

Τα αποτελέσματα αυτά πάρθηκαν τις αγωνιστικές 40-46 του πρωταθλήματος της Championship το οποίο μας απασχολεί, από τις 27/03/2010 μέχρι τις 02/05/2010. Όπως ανέφερα προηγουμένως οι τιμές είναι αυτές που προσέφερε η εταιρεία BWIN και τα πιο πάνω στοιχήματα τοποθετήθηκαν πραγματικά. Σε σύνολο λοιπόν 6 αγωνιστικών δηλαδή 72 παιχνιδιών, τοποθετήθηκαν 72 διαφορετικά στοιχήματα. Τα 45 από αυτά ήταν κερδισμένα δηλαδή ένα ποσοστό της τάξης του 62,5%. Οι επιστροφές που είχαμε από τα κερδισμένα στοιχήματα ήταν €81,02 αποφέροντας αύξηση 12,52% του αρχικού κεφαλαίου.

| Ημερομηνία | Είδος στοιχήματος | Διοργάνωση | Στοίχημα | Αποτέλεσμα | Ποντ. | Αποδόσεις | Κέρδος |
|---|---|---|---|---|---|---|---|
| 8/4/2010 3:44 μμ | Μονό στοίχημα | Σέφιλντ Γιουν. - Κόβεντρι (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.02 | 1.65 | 1.68 |
| 8/4/2010 3:44 μμ | Μονό στοίχημα | Πρέστον - Σκάνθορπ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | 1.00 | 1.85 | 1.85 |
| 8/4/2010 3:43 μμ | Μονό στοίχημα | Πίτερμπορο - Λέστερ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Πάνω από 2,5 | 1.00 | 1.75 | - |
| 8/4/2010 3:43 μμ | Μονό στοίχημα | Μπάρνσλεϊ - Ντέρμπι (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.00 | 1.75 | 1.75 |
| 8/4/2010 3:43 μμ | Μονό στοίχημα | Νιούκαστλ - Μπλάκπουλ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | 1.00 | 1.75 | 1.75 |
| 8/4/2010 3:43 μμ | Μονό στοίχημα | Νότιγχαμ - Ίπσουιτς (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | 1.00 | 2.10 | 2.10 |
| 8/4/2010 3:43 μμ | Μονό στοίχημα | Ντόνκαστερ - Γουέστ Μπρομ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | 1.00 | 1.95 | 1.95 |
| 8/4/2010 3:42 μμ | Μονό στοίχημα | Μίντλεσμπρο - Σέφιλντ Γουέν. (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.00 | 1.75 | 1.75 |
| 8/4/2010 3:42 μμ | Μονό στοίχημα | Κρίσταλ Πάλας - ΚΠΡ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.00 | 1.65 | 1.65 |
| 8/4/2010 3:42 μμ | Μονό στοίχημα | Κάρντιφ - Ρέντινγκ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Κάτω από 2,5 | 1.00 | 1.90 | - |
| 8/4/2010 3:41 μμ | Μονό στοίχημα | Γουότφορντ - Πλίμουθ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Κάτω από 2,5 | 1.00 | 2.02 | - |
| 8/4/2010 3:39 μμ | Μονό στοίχημα | Μπρίστολ Σίτι - Σουόνσι (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.00 | 1.55 | 1.55 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Κόβεντρι - Πρέστον (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.00 | 1.75 | 1.75 |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Λέστερ - Γουότφορντ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Πάνω από 2,5 | 1.00 | 1.70 | - |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Μπλάκπουλ - Νότιγχαμ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Πάνω από 2,5 | 1.00 | 1.65 | - |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Ντέρμπι - Κρίσταλ Πάλας (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.00 | 1.65 | 1.65 |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Ρέντινγκ - Πίτερμπορο (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Πάνω από 2,5 | 1.00 | 1.80 | - |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | ΚΠΡ - Κάρντιφ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Κάτω από 2,5 | 1.00 | 1.85 | - |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Σκάνθορπ - Μπρίστολ Σίτι (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | 1.00 | 1.95 | 1.95 |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Γουέστ Μπρομ - Μίντλεσμπρο (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Κάτω από 2,5 | 1.00 | 1.85 | - |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Σέφιλντ Γουέν. - Σέφιλντ Γιουν. (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Κάτω από 2,5 | 1.00 | 2.02 | - |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Πλίμουθ - Νιούκαστλ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.00 | 1.70 | 1.70 |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Σουόνσι - Μπάρνσλεϊ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Πάνω από 2,5 | 1.00 | 1.65 | - |
| 15/4/2010 3:25 μμ | Μονό στοίχημα | Ίπσουιτς - Ντόνκαστερ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 1.00 | 1.70 | 1.70 |



| Ημερομηνία | Είδος στοιχήματος | Διοργάνωση | Στοίχημα | Αποτέλεσμα | Ποντ. | Αποδόσεις | Κέρδος |
|---|---|---|---|---|---|---|---|
| | | Μπάρνσλεϊ - ΚΠΡ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | | | |
| 24/4/2010 2:52 μμ | Παρολί στοίχημα | Μπρίστολ Σίτι - Ντέρμπι (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | 2.25 | 7.21 | 16.23 |
| | | Κάρντιφ - Σέφιλντ Γουέν. (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | | | |
| | | Νόττιγχαμ - Πλίμουθ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | | | |
| | | Μίντλεσμπρο - Κόβεντρι (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Κάτω από 2,5 | | | |
| 24/4/2010 2:51 μμ | Τριάδες (3/4) | | | | 4.00 | - | 5.67 |
| | | Νιούκαστλ - Ίπσουιτς (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | | | |
| | | Γουότφορντ - Ρέντινγκ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | | | |
| | | Πίτερμπορο - Μπλάκπουλ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | | | |
| | | Πρέστον - Λέστερ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | | | |
| 24/4/2010 2:50 μμ | Τετράδες (4/5) | Σέφιλντ Γιουν. - Σουόνσι (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | 2.50 | - | 26.93 |
| | | Ντόνκαστερ - Σκάνθορπ (Πόσα γκολ θα σημειωθούν;) | Πάνω από 2,5 | Πάνω από 2,5 | | | |
| | | Κρίσταλ Πάλας - Γουέστ Μπρομ (Πόσα γκολ θα σημειωθούν;) | Κάτω από 2,5 | Κάτω από 2,5 | | | |

Στο πιο πάνω στοίχημα επέλεξα να συνδυάσω τα παιχνίδια και να μην δημιουργήσω ένα στοίχημα για το κάθε ένα, όπως έκανα προηγουμένως. Σε περίπτωση που έχουμε ικανοποιητικό αριθμό επιτυχιών την συγκεκριμένη αγωνιστική όπως για παράδειγμα σε αυτή που φαίνεται πιο πάνω (11 στα 12), ενδεχομένως να πολλαπλασιαστεί το κέρδος. Υπάρχει όμως και η περίπτωση να λειτουργήσει εντελώς εις βάρος μας στοιχηματικά αφού αν για παράδειγμα έχουμε 9 επιτυχίες σε σύνολο 12 αγώνων και δημιουργήσουμε συνδυασμούς των τεσσάρων παιχνιδιών υπάρχει ενδεχόμενο να χάσουν και τα τρία μας στοιχήματα. Η δημιουργία τέτοιων συστημάτων από αγώνες θα μπορούσε να αποτελέσει πρόταση για μελέτη και μελλοντική εργασία. Προς το παρόν η προσπάθεια μου θα επικεντρωθεί στο κατά πόσον υπάρχει περιθώριο για απόκτηση κέρδους με τα ποσοστά επιτυχίας που έχουν επιτευχθεί από το δίκτυο μου και με βάση τις αποδόσεις που προσφέρονται στην αγορά χωρίς να συνδυάζονται οι αγώνες.

Όπως είδαμε λοιπόν, για τους αγώνες στους οποίους είχαμε πρόσβαση σε πραγματικές αποδόσεις μπορούμε να μιλήσουμε για πραγματικό κέρδος αλλά και για ποσοστά αύξησης του αρχικού μας κεφαλαίου. Δυστυχώς αυτή η δυνατότητα δεν υπάρχει για τα προηγούμενα παιχνίδια τα οποία έχουν ολοκληρωθεί, αφού δεν αποθηκεύονται πουθενά αυτές οι πληροφορίες. Όπως όμως μπορούμε να διαπιστώσουμε πολύ εύκολα, ο μέσος

όρος απόδοσης που δίδεται από την εταιρεία στην οποία αναφερόμαστε είναι πάντα 1,85 ή πολύ κοντά σε αυτό. Επίσης με δεδομένο ότι το δίκτυο μας δεν βρίσκει μόνο τα πιθανά αποτελέσματα, κάτι που φαίνεται στα παραδείγματα πιο πάνω, μπορούμε να θεωρήσουμε τον αριθμό αυτό εντελώς αντιπροσωπευτικό.

$$\frac{1,95 + 1,75}{2} = 1,85 \qquad \frac{1,7 + 2,02}{2} = 1,86 \qquad \frac{1,65 + 2,10}{2} = 1,87$$

$$\frac{1,8 + 1,9}{2} = 1,85 \qquad \frac{1,85 + 1,85}{2} = 1,85$$

Με αυτά τα δεδομένα αν θεωρήσουμε ως βέλτιστο αποτέλεσμα του δικτύου μας τις $\frac{320}{480}$ επιτυχίες που πέτυχε με το σύνολο παραμέτρων 9 που αναφέραμε πιο πάνω την αγωνιστική περίοδο 2009 σαν testing set τότε έχουμε πετύχει ποσοστό επιτυχίας της τάξεως του **68,33%.** Το ποσοστό αυτό μπορεί να θεωρηθεί αρκετά ικανοποιητικό αφού αποτελεί το ψηλότερο από τα ποσοστά επιτυχίας που έχουν πετύχει οι τέσσερεις προηγούμενοι οι οποίοι ασχολήθηκαν με τη συγκεκριμένη έρευνα και των οποίων η διπλωματική αυτή εργασία αποτελεί συνέχεια της μελέτης και των συμπερασμάτων τους.

Όσον αφορά το στοιχηματικό κέρδος και κατ' επέκταση την αύξηση του αρχικού κεφαλαίου το οποίο θα επενδύαμε, αυτό θα ήταν της τάξεως του **26,41%**. Το κέρδος αυτό υπολογίζεται με μέσο όρο 1,85 απόδοση σε κάθε παιχνίδι όπως αυτή υπολογίστηκε πιο πάνω. Το ποσοστό αυτό είναι επίσης το πιο μεγάλο που έχει επιτευχθεί στις πέντε αυτές έρευνες που έχουν γίνει συμπεριλαμβανομένης και της δικής μου. Επίσης αξίζει να αναφερθεί ότι στις δύο προηγούμενες έρευνες που έγιναν στο συγκεκριμένο πρόβλημα, οι Turner [3] και Νεοκλέους [4] χρησιμοποίησαν σαν μέσο όρο απόδοσης το 2 και όχι το 1,85. Ίσως, με την πάροδο των χρόνων οι εταιρείες στοιχημάτων να μετέβαλαν τις τιμές τους αφού σίγουρα και οι ίδιες χρησιμοποιούν πλέον πιο εξελιγμένα συστήματα και μεθόδους έτσι ώστε να διασφαλίζουν το κέρδος τους. Στις μέρες μας πάντως είναι αδύνατο να βρεις εταιρεία στοιχημάτων που να σου προσφέρει τέτοιες αποδόσεις για το συγκεκριμένο είδος στοιχήματος. Με αυτά τα δεδομένα λοιπόν το 26,41% αποτελεί ουσιαστικότατη βελτίωση του 23% που είχε επιτευχθεί προηγουμένως αν αναλογιστούμε την μεγάλη διαφορά στο μέσο όρο απόδοσης του σήμερα με τα προηγούμενα χρόνια.

Το ποσοστό αύξησης του κεφαλαίου μπορεί να υπολογιστεί είτε

**Α)** πολλαπλασιάζοντας το ποσοστό επιτυχίας με τον μέσο όρο απόδοσης ανά παιχνίδι δηλαδή

$68,33 * 1,85 = 126,41$ ανά **100** στοιχηματικές μονάδες

**Β)** είτε πολλαπλασιάζοντας τις επιτυχείς προβλέψεις με 1,85 και διαιρώντας δια τον συνολικό αριθμό προβλέψεων δηλαδή

$\left(\frac{328 * 1,85}{480}\right) = 1,2641$ άρα ποσοστό κέρδους **26,41%.**

Εύκολα μπορεί κάποιος να αντιληφθεί ότι ένα ποσοστό της τάξεως του 26,41% είναι αρκετά αξιοσημείωτο. Η φύση όμως του ποδοσφαιρικού στοιχήματος είναι τέτοια που τίποτα δεν είναι εξασφαλισμένο. Είναι γενικά αποδεκτό στον κόσμο του στοιχήματος και επιβεβαιωμένο από επαγγελματίες τζογαδόρους αλλά και από προσωπική μου εμπειρία ότι ειδικά στο ποδόσφαιρο τα πάντα μπορούν να συμβούν και κάθε αποτέλεσμα είναι πιθανό. Ακόμη και για τα αποτελέσματα τα οποία δεν προβλέφτηκαν σωστά, δεν μπορείς να ξέρεις με σιγουριά αν το αποτέλεσμα το οποίο πρόβλεψε το δίκτυο σου δεν ήταν το "πιθανότερο" για να προκύψει και ο λόγος που έχασες δεν ήταν κάποιοι αστάθμητοι παράγοντες και συγκυρίες όπως η τύχη, οι καιρικές συνθήκες, ο διαιτητής κτλ.

Ένας άλλος τρόπος να ελέγξεις αν στο δίκτυο σου πράγματι έγινε σωστή εκπαίδευση είναι κατά την γνώμη μου να συγκρίνεις τις προβλέψεις τις οποίες δίνει με τις αποδόσεις των εταιρειών στοιχημάτων, ανεξαρτήτως του αποτελέσματος του αγώνα. Αν λοιπόν η πλειοψηφία των προβλέψεων του δικτύου ταυτίζεται με τα πιο "πιθανά" για να προκύψουν αποτελέσματα, όπως αυτά δίνονται από τις εταιρείες στοιχημάτων τότε σίγουρα το νευρωνικό δίκτυο δουλεύει σωστά. Αν και γι' αυτό δεν υπάρχει μαθηματική απόδειξη, εντούτοις το μέγεθος της βιομηχανίας των τυχερών παιχνιδιών και ιδίως των ποδοσφαιρικών στοιχημάτων σε συνδυασμό με τον τεράστιο τζίρο των δισεκατομμυρίων δολαρίων που διακυβεύονται κάθε χρόνο δεν αφήνουν περιθώρια σε κανένα να το αμφισβητήσει.

Όσον αφορά το δικό μας δίκτυο, από όποια πτυχή και αν προσπαθήσεις να το ελέγξεις τα αποτελέσματα του φαίνονται και είναι επιτυχημένα. Τα ποσοστά επιτυχίας ανέβηκαν αισθητά ενώ το ίδιο έγινε και με το ποσοστό αύξησης του αρχικού κεφαλαίου το οποίο επενδύθηκε στα αποτελέσματα της έρευνας μας.

## 7.2 Αποτελέσματα SVM simulator

Όπως ανέφερα και στο προηγούμενο κεφάλαιο για την εξαγωγή αποτελεσμάτων με την χρήση των Support Vector Machines θα χρησιμοποιηθεί ο προσομοιωτής SVM Classifer. Ο σκοπός μου σε αυτή την περίπτωση ήταν να εξαχθούν κάποια αποτελέσματα έτσι ώστε να μπορούν να συγκριθούν με τα αποτελέσματα του Radial Basis Function δικτύου. Στόχος είναι με βάση την σύγκριση αυτή να εξεταστεί κατά πόσο η χρήση των Support Vector Machines μπορεί να βοηθήσει στην βελτίωση των αποτελεσμάτων στην πρόβλεψη ποδοσφαιρικών αποτελεσμάτων και πιο συγκεκριμένα στο δικό μας πρόβλημα.
Χρησιμοποιώντας λοιπόν τα ίδια δεδομένα με το Radial Basis Function Network αλλά στη νέα τους μορφή έτσι ώστε να είναι κατανοητά από τον SVM simulator, έτρεξα όλους τους πιθανούς συνδυασμούς των επιλογών που μου έδινε την δυνατότητα να αλλάξω ο προσομοιωτής. Σε αυτή την περίπτωση το training set είναι για όλες τις δοκιμές που παρουσιάζονται οι αγωνιστικές περίοδοι 2000/2001 – 2008/209.
Και σε αυτή την περίπτωση τα αποτελέσματα που προέκυψαν είναι πάρα πολλά έτσι ώστε να μπορώ να σας τα παρουσιάσω όλα. Έτσι, επέλεξα κάποια από αυτά τα οποία θεωρώ ότι έχουν ενδιαφέρον και άξιζαν αναφοράς.
Όπως και στα αποτελέσματα που προέκυψαν στο Radial Basis Function Network και παρουσιάστηκαν στο προηγούμενο υποκεφάλαιο, εξαιρούνται οι έξι πρώτοι αγώνες κάθε αγωνιστικής περιόδου, αφού κάθε χρονιά πρέπει να δημιουργηθεί ένα απαραίτητο αντιπροσωπευτικό δείγμα για το πώς συμπεριφέρεται η κάθε ομάδα την τρέχουσα περίοδο. Η αγωνιστική περίοδος στην οποία αναφερόμαστε είναι η περίοδος 2009/2010, χρονιά που δεν περιέχεται στο training set και για όλα τα παραδείγματα θα αποτελεί το testing set μας. Στις πιο κάτω δοκιμές έχω 468 αντί 480 που είχα στις δοκιμές του RBF. Ο λόγος είναι ότι δεν περιλαμβάνεται η τελευταία αγωνιστική του έτους 2009-2010 αφού όταν έκανα τις δοκιμές μου και κατέγραφα τα αποτελέσματα δεν είχε ολοκληρωθεί ακόμη σε αντίθεση με τα RBF τα οποία υπολόγισα 2 εβδομάδες μετά, όταν ήδη ολοκληρώθηκε το πρωτάθλημα.

**Σύνολο Δεδομένων 1**
**SVM Type : C-SVC (cost 1 , Wi -w1 1 -w-1 1)**
**Kernel Type : Radial Basis (gamma 0,5)**

Accuracy = 61.53846153846154% (288/468) (classification)
Mean squared error = 1.5384615384615385 (regression)
Squared correlation coefficient = 0.07596463675953705 (regression)
Όπως βλέπουμε, σε σύνολο 468 αγώνων έχουμε 288 επιτυχίες, δηλαδή ποσοστό επιτυχίας **61,53%.** Όσον αφορά το στοιχηματικό κέρδος και κατ' επέκταση την αύξηση του αρχικού κεφαλαίου το οποίο θα επενδύαμε, αυτό θα ήταν της τάξεως του **13,83%** αν θεωρήσουμε το 1,85 ως την μέση απόδοση που θα μας προσφέρεται για κάθε αγώνα, όπως δηλαδή κάναμε και προηγουμένως.

**Σύνολο Δεδομένων 2**
**SVM Type : C-SVC (cost 1 , Wi -w1 1 -w-1 1)**
**Kernel Type : Radial Basis (gamma 0,9)**
Σε αυτό το παράδειγμα αυξήσαμε το gamma σε 0,9 από 0,5 που ήταν προηγουμένως. Τα αποτελέσματα ήταν εμφανώς καλύτερα.
Accuracy = 68.37606837606837% (320/468) (classification)
Mean squared error = 1.264957264957265 (regression)
Squared correlation coefficient = 0.1622057320625913 (regression)

Όπως βλέπουμε, σε σύνολο 468 αγώνων έχουμε 320 επιτυχίες, δηλαδή ποσοστό επιτυχίας **68,37%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **26,48%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

**Σύνολο Δεδομένων 3**
**SVM Type : C-SVC (cost 1 , Wi -w1 1 -w-1 1)**
**Kernel Type : Radial Basis (gamma 0,2)**
Εδώ δοκιμάσαμε να μειώσουμε το gamma σε 0,2 έτσι ώστε να βεβαιωθούμε ότι δεν ήταν τυχαία η βελτίωση που παρατηρήθηκε προηγουμένως με την αύξηση της συγκεκριμένης μεταβλητής. Η πρόβλεψή μας για χειρότερα αποτελέσματα επαληθεύτηκε.
Accuracy = 54.91452991452992% (257/468) (classification)
Mean squared error = 1.8034188034188035 (regression)
Squared correlation coefficient = NaN (regression)

Όπως βλέπουμε, σε σύνολο 468 αγώνων έχουμε μόνο 257 επιτυχίες, δηλαδή ποσοστό επιτυχίας **54,91%.** Το στοιχηματικό κέρδος θα ήταν της τάξεως του **1,58%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

**Σύνολο Δεδομένων 4**

**SVM Type : C-SVC (cost 2 , Wi -w1 1 -w-1 1)**

**Kernel Type : Radial Basis (gamma 0,9)**

Εδώ δοκιμάσαμε να αυξήσουμε το cost και να διατηρήσουμε το gamma σταθερό. Τα αποτελέσματα βελτιώθηκαν.

Accuracy = 69.87179487179486% (327/468) (classification)

Mean squared error = 1.205128205128205 (regression)

Squared correlation coefficient = 0.15167420489044703 (regression)

Όπως βλέπουμε, σε σύνολο 468 αγώνων έχουμε 327 επιτυχίες, δηλαδή ποσοστό επιτυχίας **69,87%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **29,25%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

**Σύνολο Δεδομένων 5**

**SVM Type : C-SVC (cost 5 , Wi -w1 1 -w-1 1)**

**Kernel Type : Radial Basis (gamma 0,9)**

Εδώ δοκιμάσαμε να αυξήσουμε το cost ακόμη περισσότερο και να διατηρήσουμε το gamma σταθερό. Τα αποτελέσματα ήταν περίπου τα ίδια με τα προηγούμενα, ελαφρώς χειρότερα.

Accuracy = 69.01709401709401% (323/468) (classification)

Mean squared error = 1.2393162393162394 (regression)

Squared correlation coefficient = 0.13633322479248194 (regression)

Όπως βλέπουμε, σε σύνολο 468 αγώνων έχουμε 323 επιτυχίες, δηλαδή ποσοστό επιτυχίας **69,01%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **27,66%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

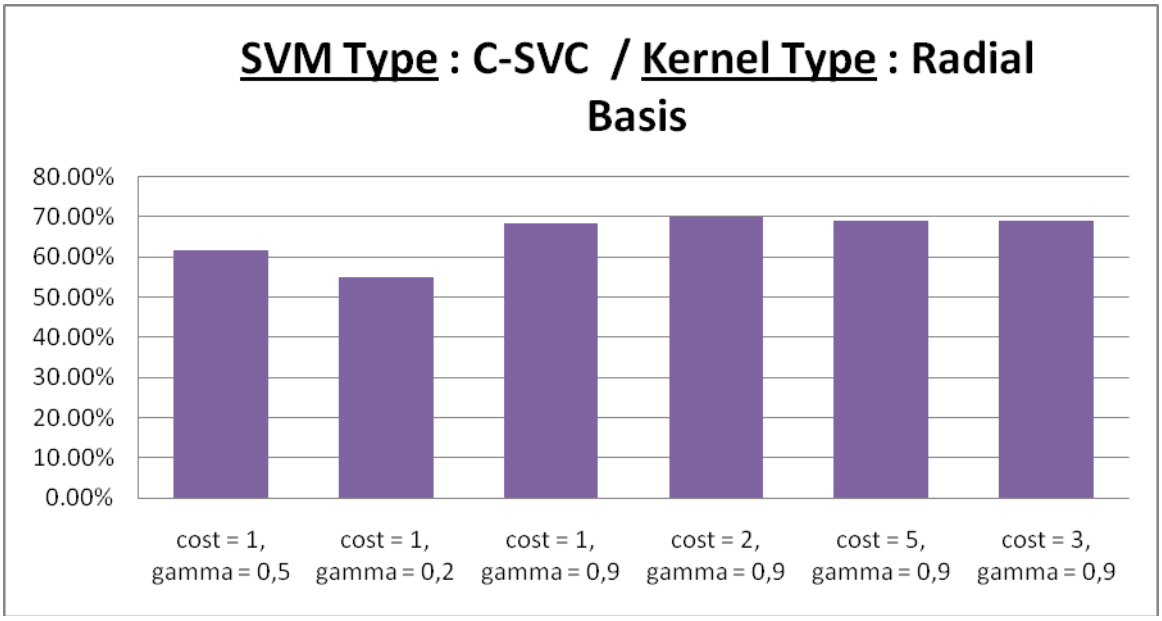
**Σύνολο Δεδομένων 6**

**SVM Type : C-SVC (cost 3 , Wi -w1 1 -w-1 1)**

**Kernel Type : Radial Basis (gamma 0,9)**

Εδώ δοκιμάσαμε να επαναφέρουμε το gamma σε 0,9 και να μειώσουμε το cost σε 3 αντί 5. Τα αποτελέσματα ήταν ακριβώς τα ίδια με το προηγούμενο παράδειγμα.

Accuracy = 69.01709401709401% (323/468) (classification)
Mean squared error = 1.2393162393162394 (regression)
Squared correlation coefficient = 0.13633322479248194 (regression)




Στην πιο πάνω γραφική φαίνονται συγκεντρωμένα τα αποτελέσματα που αναφέραμε. Παρατηρούμε ότι τα καλύτερα αποτελέσματα σε αυτό τον συνδυασμό SVM Type και Kernel Type προέκυψαν με cost = 2 και gamma = 9. Το ποσοστό επιτυχίας σε εκείνη την περίπτωση ανήλθε στο **69,87%.**

Στα πιο κάτω σύνολα δοκίμασα σαν Kernel Type τον Linear. Τα αποτελέσματα δεν ήταν καθόλου καλά και δεν θα τα σχολιάσω καθόλου. Απλά αναφέρονται για σκοπούς σύγκρισης.

**Σύνολο Δεδομένων 7**

**SVM Type : C-SVC (cost 1 , Wi -w1 1 -w-1 1)**

**Kernel Type : Linear**

Accuracy = 54.91452991452992% (257/468) (classification)

Mean squared error = 1.8034188034188035 (regression)

Squared correlation coefficient = NaN (regression)

**Σύνολο Δεδομένων 8**

**SVM Type : C-SVC (cost 3 , Wi -w1 1 -w-1 1)**

**Kernel Type : Linear**

Accuracy = 55.12820512820513% (258/468) (classification)

Mean squared error = 1.794871794871795 (regression)

Squared correlation coefficient = 0.002608157341912175 (regression)

**Σύνολο Δεδομένων 8**

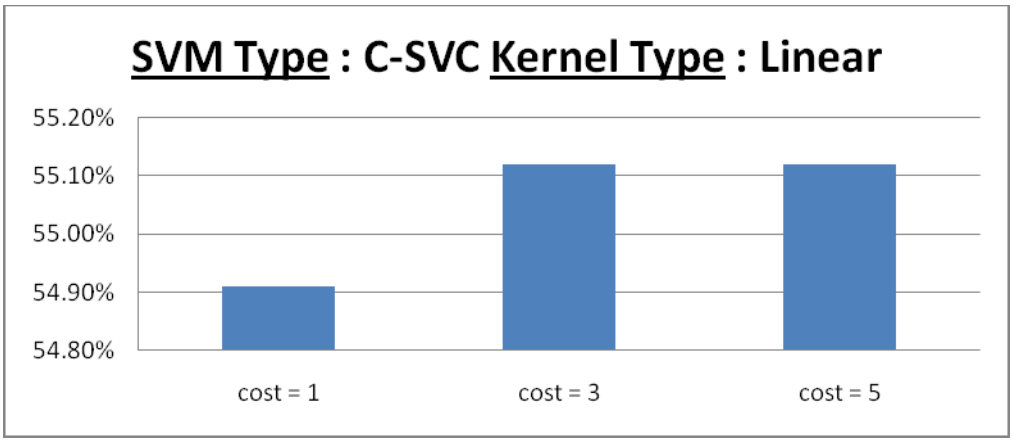
**SVM Type : C-SVC (cost 5 , Wi -w1 1 -w-1 1)**

**Kernel Type : Linear**

Accuracy = 55.12820512820513% (258/468) (classification)

Mean squared error = 1.794871794871795 (regression)

Squared correlation coefficient = 0.002608157341912175 (regression)



**Σύνολο Δεδομένων 9**

**SVM Type : C-SVC (cost 1 , Wi -w1 1 -w-1 1)**

**Kernel Type : Sigmoid (gamma = 0.5 , coef0 = 0)**

Accuracy = 54.91452991452992% (257/468) (classification)

Mean squared error = 1.8034188034188035 (regression)

Squared correlation coefficient = NaN (regression)

**Σύνολο Δεδομένων 10**

**SVM Type : C-SVC (cost 1 , Wi -w1 1 -w-1 1)**

**Kernel Type : Sigmoid (gamma = 0.9 , coef0 = 0)**

Accuracy = 54.91452991452992% (257/468) (classification)

Mean squared error = 1.8034188034188035 (regression)

Squared correlation coefficient = NaN (regression)

**Σύνολο Δεδομένων 11**

**SVM Type : C-SVC (cost 9 , Wi -w1 1 -w-1 1)**

**Kernel Type : Sigmoid (gamma = 0.9 , coef0 = 1)**

Accuracy = 54.91452991452992% (257/468) (classification)

Mean squared error = 1.8034188034188035 (regression)

Squared correlation coefficient = NaN (regression)

**Σύνολο Δεδομένων 12**

**SVM Type : C-SVC (cost 9 , Wi -w1 1 -w-1 1)**

**Kernel Type : Sigmoid (gamma = 0.9 , coef0 = 4)**

Accuracy = 54.91452991452992% (257/468) (classification)

Mean squared error = 1.8034188034188035 (regression)

Squared correlation coefficient = NaN (regression)

Στα πιο πάνω παραδείγματα χρησιμοποίησα τον Sigmoid σαν Kernel Type. Ότι παραμέτρους και αν άλλαζα όμως τα αποτελέσματα παρέμεναν τα ίδια με πολύ χαμηλό ποσοστό μάλιστα. Έτσι αποφάσισα ότι ο συγκεκριμένος τύπος Kernel δεν μπορεί να προσφέρει στο πρόβλημα μας.

Στα πιο κάτω παραδείγματα χρησιμοποίησα τον nu - SVC σαν SVM Type.

**Σύνολο Δεδομένων 13**

**SVM Type : nu-SVC (nu 0,5 , Wi -w1 1 -w-1 1)**

**Kernel Type : Radial Basis (gamma 0,5)**

Accuracy = 67.3076923076923% (315/468) (classification)

Mean squared error = 1.3076923076923077 (regression)

Squared correlation coefficient = 0.11355691063280977 (regression)

Σε σύνολο 468 αγώνων έχουμε 315 επιτυχίες, δηλαδή ποσοστό επιτυχίας **67,30%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **24,50%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

**Σύνολο Δεδομένων 14**

**SVM Type : nu-SVC (nu 0,5 , Wi -w1 1 -w-1 1)**

**Kernel Type : Radial Basis (gamma 0,9)**

Accuracy = 69.65811965811966% (326/468) (classification)

Mean squared error = 1.2136752136752136 (regression)

Squared correlation coefficient = 0.14768136626056932 (regression)

Σε σύνολο 468 αγώνων έχουμε 326 επιτυχίες, δηλαδή ποσοστό επιτυχίας **69,65%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **28,85%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

**Σύνολο Δεδομένων 15 – Καλύτερα αποτελέσματα**

**SVM Type : nu-SVC (nu 0,5 , Wi -w1 1 -w-1 1)**

**Kernel Type : Radial Basis (gamma 1,5)**

Accuracy = 70.2991452991453% (329/468) (classification)

Mean squared error = 1.188034188034188 (regression)

Squared correlation coefficient = 0.15710219048672244 (regression)

Σε σύνολο 468 αγώνων έχουμε 329 επιτυχίες, δηλαδή ποσοστό επιτυχίας **70,29%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **30%** με 1,85 ως μέση απόδοση για κάθε αγώνα. Το ποσοστό αυτό ήταν το καλύτερο που επιτεύχθηκε μέχρι στιγμής με την χρήση τόσο του SVM simulator αλλά και του Radial Basis Function network.

**Σύνολο Δεδομένων 16**

**SVM Type : nu-SVC (nu 0,5 , Wi -w1 1 -w-1 1)**
**Kernel Type : Radial Basis (gamma 2)**
Accuracy = 69.65811965811966% (326/468) (classification)
Mean squared error = 1.2136752136752136 (regression)
Squared correlation coefficient = 0.15035863601566654 (regression)

Σε σύνολο 468 αγώνων έχουμε 326 επιτυχίες, δηλαδή ποσοστό επιτυχίας **69,65%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **28,85%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

**Σύνολο Δεδομένων 17**

Σε αυτή την δοκιμή επιχείρησα να μεταβάλω τη μεταβλητή nu από 0,5 σε 0,9.
**SVM Type : nu-SVC (nu 0,9 , Wi -w1 1 -w-1 1)**
**Kernel Type : Radial Basis (gamma 1,5)**
Accuracy = 69.65811965811966% (326/468) (classification)
Mean squared error = 1.2136752136752136 (regression)
Squared correlation coefficient = 0.15035863601566654 (regression)

Σε σύνολο 468 αγώνων έχουμε και εδώ 326 επιτυχίες, δηλαδή ποσοστό επιτυχίας **69,65%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **28,85%** με 1,85 ως μέση απόδοση για κάθε αγώνα.

**<u>Σύνολο Δεδομένων 18</u>**

Σε αυτή την δοκιμή επιχείρησα να μειώσω τη μεταβλητή nu σε 0,2

**<u>SVM Type</u> : nu-SVC (nu 0,2 , Wi -w1 1 -w-1 1)**

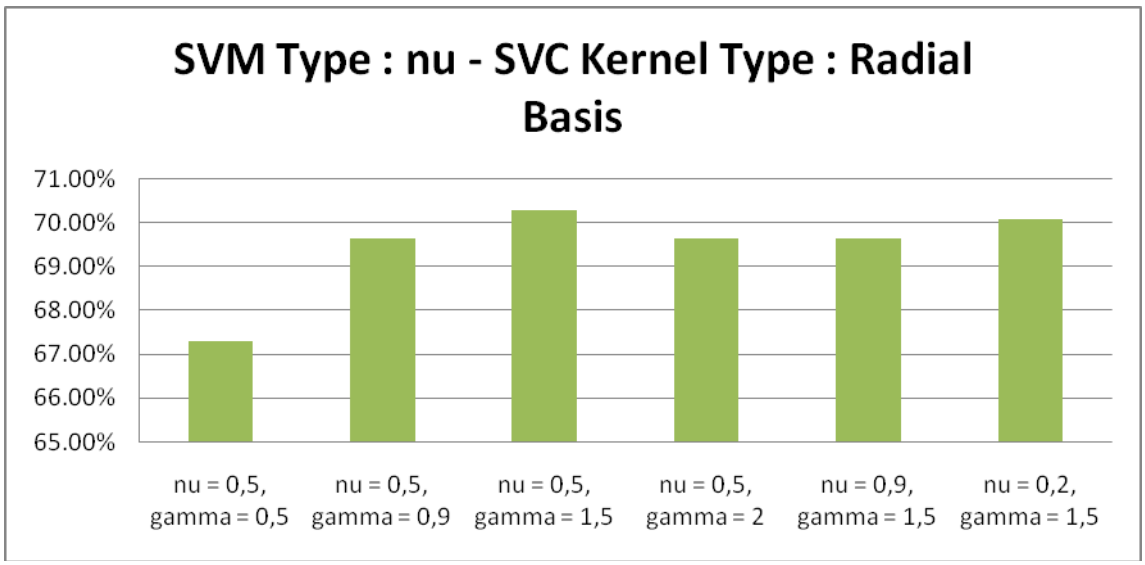
**<u>Kernel Type</u> : Radial Basis (gamma 1,5)**

Accuracy = 70.08547008547008% (328/468) (classification)

Mean squared error = 1.1965811965811965 (regression)

Squared correlation coefficient = 0.15349718746751917 (regression)

Σε σύνολο 468 αγώνων έχουμε και 328 επιτυχίες, δηλαδή ποσοστό επιτυχίας **70,08%.** Όσον αφορά το στοιχηματικό κέρδος αυτό θα ήταν της τάξεως του **29,64%** με 1,85 ως μέση απόδοση για κάθε αγώνα.




Στην πιο πάνω γραφική φαίνονται συγκεντρωμένα τα αποτελέσματα που αναφέραμε. Παρατηρούμε ότι τα καλύτερα αποτελέσματα σε αυτό τον συνδυασμό SVM Type και Kernel Type προέκυψαν με nu = 0,5 και gamma = 1,5. Το ποσοστό επιτυχίας σε εκείνη την περίπτωση ανήλθε στο **70,29%.**

# Κεφάλαιο 8

## Συμπεράσματα και μελλοντική εργασία

8.1 Γενικά συμπεράσματα……..……...……..... 93
8.2 Μελλοντική εργασία……...……..... 99

### 8.1 Γενικά συμπεράσματα

Σε αυτό το κεφάλαιο θα προσπαθήσω περιληπτικά να επισημάνω τα πιο σημαντικά συμπεράσματα που προέκυψαν από την εργασία αυτή, τι σημαίνουν τα αποτελέσματα που επιτεύχθηκαν για την ίδια την έρευνα και τι πιστεύω ότι μπορεί να γίνει στο μέλλον έτσι ώστε η δουλειά αυτή να επεκταθεί και να αποτελέσει τη βάση για έρευνες που θα αποφέρουν ακόμη καλύτερα αποτελέσματα.

Κατ' αρχάς, πιστεύω ότι η επιλογή της πρόβλεψης του συνολικού αριθμού τερμάτων που πρόκειται να σημειωθούν σε ένα παιχνίδι και κατ' επέκταση η χρησιμοποίηση του στο είδος στοιχήματος OVER 2,5 - UNDER 2,5 ήταν απόλυτα επιτυχημένη. Αυτό φάνηκε ξεκάθαρα από τα αποτελέσματα που προέκυψαν στο τέλος της εργασίας που είναι μακράν καλύτερα από οτιδήποτε είχε επιτευχθεί στο παρελθόν στις υπόλοιπες κατηγορίες στοιχημάτων στις οποίες επιχειρήθηκε, όπως για παράδειγμα την πρόβλεψη του νικητή. Επίσης, εκτός από τα μεγαλύτερα ποσοστά επιτυχίας που πετύχαμε με την επιλογή αυτού του είδους στοιχήματος είχαμε και αρκετά μεγάλο ποσοστό αύξησης του αρχικού μας κεφαλαίου, ή καλύτερα μεγαλύτερο κέρδος. Και σε αυτή την περίπτωση έπαιξε ρόλο το είδος του στοιχήματος στο οποίο επιλέξαμε να επικεντρωθούμε αφού η μικρότερη απόδοση που συναντούμε στο συγκεκριμένο πρωτάθλημα για την κατηγορία αυτή στοιχημάτων είναι 1,65 σε αντίθεση με την επιλογή του νικητή που αν για παράδειγμα η μία ομάδα είναι το απόλυτο φαβορί και αγωνίζεται στην έδρα της μπορεί η απόδοση της να μην ξεπερνά το 1,10. Σε αυτή την περίπτωση έστω και αν η πρόβλεψη είναι σωστή για το συγκεκριμένο αγώνα, δεν μου προσφέρει τίποτα ουσιαστικά στην προσπάθεια για αύξηση του αρχικού κεφαλαίου.

Επίσης, θεωρώ ότι η επιλογή του πρωταθλήματος της Championship για την εκπαίδευση του δικτύου και την πρόβλεψη αποτελεσμάτων στη συγκεκριμένη κατηγορία στοιχήματος ήταν απόλυτα επιτυχημένη. Το εν λόγω πρωτάθλημα με τον μεγάλο αριθμό αγωνιστικών αλλά και ομάδων από τις οποίες αποτελείται προσφέρεται για τον συγκεκριμένο σκοπό αφού μας προσφέρει πληθώρα αγώνων για πρόβλεψη και εξαγωγή συμπερασμάτων τα οποία προκύπτουν από αυτά. Ακόμη θεωρείται ένα από τα πλέον αξιόπιστα πρωταθλήματα στον κόσμο κάτι που μαρτυρούν τόσο οι αποδόσεις που προσφέρονται για τους αγώνες του από τις εταιρείες στοιχημάτων όσο και το γεγονός ότι κανένας αγώνας δεν αφαιρέθηκε ποτέ από το κουπόνι του στοιχήματος λόγω προκατάληψης για "περίεργα αποτελέσματα", κάτι που συμβαίνει αρκετά συχνά σε άλλα πρωταθλήματα. Τέλος επειδή είναι αρκετά γνωστό πρωτάθλημα και προκαλεί το ενδιαφέρον των τζογαδόρων, το ιστορικό των αγώνων υπάρχει και είναι εύκολα προσπελάσιμο για πολλά χρόνια πριν από διάφορες πηγές, κυρίως στο διαδίκτυο.

Ακόμη, ένας πολύ σημαντικός παράγοντας στην όλη διαδικασία είναι τα σύνολα δεδομένων εισόδου που δόθηκαν στο δίκτυο για την εκπαίδευση του αλλά και για την χρησιμοποίηση τους για τις προβλέψεις του. Όπως ανέφερα και στο κεφάλαιο στο οποίο αναλύονται τα αποτελέσματα, έχω καταλήξει ότι τα καλύτερα αποτελέσματα προκύπτουν με τη χρησιμοποίηση του διανύσματος εισόδου 23 διαστάσεων για την κάθε ομάδα. Αν και συμφωνώ με την άποψη ότι περισσότερη σημασία έχει η τρέχουσα αγωνιστική κατάσταση της ομάδας η οποία προκύπτει από την αγωνιστική συμπεριφορά της τους τελευταίους πέντε με έξι αγώνες, εντούτοις από την μεταβολή της δομής του δικτύου δια μέσου της εκπαίδευσης του φαίνεται ξεκάθαρα ότι είναι σε θέση να αποφασίσει το ίδιο σε ποιες από τις παραμέτρους που του δόθηκαν θα επικεντρωθεί και ποιες θα λάβει λιγότερο υπ' όψην. Εξάλλου αυτό είναι ένα από τα πλεονεκτήματα των νευρωνικών δικτύων, ότι έχουν την δυνατότητα να ανέχονται τα λάθη και να απομονώνουν την άσχετη πληροφορία όπως ακριβώς και οι κανονικοί εγκέφαλοι. Το αρνητικό σε αυτή την περίπτωση είναι ότι αυξάνεται λίγο η πολυπλοκότητα του δικτύου αλλά αφού παράγονται καλά αποτελέσματα δεν υπάρχει λόγος να μας απασχολεί το θέμα αυτό. Σε δοκιμές που έκανα πάντως με μικρότερα διανύσματα εισόδου τα οποία επικεντρώνονταν κυρίως στην αγωνιστική συμπεριφορά της ομάδας τους τελευταίους πέντε με έξι αγώνες , τα αποτελέσματα που προέκυψαν δεν ήταν τόσο καλά αφού κατά πάσα πιθανότητα δεν μπορούσαν να σκιαγραφήσουν

την συνολική εικόνα της ομάδας και τον τρόπο με τον οποίο θα συμπεριφερόταν στον προσεχή της αγώνα ο οποίος μας ενδιέφερε.

Όσο αφορά το δίκτυο μας και τις παραμέτρους οι οποίες το καθορίζουν, αυτό που εμπειρικά προέκυψε μέσα από τις πολλές δοκιμές που επιχείρησα είναι ότι δεν υπάρχει κάποιος κανόνας που μπορούμε να αναφέρουμε ή κάποιο ασφαλές συμπέρασμα στο οποίο εξάχθηκε σχετικά με αυτές. Αυτό που μπορώ με βεβαιότητα να επισημάνω είναι η εξαιρετικά μεγάλη ευαισθησία του δικτύου σε οποιαδήποτε αλλαγή επιχειρηθεί σε αυτό. Πιο συγκεκριμένα, μια μικρή μεταβολή στο ρυθμό μάθησης ίσως μεταβάλει την δομή του δικτύου και κατ' επέκταση τα αποτελέσματα στον μέγιστο βαθμό. Το ίδιο ισχύει και για τις υπόλοιπες μεταβλητές που το αποτελούν αλλά και για την ίδια την δομή του δικτύου όπως ο αριθμός ων κρυφών νευρώνων και το μέγεθος των διανυσμάτων εισόδου. Επίσης μεγάλο ρόλο παίζει ο αριθμός των εποχών που εκπαιδεύεται το δίκτυο. Μικρός αριθμός εποχών ίσως να μην φτάνουν για το δίκτυο ώστε να αφομοιώσει τη συμπεριφορά των ομάδων σε σχέση με τα δεδομένα τα οποία του δίνονται, ενώ μεγαλύτερος από το κανονικό αριθμός εποχών ενδεχομένως να επιφέρει αντίθετα αποτελέσματα αναγκάζοντας το δίκτυο να ανακαλύψει συσχετίσεις μεταξύ των δεδομένων εισόδου και επιθυμητών αποτελεσμάτων οι οποίες δεν ισχύουν στη πραγματικότητα.

Ακόμη πολλές φορές πρέπει να βρεθεί ο σωστός συνδυασμός των τιμών των διαφόρων παραμέτρων. Για παράδειγμα, αν με ρυθμό μάθησης 0.002 και αριθμό εποχών 800 επιτευχθεί ένα συγκεκριμένο ποσοστό επιτυχίας, εντούτοις αν ο δοκιμάσουμε να αλλάξουμε τον αριθμό εποχών σε 1000 ίσως να πρέπει να αλλάξει και ο ρυθμός μάθησης για να βελτιωθούν τα αποτελέσματα μας. Αν σε αυτά προσθέσουμε και τον αριθμό των ποδοσφαιρικών περιόδων στις οποίες εκπαιδεύεται το δίκτυο, τον αριθμό των διαστάσεων του διανύσματος εισόδου αλλά και τον αριθμό των κρυφών νευρώνων, εύκολα μπορεί κανείς να αντιληφθεί ότι οι συνδυασμοί είναι πάρα πολλοί και η διαδικασία εύρεσης του ενός που θα αποφέρει τα καλύτερα αποτελέσματα είναι πολύ χρονοβόρα αφού μπορεί να γίνει μόνο με την πρακτική "της *δοκιμής και* του λάθους". Όσον αφορά την δική μας περίπτωση, πιστεύω ότι εξάντλησα αρκετά μεγάλο μέρος των συνδυασμών αυτών, στα πλαίσια πάντοτε των πεδίων τιμών που θεωρούσα λογικές με την εξοικείωση που απέκτησα αναφορικά με το συγκεκριμένο δίκτυο.

Μια άλλη παράμετρος η οποία οδήγησε σε βελτίωση των αποτελεσμάτων του δικτύου μου με όλες τις υπόλοιπες να παραμένουν σταθερές ήταν ο αριθμός των ποδοσφαιρικών

περιόδων στις οποίες εκπαιδεύεται το δίκτυο. Στα πρώτες βδομάδες τις οποίες πραγματοποίησα τις δοκιμές μου χρησιμοποιούσα για εκπαίδευση τις τελευταίες 4 περιόδους που προηγήθηκαν αυτής στην οποία θα έλεγχα τα αποτελέσματα μου. Συγκεκριμένα χρησιμοποιούσα τις περιόδους 2004/2005 – 2007/2008 για την εκπαίδευση του δικτύου μου και την περίοδο 2008/2009 για τον έλεγχο των αποτελεσμάτων. Αυτό το έκανα γιατί θεωρούσα ότι αν έδινα στο δίκτυο πολλές περιόδους για να εκπαιδευτεί σε αυτές, δεν θα μπορούσε να βρει εύκολα την συσχέτιση μεταξύ των δεδομένων εισόδου και των επιθυμητών αποτελεσμάτων. Πίστευα ότι έτσι θα το ανάγκαζα να "μάθει" συμπεριφορές ομάδων που στην ουσία αγωνίζονταν σε διαφορετικό πρωτάθλημα, αφού η πάροδος τόσων πολλών χρόνων και η εξέλιξη του ποδοσφαίρου όλο αυτό το διάστημα προκάλεσαν αυτή την αλλαγή. Εντούτοις στα πλαίσια των πολλών δοκιμών που επιχείρησα, έδωσα στο δίκτυο μου τα αποτελέσματα διπλάσιων περιόδων για την εκπαίδευση του και πιο συγκεκριμένα τις περιόδους 2000/2001 – 2007/2008. Ενώ περίμενα ότι τα αποτελέσματα δεν θα ήταν αρκετά ικανοποιητικά, προς έκπληξη μου ανακάλυψα ότι αυτό επέφερε σημαντική βελτίωση τους. Προσπαθώντας να εξηγήσω την βελτίωση αυτή κατέληξα στο συμπέρασμα ότι αυτό συμβαίνει για δυο λόγους. Πρώτο οφείλεται στο πρωτάθλημα το οποίο χρησιμοποίησα το οποίο με το πέρασμα των χρόνων δεν αλλοιώθηκε και οι συμπεριφορές των ομάδων που το αποτελούν δεν άλλαξαν σε μεγάλο βαθμό. Δεύτερος λόγος κατά τη άποψη μου είναι το γεγονός ότι δίνοντας του περισσότερες περιόδους για να εκπαιδευτεί του έδωσα την δυνατότητα να έχει πιο ομαλές μεταβολές των παραμέτρων του αφού είχε μεγαλύτερη ποικιλία αποτελεσμάτων να εκπαιδευτεί και έγινε πιο ανεκτικό στο θόρυβο, δηλαδή τα περίεργα αποτελέσματα τα οποία του προκαλούσαν δραματικές αλλαγές προς το χειρότερο. Αυτό γιατί αφού αυξήθηκε ο αριθμός των παιχνιδιών, μου δόθηκε η ευκαιρία να μειώσω τον αριθμό των εποχών για κάθε περίοδο καθώς επίσης και τον ρυθμό μάθησης. Έτσι, το δίκτυο μου συναντούσε λιγότερες φορές ένα "περίεργο" αποτέλεσμα (όπως για παράδειγμα ένα παιχνίδι στο οποίο σημειώθηκαν 13 τέρματα) και η αλλαγή που γινόταν σε αυτό ήταν μικρότερη λόγω του μειωμένου ρυθμού μάθησης.

Όσον αφορά τα αποτελέσματα καθ' αυτά, με την χρήση του Radial Basis Function Network επιτεύχθηκε ποσοστό 68,33% το οποίο υπολογίζεται ότι θα αποφέρει κέρδος της τάξεως του 26,41% με 1,85 ως μέση απόδοση για κάθε αγώνα. Το ποσοστό αυτό είναι το καλύτερο που έχει επιτευχθεί μέχρι στιγμής, αναφερόμενος πάντοτε στις 4

προηγούμενες μελέτες που είχα μελετήσει και περιγράψει στο κεφάλαιο 1. Μάλιστα, η αύξηση αυτή είναι της τάξης του 6,33% η οποία στο είδος του προβλήματος μας είναι σημαντική, αφού το προηγούμενο καλύτερο ποσοστό σωστής πρόβλεψης ήταν αυτό του Νεοκλέους (2007) και ανερχόταν στο 62%. Αυτό αποδεικνύει και το ποσοστό αύξησης του αρχικού κεφαλαίου το οποίο ανέρχεται στο 26,41% αντί 22% που είχε επιτευχθεί προηγουμένως. Όπως έχω αναφέρει όμως και σε προηγούμενο κεφάλαιο, το ποσοστό αυτό είχε υπολογιστεί με μέσο όρο απόδοσης ανά παιχνίδι το 2 ενώ στην δική μας περίπτωση το ποσοστό κέρδους σε σχέση με το αρχικό κεφάλαιο υπολογίστηκε με μέσο όρο απόδοσης το 1,85. Γι' αυτό τον λόγο μπορεί να φαίνεται μικρό αλλά στην πραγματικότητα αποτελεί ουσιαστικότατη βελτίωση.

Τέλος, με την χρήση του Support Vector Machine simulator επιτεύχθηκε ποσοστό επιτυχίας 70,29% και στοιχηματικό κέρδος της τάξεως του 30%. Το ποσοστό αυτό ήταν ακόμη καλύτερο και από αυτό που επιτεύχθηκε με την χρήση του Radial Basis Function network αποδεικνύοντας ότι με την χρήση των Support Vector Machines υπάρχει η δυνατότητα για περαιτέρω βελτίωση των αριθμών που ανέφερα πιο πάνω και κατ' επέκταση περισσότερο στοιχηματικό κέρδος που στην πραγματικότητα είναι το ζητούμενο.

Τελειώνοντας λοιπόν τη διπλωματική αυτή εργασία μπορώ να πω ότι προσωπικά αισθάνομαι πολύ ικανοποιημένος από την δουλειά που έχει γίνει και από τα αποτελέσματα που έχουν προκύψει. Η επίλυση του προβλήματος της πρόβλεψης ποδοσφαιρικών αποτελεσμάτων με την χρήση των Radial Basis Function νευρωνικών δικτύων έγινε με μεγάλη επιτυχία και έδωσε αποτελέσματα καλύτερα από αυτά που είχαν επιτευχθεί με την χρήση άλλων νευρωνικών δικτύων και μεθόδων. Ακόμη επιβεβαιώθηκε η άποψη – υποψία που θέλαμε να ελέγξουμε, ότι δηλαδή με την χρήση των SVM υπάρχει η δυνατότητα για να επιτευχθούν ακόμη καλύτερα αποτελέσματα στο συγκεκριμένο πρόβλημα. Οι δύο βασικοί στόχοι που είχαν τεθεί από την αρχή λοιπόν της εργασίας τους οποίους προανέφερα φαίνεται ότι επιτεύχθηκαν στο έπακρο.

## 8.2 Μελλοντική εργασία

Με το πέρας της διπλωματικής αυτής εργασίας λοιπόν και με βάση τα αποτελέσματα τα οποία προέκυψαν από αυτή, προκύπτει ότι πιθανών να υπάρχουν ακόμη περιθώρια βελτίωσης των αποτελεσμάτων και αύξησης των ποσοστών επιτυχίας στο ποδοσφαιρικό στοίχημα με την χρήση τόσο των νευρωνικών δικτύων αλλά και των Support Vector Machines. Την άποψη μου αυτή την εκφράζω με κάποια επιφύλαξη αφού δεν μπορούμε να αναφέρουμε με βεβαιότητα μέχρι που μπορούν να φτάσουν τα ποσοστά επιτυχίας και ούτε υπάρχει κάποιο όριο το οποίο να μπορέσουμε να θέσουμε ως στόχο λόγω της φύσης το προβλήματος που καλούμαστε να επιλύσουμε. Το γεγονός όμως ότι κάθε προσπάθεια που γινόταν στο παρελθόν απέφερε καλύτερα αποτελέσματα από τις προηγούμενες, κάτι που συνέβηκε και στην δική μας περίπτωση με κάνει να αισθάνομαι αισιόδοξος ότι η συγκεκριμένη έρευνα δεν έχει φτάσε ακόμη στα "όρια" της. Δυστυχώς λόγω του περιορισμένου χρονικού διαστήματος που είχα στη διάθεσή μου για την ολοκλήρωση και παράδοση της εργασίας αυτής δεν μου δόθηκε η δυνατότητα να προσπαθήσω ο ίδιος να αποδείξω τι πραγματικά ισχύει αλλά πιστεύω ότι πολύ σύντομα στο μέλλον θα γίνει και αυτό, φτάνει να συνεχιστεί η έρευνα αυτή και να μην σταματήσει η προσπάθεια. Τελειώνοντας λοιπόν, και λαμβάνοντας υπόψη τις εμπειρίες τις οποίες αποκόμισα στην όλη προσπάθεια που έγινε όσον αφορά το συγκεκριμένο πρόβλημα θα αναφέρω κάποιες προτάσεις που πιστεύω ότι αξίζει να εξεταστούν έτσι ώστε να δοθεί μια συνέχεια στην δουλειά που έχει ήδη γίνει.

Το πρώτο που έχω να προτείνω είναι η δημιουργία ενός Support Vector Machine αποκλειστικά για το συγκεκριμένο πρόβλημα. Μπορεί ο SVM simulator που χρησιμοποιήθηκε να αποδείχτηκε σε προηγούμενες μελέτες ότι είναι από τους καλύτερους που υπάρχουν διαθέσιμοι στο διαδίκτυο, αλλά πιστεύω ότι θα μπορούσε να δημιουργηθεί ένα Support Vector Machine από την αρχή και αποκλειστικά για το συγκεκριμένο πρόβλημα έτσι ώστε να είναι κάπως πιο ειδικό και προσαρμοσμένο στις συγκεκριμένες ανάγκες του προβλήματος αυτού. Μια τέτοια προσπάθεια θα έδινε στο άτομο που θα την επιχειρούσε την δυνατότητα να καταλάβει τα Support Vector Machines ακόμη καλύτερα και να εξοικειωθεί περισσότερο μαζί τους αφού θα έφτιαχνε ένα από την αρχή, καθώς επίσης θα του έδινε την δυνατότητα να το προσαρμόσει στο πρόβλημα μας. Αν θεωρήσουμε ως δεδομένο ότι με την χρήση των SVM υπάρχουν

δυνατότητες για εξασφάλιση καλύτερων αποτελεσμάτων στο συγκεκριμένο πρόβλημα, αφού κάτι τέτοιο είναι γενικότερα αποδεκτό για τα SVM, τότε πιστεύω πως αξίζει τον κόπο να γίνει η προσπάθεια αυτή.

Επίσης θα μπορούσε να γίνει προσπάθεια για μεταβολή των παραγόντων που καθορίζουν ένα νευρωνικό δίκτυο όπως για παράδειγμα ο ρυθμός μάθησης, η ορμή, ο αριθμός των επαναλήψεων, τα Data Sets κτλ. Επίσης θα μπορούσε να καθοριστεί ένα άλλο πρωτάθλημα στο οποίο η αγωνιστική συμπεριφορά των ομάδων να μην είναι τόσο απρόβλεπτη. Προσωπική μου άποψη είναι ότι τα περιθώρια βελτίωσης στα ποσοστά επιτυχίας χρησιμοποιώντας τις συγκεκριμένες τεχνικές έχουν στενέψει αφού τα Data Sets μεταβλήθηκαν με κάθε δυνατό τρόπο ενώ το ίδιο έγινε και με την δομή του δικτύου. Όσον αφορά το πρωτάθλημα ίσως να έχουμε καλύτερα ποσοστά επιτυχίας αλλά πιστεύω ότι θα έχουμε πολύ μικρότερο κέρδος αφού οι τιμές που θα προσφέρονται θα είναι πολύ πιο χαμηλές λόγω της συμπεριφοράς αυτής. Αυτές όμως οι απόψεις δεν μπορούν να αποδειχθούν προς το παρόν ούτε μαθηματικά ούτε με κάποιο άλλο τρόπο έτσι οι προτάσεις αυτές δεν μπορούν να πάψουν από το να αποτελούν επιλογές.

Τέλος, θα μπορούσε να δημιουργηθεί ένα RBF δίκτυο στο οποίο να δίνεται σαν έξοδος και σαν επιθυμητό αποτέλεσμα η κατηγορία (More ή Less) στην οποία ανήκει ένα συγκεκριμένο παιχνίδι και όχι ο ακριβής αριθμός τερμάτων που θα σημειωθούν σε αυτό όπως επιχειρήθηκε στην παρούσα διπλωματική εργασία. Με αυτό τον τρόπο θα ζητείται από το δίκτυο κάτι πολύ πιο απλό από αυτό που ζητούμε τώρα αφού θα πρέπει απλά να κάνει μια κατηγοριοποίηση μεταξύ μόλις δύο κατηγοριών. Φυσικά το ίδιο θα συμβαίνει και με τα δεδομένα που θα δέχεται αφού η πληροφορίες που θα παίρνει για ένα παιχνίδι θα είναι απλώς αν σε αυτό έχουν σημειωθεί περισσότερα ή λιγότερα από 2,5 τέρματα στερώντας του σημαντικά πλεονεκτήματα τα οποία είχε το δικό μας δίκτυο. Για παράδειγμα, μπορεί το δίκτυο να έχει μόνο να επιλέξει μεταξύ 2 επιλογών για ένα συγκεκριμένο παιχνίδι, αλλά όσον αφορά την εκπαίδευση του θα πάρει το ίδιο feedback για 2 παιχνίδια που προέβλεψε ότι θα τελείωναν less και τέλειωσαν 2-1 και 6-5. Δεν μπορώ να ξέρω τι είδους διαφοροποίηση θα προκαλέσει στα αποτελέσματα η συγκεκριμένη αλλαγή όμως θεωρώ ότι θα ήταν μια καλή ιδέα για μελλοντική υλοποίηση.

Ακόμη, μετά από εισήγηση του δευτέρου εξεταστή της εργασίας αυτής τέθηκε η πρόταση για να αυξηθούν οι νευρώνες στο output layer μεταβάλλοντας τον τρόπο που κωδικοποιείται η έξοδος του δικτύου. Σε αυτή την περίπτωση πρέπει να αλλάξει ολόκληρη η διαδικασία που εκπαιδεύεται το δίκτυο αφού πρέπει να γίνει με βάση το επιθυμητό output. Με την αλλαγή αυτή πιστεύεται ότι υπάρχει κάποια σημαντική πιθανότητα για να μεταβληθούν τα αποτελέσματα, με την πεποίθηση ότι θα είναι προς το καλύτερο, λόγω του ότι με την αύξηση των εξόδων (“fan out” του δικτύου) θα έχουμε μια πιο ευρεία κωδικοποίηση του αποτελέσματος με περισσότερα βάρη. Η αύξηση αυτή όμως πρέπει να γίνει με μέτρο και επιφύλαξη για αποφυγή τυχών υπερεκπαίδευσης. Δυστυχώς δεν είχα τον απαιτούμενο χρόνο για να προσπαθήσω να υλοποιήσω ο ίδιος την πολύ ενδιαφέρουσα αυτή εισήγηση και έτσι την καταγράφω και αυτή στις προτάσεις για μελλοντική εργασία.

## Βιβλιογραφία - Αναφορές


[1] Matt Jackson, MSc project, School of Computer Science and Information Systems, Birkbeck College, 2001

[2] Kevin Davies, MSc project, School of Computer Science and Information Systems, Birkbeck College, 2004

[3] Sian Turner, MSc project, School of Computer Science and Information Systems, Birkbeck College, 2005

[4] Θεοφάνης Νεοκλέους , Ατομική Διπλωματική Εργασία, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου 2007

[5] Υλικό Διαλέξεων Πληροφοριακών Συστημάτων Μάθησης (ΕΠΛ442), Πανεπιστήμιο Κύπρου, Δρ. Χρίστος Χριστοδούλου

[6] Κωνσταντίνος Διαμαντάρας, *Τεχνητά Νευρωνικά Δίκτυα*, Εκδόσεις Κλειδάριθμος, 2007

[7] Ειρήνη Γεωργίου, Χρήση των SVM's στη εκτίμηση τιμών ακινήτων, Ατομική Διπλωματική Εργασία, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου 2009

[8] Schölkopf Bernhard, Alex Smola, "Learning with Kernels", MIT Press, Cambridge, MA, 2002

[9] Goddard J. and Asimakopoulos I. Modelling Football Match Results and the Efficiency of Fixed-Odds Betting, Journal of Forecasting, 2004
http://stat-www.berkeley.edu/~aldous/157/Papers/goddard.pdf

[10] CHRISTOPHER J.C. BURGES, Bell Laboratories, Lucent Technologies "A Tutorial on Support Vector Machines for Pattern Recognition" http://www.telecom.tuc.gr/patreco/res/patreco0708/SVM_tutorial.pdf

[11] Nello Cristianini, BIOwulf Technologies, "Support Vector and Kernel Machines" http://www.telecom.tuc.gr/patreco/res/patreco0708/icml-tutorial.pdf

[12] Vapnik, V. (1979). Estimation of Dependences Based on Empirical Data [in Russian], Nauka: Moscow. (1982). English Translation: Springer Verlag: New York.

[13] Vapnik, V. (1995). The Nature of Statistical Learning Theory. Springer Verlag: New York.

# Παράρτημα Α - Υλοποίηση RBF Network

## Κλάση FileEditor. Java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;
import java.io.StreamTokenizer;

public class FileEditor {

	public static void main(String[] args) throws IOException

		String Currentteam = "MIDDLESBROUGH";
		int numberOfGames = 37;
		String team1 = "";
		String team2 = "";
		String s3 = "";
		int score1 = 0;
		int score2 = 0;
		String seasonYear = "2009";
		try {
			s3 = new java.io.File(".").getCanonicalPath();
		} catch (IOException e) {
			e.printStackTrace();
		}
		s3 = s3 + "\\data\\" + seasonYear + "\\TEMPSEASON.txt";

		Reader r = new BufferedReader(new FileReader(s3));
		StreamTokenizer stok = new StreamTokenizer(r);
		stok.parseNumbers();
		stok.nextToken();
		FileWriter fstream = new FileWriter(Currentteam + ".txt");
		BufferedWriter out = new BufferedWriter(fstream);

		for (int i = 0; i < numberOfGames * 12; i++) {
			team1 = stok.sval;
			stok.nextToken();
			team2 = stok.sval;
			stok.nextToken();
			score1 = (int) stok.nval;
			stok.nextToken();
			score2 = (int) stok.nval;
			stok.nextToken();
			if (Currentteam.equals(team1) ||
Currentteam.equals(team2)) {
				out.write(team1 + " " + team2 + " " + score1 +
" " + score2
					+ " " + " \n");
			}
		}
		out.close();
		}
}
```

## Κλάση HiddenLayerNode. Java

```
import java.io.BufferedWriter;

public class HiddenLayerNode {
	// public double R = 1;

	public double InputVector[];
	public double CentreVector[];
	public double output;
	public double gausianresult;
	public double eucleidianDistance;
	public double S; // Sigma
	public double N; // Learning Rate
	public double C; // Coefficient

	// public double error;

	HiddenLayerNode(double s, double c, double n) {
		CentreVector = new double[46];
		InputVector = new double[46];
		this.C = c;
		this.N = n;
		this.S = s;
	}

	double computeEucleidianDistance() {
		double distance = 0;
		for (int i = 0; i < 46; i++) {
			distance = distance
					+ Math.pow((this.InputVector[i] -
this.CentreVector[i]), 2);
		}
		distance = Math.sqrt(distance);
		eucleidianDistance = distance;
		return distance;

	}

	double GaussianFunction() {
		gausianresult = Math.exp((-1 *
Math.pow(eucleidianDistance, 2))
				/ Math.pow(S, 2));
		return gausianresult;
	}

	double computeOutput() {
		output = this.C * this.gausianresult;
		return output;
	}

	// double computeThisNeuronError(double epi8imito) {
	// this.error=epi8imito-this.output;
	// return this.error;
	// }

	public void changeC(double error, double output) {
		this.C = this.C + this.N * error * output;
	}
```

```
	public void changeCentre(double error) {
		double difference = 0;
		for (int i = 0; i < 46; i++) {
			difference = (this.N * this.C * error *
this.gausianresult * (this.InputVector[i] - this.CentreVector[i]))
					/ Math.pow(this.S, 2);
			this.CentreVector[i] = this.CentreVector[i] +
difference;
			/*
			 * if(this.CentreVector[i]<0){
this.CentreVector[i]=0; }
			 */
		}
	}

	public void changeS(double error) {
		this.S = this.S
				+ (this.N * this.C * error *
this.gausianresult * this.output)
				/ Math.pow(this.S, 3);

	}

	public void printCentre(BufferedWriter out) {
		try {

			out.write("\nCentre = ");
			for (int i = 0; i < 4; i++) {
				out.write(this.CentreVector[i] + " ");
			}
		} catch (Exception e) {// Catch exception if any
			System.err.println("Error: " + e.getMessage());
		}

	}
}
```

## Κλάση Match.java

```
public class Match {

	String homeTeam;
	String awayTeam;
	double goalsHomeTeam;
	double goalsAwayTeam;
	double epithimito;

	Match(String ht, String at,double gHT, double gAT ){
		this.homeTeam=ht;
		this.awayTeam=at;
		this.goalsHomeTeam=gHT;
		this.goalsAwayTeam=gAT;
		this.epithimito=this.goalsHomeTeam+this.goalsAwayTeam;

	}


}
```

## Κλάση OutputNode.java

```
public class OutputNode {
	public double input[];
	double sum = 0;
	double error;

	OutputNode() {
		input = new
double[TrainingProcedure.NUMBER_OF_HIDDEN_LAYER_NODES];
	}

	double computeSum() {
		this.sum=0;
		for (int i = 0; i <
TrainingProcedure.NUMBER_OF_HIDDEN_LAYER_NODES; i++) {
			this.sum = this.sum + input[i];
		}
		return sum;
	}

	double computeError(double epi8imito, double pragmatiko) {

		//error = Math.pow(epi8imito - pragmatiko, 2) / 2;
		error =epi8imito - pragmatiko;
		return error;

	}
}
```

## Κλάση Season.java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.io.StreamTokenizer;

public class Season {

	Team LeaqueTable[];
	Match Games[][];
	final int NUMBEROFROUNDS = 46;
	final int GAMESPERROUND = 12;
	String seasonYear="";

	Season(int numberOfTeams,String year) {
		LeaqueTable = new Team[numberOfTeams];
		Games = new Match[46][12];
		this.seasonYear=year;
	}

	void initializeTable(int numberOfTeams,int  numberofrounds)
throws IOException {

		String s3 = "";
```

```
		try {
			s3 = new java.io.File(".").getCanonicalPath();
		} catch (IOException e) {
			e.printStackTrace();
		}
		s3 = s3 + "\\data\\"+this.seasonYear+ "\\TEAMS.txt";

		Reader r = new BufferedReader(new FileReader(s3));
		StreamTokenizer stok = new StreamTokenizer(r);

		stok.parseNumbers();
		stok.nextToken();
		for (int i = 0; i < numberOfTeams; i++) {
			LeaqueTable[i]= new Team(stok.sval,this.seasonYear);
			LeaqueTable[i].createMatchTable(numberofrounds);
			LeaqueTable[i].computeStatisticsForHomeGames(6);

			LeaqueTable[i].computeStatisticsForAwayGames(6);

LeaqueTable[i].computeStatisticsForHomeGamesLastSixGames(6);

LeaqueTable[i].computeStatisticsForAwayGamesLastSixGames(6);
			LeaqueTable[i].computeTotalStatistics(6);
			stok.nextToken();
		}
}

void initializeGames(int numberOfRounds) throws IOException {
		String team1;
		String team2;
		String s3 = "";
		try {
			s3 = new java.io.File(".").getCanonicalPath();
		} catch (IOException e) {
			e.printStackTrace();
		}
		s3 = s3 + "\\data\\"+this.seasonYear+ "\\SEASON.txt";

		Reader r = new BufferedReader(new FileReader(s3));
		StreamTokenizer stok = new StreamTokenizer(r);

		stok.parseNumbers();
		stok.nextToken();
		for (int i = 0; i < 6 * 12 * 2; i++)
			stok.nextToken();

		for (int i = 0; i < numberOfRounds - 6; i++) {
			for (int j = 0; j < this.GAMESPERROUND; j++) {
				team1 = stok.sval;
				stok.nextToken();
				team2 = stok.sval;
				stok.nextToken();
				this.Games[i][j] = new Match(team1,team2,0,0);
			}
		}

}


void printSeasonGames(int numberOfRounds) {
```

```
            for (int i = 0; i < numberOfRounds - 6; i++)
                  for (int j = 0; j < this.GAMESPERROUND; j++)
                        System.out.println(this.Games[i][j].homeTeam +
" "
                                          + this.Games[i][j].awayTeam);

      }

      void recalculateStatistics(int round) {
         for (int i = 0; i < 24; i++) {
               LeaqueTable[i].computeStatisticsForHomeGames(round-1);

               LeaqueTable[i].computeStatisticsForAwayGames(round-1);


               LeaqueTable[i].computeStatisticsForHomeGamesLastSixGames(r
               ound-1);
               LeaqueTable[i].computeStatisticsForAwayGamesLastSixGames(r
               ound-1);
               LeaqueTable[i].computeTotalStatistics(round-1);
         }
      }
}
```

## Κλάση Team.java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.io.StreamTokenizer;

public class Team {

      String teamName;
      Match matchTable[];
      double statistics[];
      double epithimito[];
      public final int NUMBEROFMATCHES = 46;
      String seasonYear = "";

      Team(String teamName, String year) {
            matchTable = new Match[46];
            statistics = new double[23];
            this.teamName = teamName;
            epithimito = new double[46];
            this.seasonYear = year;
      }

      void createMatchTable(int numberOfRounds) throws IOException {
            String homeTeam;
            String awayTeam;
            double goals1;
            double goals2;
            String s3 = "";
            try {
                  s3 = new java.io.File(".").getCanonicalPath();
            } catch (IOException e) {
                  e.printStackTrace();
```

```
            }

            s3 = s3 + "\\data\\" + this.seasonYear + "\\" +
this.teamName + ".txt";

            Reader r = new BufferedReader(new FileReader(s3));
            StreamTokenizer stok = new StreamTokenizer(r);
            stok.parseNumbers();
            stok.nextToken();

            for (int i = 0; i < numberOfRounds; i++) {
                  homeTeam = stok.sval;
                  stok.nextToken();
                  awayTeam = stok.sval;
                  stok.nextToken();
                  goals1 = stok.nval;
                  stok.nextToken();
                  goals2 = stok.nval;
                  stok.nextToken();
                  this.matchTable[i] = new Match(homeTeam, awayTeam,
goals1, goals2);
                  epithimito[i] = goals1 + goals2;
            }

      }

      void printMatchTable(int numberOfRounds) {
            for (int i = 0; i < numberOfRounds; i++) {
                  System.out.print(this.matchTable[i].homeTeam);
                  System.out.print(" - ");
                  System.out.print(this.matchTable[i].awayTeam);
                  System.out.print("\t\t");
                  System.out.print((int)
this.matchTable[i].goalsHomeTeam);
                  System.out.print(" - ");
                  System.out.println((int)
this.matchTable[i].goalsAwayTeam);

            }
      }

      void computeStatisticsForHomeGames(int numberOfRounds) {
            double goalsFor = 0;
            double goalsAgainst = 0;
            double gamesScored = 0;
            double gamesOpponendScored = 0;
            double gamesGoneOver = 0;
            int numberOfHomeGames = 0;
            double temp = 0;

            for (int i = 0; i < numberOfRounds; i++)
                  if (matchTable[i].homeTeam.equals(this.teamName)) {
                        numberOfHomeGames++;
                        goalsFor = goalsFor +
matchTable[i].goalsHomeTeam;
                        goalsAgainst = goalsAgainst +
matchTable[i].goalsAwayTeam;
                        if (matchTable[i].goalsHomeTeam != 0)
                              gamesScored++;
                        if (matchTable[i].goalsAwayTeam != 0)
```

```
                              gamesOpponendScored++;
                        if (matchTable[i].goalsHomeTeam +
matchTable[i].goalsAwayTeam > 2.5)
                              gamesGoneOver++;

                  }
            temp = (goalsFor) / (numberOfHomeGames);
            this.statistics[0] = (temp - 0) / (4 - 0);
            // this.statistics[0]= (goalsFor) / (numberOfHomeGames);
            this.statistics[1] = gamesScored / numberOfHomeGames;

            temp = (goalsAgainst) / (numberOfHomeGames);
            this.statistics[2] = (temp - 0) / (3 - 0);
            // this.statistics[2]=(goalsAgainst) /
(numberOfHomeGames);

            this.statistics[3] = gamesOpponendScored /
numberOfHomeGames;
            this.statistics[8] = gamesGoneOver / numberOfHomeGames;

      }

      void computeStatisticsForAwayGames(int numberOfRounds) {
            double goalsFor = 0;
            double goalsAgainst = 0;
            double gamesScored = 0;
            double gamesOpponendScored = 0;
            double gamesGoneOver = 0;
            int numberOfAwayGames = 0;
            double temp;

            for (int i = 0; i < numberOfRounds; i++)
                  if (matchTable[i].awayTeam.equals(this.teamName)) {
                        numberOfAwayGames++;
                        goalsFor = goalsFor +
matchTable[i].goalsAwayTeam;
                        goalsAgainst = goalsAgainst +
matchTable[i].goalsHomeTeam;
                        if (matchTable[i].goalsAwayTeam != 0)
                              gamesScored++;
                        if (matchTable[i].goalsHomeTeam != 0)
                              gamesOpponendScored++;
                        if (matchTable[i].goalsHomeTeam +
matchTable[i].goalsAwayTeam > 2.5)
                              gamesGoneOver++;

                  }
            // System.out.println(sum);

            temp = (goalsFor) / (numberOfAwayGames);
            this.statistics[4] = (temp - 0) / (3 - 0);
            // this.statistics[4]= (goalsFor) / (numberOfAwayGames);

            this.statistics[5] = gamesScored / numberOfAwayGames;

            temp = (goalsAgainst) / (numberOfAwayGames);
            this.statistics[6] = (temp - 0) / (4 - 0);
            // this.statistics[6] =(goalsAgainst) /
(numberOfAwayGames);
```

```
            this.statistics[7] = gamesOpponendScored /
numberOfAwayGames;
            this.statistics[9] = gamesGoneOver / numberOfAwayGames;

      }

      void computeStatisticsForHomeGamesLastSixGames(int round) {
            double goalsFor = 0;
            double goalsAgainst = 0;
            double gamesScored = 0;
            double gamesOpponendScored = 0;
            double gamesGoneOver = 0;
            int numberOfHomeGames = 0;
            double temp;
            // round = round - 1;
            for (int i = round - 6; i < round; i++)
                  if (matchTable[i].homeTeam.equals(this.teamName)) {
                        numberOfHomeGames++;
                        goalsFor = goalsFor +
matchTable[i].goalsHomeTeam;
                        goalsAgainst = goalsAgainst +
matchTable[i].goalsAwayTeam;
                        if (matchTable[i].goalsHomeTeam != 0)
                              gamesScored++;
                        if (matchTable[i].goalsAwayTeam != 0)
                              gamesOpponendScored++;
                        if (matchTable[i].goalsHomeTeam +
matchTable[i].goalsAwayTeam > 2.5)
                              gamesGoneOver++;

                  }
            if (numberOfHomeGames != 0) {

                  temp = (goalsFor) / (numberOfHomeGames);
                  this.statistics[10] = (temp - 0) / (5 - 0);
                  // this.statistics[10] = (goalsFor) /
(numberOfHomeGames);
                  this.statistics[11] = gamesScored /
numberOfHomeGames;

                  temp = (goalsAgainst) / (numberOfHomeGames);
                  this.statistics[12] = (temp - 0) / (4 - 0);
                  // this.statistics[12] =(goalsAgainst) /
(numberOfHomeGames);

                  this.statistics[13] = gamesOpponendScored /
numberOfHomeGames;
                  this.statistics[18] = gamesGoneOver /
numberOfHomeGames;
            } else {
                  this.statistics[10] = 0;
                  this.statistics[11] = 0;
                  this.statistics[12] = 0;
                  this.statistics[13] = 0;
                  this.statistics[18] = 0;
            }

      }

      void computeStatisticsForAwayGamesLastSixGames(int round) {
```

```
            double goalsFor = 0;
            double goalsAgainst = 0;
            double gamesScored = 0;
            double gamesOpponendScored = 0;
            double gamesGoneOver = 0;
            int numberOfAwayGames = 0;
            double temp;
            // round = round - 1;
            for (int i = round - 6; i < round; i++)
                  if (matchTable[i].awayTeam.equals(this.teamName)) {
                        numberOfAwayGames++;
                        goalsFor = goalsFor +
matchTable[i].goalsAwayTeam;
                        goalsAgainst = goalsAgainst +
matchTable[i].goalsHomeTeam;
                        if (matchTable[i].goalsAwayTeam != 0)
                              gamesScored++;
                        if (matchTable[i].goalsHomeTeam != 0)
                              gamesOpponendScored++;
                        if (matchTable[i].goalsHomeTeam +
matchTable[i].goalsAwayTeam > 2.5)
                              gamesGoneOver++;
                  }
            // System.out.println(sum);
            if (numberOfAwayGames != 0) {

                  temp = (goalsFor) / (numberOfAwayGames);
                  this.statistics[14] = (temp - 0) / (4 - 0);
                  // this.statistics[14] =(goalsFor) /
(numberOfAwayGames);

                  this.statistics[15] = gamesScored /
numberOfAwayGames;

                  temp = (goalsAgainst) / (numberOfAwayGames);
                  this.statistics[16] = (temp - 0) / (4 - 0);
                  // this.statistics[16] =(goalsAgainst) /
(numberOfAwayGames);

                  this.statistics[17] = gamesOpponendScored /
numberOfAwayGames;
                  this.statistics[19] = gamesGoneOver /
numberOfAwayGames;
            } else {
                  this.statistics[14] = 0;
                  this.statistics[15] = 0;
                  this.statistics[16] = 0;
                  this.statistics[17] = 0;
                  this.statistics[19] = 0;
            }

      }

      void computeTotalStatistics(int round) {

            double gamesGoneOver = 0;
            double goalsTotal = 0;
            double gamesGoneOverLastSixGames = 0;
            double temp;
```

```
            for (int i = 0; i < round; i++) {
                  if (matchTable[i].goalsHomeTeam +
matchTable[i].goalsAwayTeam > 2.5)
                        gamesGoneOver++;
                  goalsTotal = goalsTotal +
matchTable[i].goalsHomeTeam
                              + matchTable[i].goalsAwayTeam;
            }

            // round = round - 1;
            for (int i = round - 6; i < round; i++)
                  if (matchTable[i].goalsHomeTeam +
matchTable[i].goalsAwayTeam > 2.5)
                        gamesGoneOverLastSixGames++;

            round++;

            temp = (goalsTotal) / (round);
            this.statistics[20] = (temp - 1) / (4 - 1);
            // this.statistics[20] =(goalsTotal) / (round);

            this.statistics[21] = gamesGoneOver / (round);
            this.statistics[22] = gamesGoneOverLastSixGames / 6;

      }

      void printStatisticsTable() {
            for (int i = 0; i < 23; i++)
                  System.out.println(this.statistics[i]);

      }

}
```

## Κλάση TrainingProcedure.java

```
import java.io.*;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.io.StreamTokenizer;
import java.util.Random;

public class TrainingProcedure {

	public static final int NUMBER_OF_HIDDEN_LAYER_NODES = 45;
	int NUMBEROFTEAMS = 24;
	double dataTable[][];
	double epi8imito[];
	int correctDecision;
	int wrongDecision;
	int correctMORE;
	int correctLESS;
	int wrongMORE;
	int wrongLESS;
	Season s1;

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

		String team1;
		String team2;
		int t1;
		int t2;
		double N = 0.002;
		TrainingProcedure mainClass = new TrainingProcedure();
		String seasonStr = "";
		double iteratorError = 0;

		// Create file
		FileWriter fstream = new FileWriter("out.txt");
		BufferedWriter out = new BufferedWriter(fstream);
		FileWriter fstream2;
		BufferedWriter out2;
		FileWriter fstream3 = new FileWriter("trainingerror.txt");
		BufferedWriter out3 = new BufferedWriter(fstream3);
		FileWriter fstreamtemp = new
FileWriter("SVMtraining.txt");
		BufferedWriter outtemp = new BufferedWriter(fstreamtemp);

		HiddenLayerNode[] HiddenLayerNodes = new
HiddenLayerNode[NUMBER_OF_HIDDEN_LAYER_NODES];


		HiddenLayerNodes[0] = new HiddenLayerNode(2.51, 5, N);
		HiddenLayerNodes[1] = new HiddenLayerNode(1.59, 6, N);
		HiddenLayerNodes[2] = new HiddenLayerNode(1.22, 7, N);
		HiddenLayerNodes[3] = new HiddenLayerNode(1.77, 3, N);
		HiddenLayerNodes[4] = new HiddenLayerNode(2.65, 6, N);
		HiddenLayerNodes[5] = new HiddenLayerNode(2.25, 6, N);
		HiddenLayerNodes[6] = new HiddenLayerNode(3.2, 4, N);
		HiddenLayerNodes[7] = new HiddenLayerNode(0.85, 6, N);
```

```
HiddenLayerNodes[8] = new HiddenLayerNode(1.45, 7, N);
HiddenLayerNodes[9] = new HiddenLayerNode(4.51, 5, N);
HiddenLayerNodes[10] = new HiddenLayerNode(1.4, 5, N);
HiddenLayerNodes[11] = new HiddenLayerNode(1.32, 3, N);
HiddenLayerNodes[12] = new HiddenLayerNode(3.4, 5, N);
HiddenLayerNodes[13] = new HiddenLayerNode(2.35, 4, N);
HiddenLayerNodes[14] = new HiddenLayerNode(1.61, 4, N);
HiddenLayerNodes[15] = new HiddenLayerNode(3.25, 5, N);
HiddenLayerNodes[16] = new HiddenLayerNode(2.23, 6, N);
HiddenLayerNodes[17] = new HiddenLayerNode(2.13, 7, N);
HiddenLayerNodes[18] = new HiddenLayerNode(2.45, 7, N);
HiddenLayerNodes[19] = new HiddenLayerNode(1.33, 5, N);
HiddenLayerNodes[20] = new HiddenLayerNode(2.11, 4, N);
HiddenLayerNodes[21] = new HiddenLayerNode(1.22, 5, N);
HiddenLayerNodes[22] = new HiddenLayerNode(2.4, 4, N);
HiddenLayerNodes[23] = new HiddenLayerNode(2.35, 3, N);
HiddenLayerNodes[24] = new HiddenLayerNode(1.61, 2, N);
HiddenLayerNodes[25] = new HiddenLayerNode(2.25, 5, N);
HiddenLayerNodes[26] = new HiddenLayerNode(3.23, 7, N);
HiddenLayerNodes[27] = new HiddenLayerNode(2.13, 5, N);
HiddenLayerNodes[28] = new HiddenLayerNode(2.45, 7, N);
HiddenLayerNodes[29] = new HiddenLayerNode(1.81, 5, N);
HiddenLayerNodes[30] = new HiddenLayerNode(3.25, 5, N);
HiddenLayerNodes[31] = new HiddenLayerNode(2.23, 6, N);
HiddenLayerNodes[32] = new HiddenLayerNode(2.13, 7, N);
HiddenLayerNodes[33] = new HiddenLayerNode(2.45, 7, N);
HiddenLayerNodes[34] = new HiddenLayerNode(1.33, 5, N);
HiddenLayerNodes[35] = new HiddenLayerNode(2.11, 4, N);
HiddenLayerNodes[36] = new HiddenLayerNode(1.22, 5, N);
HiddenLayerNodes[37] = new HiddenLayerNode(2.4, 4, N);
HiddenLayerNodes[38] = new HiddenLayerNode(2.35, 3, N);
HiddenLayerNodes[39] = new HiddenLayerNode(1.61, 2, N);
HiddenLayerNodes[40] = new HiddenLayerNode(2.25, 5, N);
HiddenLayerNodes[41] = new HiddenLayerNode(3.23, 7, N);
HiddenLayerNodes[42] = new HiddenLayerNode(2.13, 5, N);
HiddenLayerNodes[43] = new HiddenLayerNode(2.45, 7, N);
HiddenLayerNodes[44] = new HiddenLayerNode(1.81, 5, N);

OutputNode outputLayer = new OutputNode();

double sum = 0;
int itterator = 0;
double epithimito;

mainClass.initializeCentreVectors(HiddenLayerNodes);

try {
      for (int season = 2001; season <= 2010; season++) {

            if (season == 2001)
                  seasonStr = "2001";
            else if (season == 2002)
                  seasonStr = "2002";
            else if (season == 2003)
                  seasonStr = "2003";
            else if (season == 2004)
                  seasonStr = "2004";
            if (season == 2005)
                  seasonStr = "2005";
            else if (season == 2006)
```

```
                        seasonStr = "2006";
                  else if (season == 2007)
                        seasonStr = "2007";
                  else if (season == 2008)
                        seasonStr = "2008";
                  else if (season == 2009)
                        seasonStr = "2009";
                  else if (season == 2010)
                        seasonStr = "2010";

                  mainClass.s1 = new Season(24, seasonStr);
                  mainClass.s1.initializeTable(24, 46);
                  mainClass.s1.initializeGames(46);
                  mainClass.UpdateGames();

                  do {
                        iteratorError = 0;
                        if ((itterator == 0) || (itterator ==
100)
                                    || (itterator == 199) ||
(itterator == 299)
                                    || (itterator == 499)) {
                              out.write("\nItteration " +
itterator);
                              out.write("\n==============\n");
                        }

                        for (int p = 0; p < 46 - 6; p++) {

      mainClass.s1.recalculateStatistics(p + 7);

                              for (int q = 0; q < 12; q++) {
                                    team1 =
mainClass.s1.Games[p][q].homeTeam;
                                    team2 =
mainClass.s1.Games[p][q].awayTeam;

                                    t1 =
mainClass.findTeamNumber(team1);
                                    t2 =
mainClass.findTeamNumber(team2);

                                    for (int i = 0; i <
NUMBER_OF_HIDDEN_LAYER_NODES; i++) {
                                          for (int j = 0; j <
23; j++) {

      HiddenLayerNodes[i].InputVector[j] =
mainClass.s1.LeaqueTable[t1].statistics[j];
                                          }
                                          for (int j = 23; j <
46; j++) {

      HiddenLayerNodes[i].InputVector[j] =
mainClass.s1.LeaqueTable[t2].statistics[j - 23];
                                          }
                                    }
                                    epithimito =
mainClass.s1.Games[p][q].epithimito;
```

```
                                        // if (season == 2010) {

                                        if (epithimito < 2.5)
                                              outtemp.write("-1\t");
                                        else
                                              outtemp.write("1\t");
                                        for (int j = 0; j < 23; j++)
{
                                              outtemp
                                                          .write(j

      + 1

      + ":"

      + mainClass.s1.LeaqueTable[t1].statistics[j]

      + "\t");
                                        }
                                        for (int j = 23; j < 46;
j++) {
                                              outtemp
                                                          .write(j

      + 1

      + ":"

      + mainClass.s1.LeaqueTable[t2].statistics[j - 23]

      + "\t");
                                        }
                                        outtemp.write("\n");
                                        // }*/

                                        for (int u = 0; u <
NUMBER_OF_HIDDEN_LAYER_NODES; u++) {
      HiddenLayerNodes[u].computeEucleidianDistance();

      HiddenLayerNodes[u].GaussianFunction();

      HiddenLayerNodes[u].computeOutput();
                                              outputLayer.input[u] =
HiddenLayerNodes[u].output;
                                        }

                                        outputLayer.computeSum();

      outputLayer.computeError(epithimito,

      outputLayer.sum);
                                        if ((itterator == 0) ||
(itterator == 100)
                                                    || (itterator ==
199) || (itterator == 299)
                                                    || (itterator ==
499)) {
      out.write(mainClass.s1.Games[p][q].homeTeam
```

```
                                                                + " - ");
      out.write(mainClass.s1.Games[p][q].awayTeam
                                                                + "\n");
                                                        out.write("Sum = " +
outputLayer.sum);
                                                        out.write("\nEpi8imito
= " + epithimito);
                                                        out.write("\nError " +
outputLayer.error);
                                                        out.write("\n");
                                                        out.write("\n");

                                                    }
                                                    iteratorError =
iteratorError
                                                                +
Math.abs(outputLayer.error);
                                                    sum = 0;
                                                    for (int i = 0; i <
NUMBER_OF_HIDDEN_LAYER_NODES; i++) {
                                                        sum = sum +
HiddenLayerNodes[i].output;

                                                    }
                                                    // sum = sum /
NUMBER_OF_HIDDEN_LAYER_NODES;

                                                    for (int i = 0; i <
NUMBER_OF_HIDDEN_LAYER_NODES; i++) {
                                                        // out.write("\nNeuron
" + i);
      HiddenLayerNodes[i].changeC(outputLayer.error,
                                                                sum);
                                                        HiddenLayerNodes[i]

      .changeCentre(outputLayer.error);

      HiddenLayerNodes[i].changeS(outputLayer.error);

                                                        if (((itterator == 0)
|| (itterator == 100)
                                                                ||
(itterator == 199)
                                                                ||
(itterator == 299) || (itterator == 499))
                                                                && (q ==
0)) {
                                                            out.write("\nC =
" + HiddenLayerNodes[i].C);
      HiddenLayerNodes[i].printCentre(out);
                                                        out.write("\nS =
" + HiddenLayerNodes[i].S);
                                                        //
out.write("\nerror = " +
                                                        //
HiddenLayerNodes[i].error);
```

```
                                                      out.write("\n");
                                                }
                                          }

                                          }

                                    }
                                    out3.write("" + iteratorError / 480 +
"\n");
                                    itterator++;
                              } while (itterator < 1);

                              out.write("\t\t\t\tPERIODOS " + seasonStr +
"\n\n");
                              itterator = 0;
                              // for (int p = 0; p < 24; p++) {
                              //
System.out.println(mainClass.s1.LeaqueTable[p].teamName);
                              //
mainClass.s1.LeaqueTable[p].printStatisticsTable();
                              // }
                        }
                        // Close the output stream
                        out.close();
                        out3.close();
                  } catch (Exception e) {// Catch exception if any
                        System.err.println("Error2: " + e.getMessage());
                  }
                  mainClass.saveCentreVectors(HiddenLayerNodes);

                  try {
                        //
==========================================================================
                        mainClass.s1 = new Season(24, "2010");

                        mainClass.s1.initializeTable(24, 46);
                        mainClass.s1.initializeGames(46);
                        mainClass.UpdateGames();

                        // Create file
                        fstream2 = new FileWriter("out2009.txt");
                        out2 = new BufferedWriter(fstream2);

                        for (int p = 0; p < 46 - 6; p++) {
                              mainClass.s1.recalculateStatistics(p + 7);
                              for (int q = 0; q < 12; q++) {
                                    team1 =
mainClass.s1.Games[p][q].homeTeam;
                                    team2 =
mainClass.s1.Games[p][q].awayTeam;

                                    t1 = mainClass.findTeamNumber(team1);
                                    t2 = mainClass.findTeamNumber(team2);

                                    for (int i = 0; i <
NUMBER_OF_HIDDEN_LAYER_NODES; i++) {
                                          for (int j = 0; j < 23; j++) {

      HiddenLayerNodes[i].InputVector[j] =
mainClass.s1.LeaqueTable[t1].statistics[j];
```

```
                                    }
                                    for (int j = 23; j < 46; j++) {

      HiddenLayerNodes[i].InputVector[j] =
mainClass.s1.LeaqueTable[t2].statistics[j - 23];
                                    }
                              }
                              // epithimito =
                              //
mainClass.s1.LeaqueTable[t1].epithimito[p+6];
                              epithimito =
mainClass.s1.Games[p][q].epithimito;

                              for (int u = 0; u <
NUMBER_OF_HIDDEN_LAYER_NODES; u++) {

      HiddenLayerNodes[u].computeEucleidianDistance();

      HiddenLayerNodes[u].GaussianFunction();

      HiddenLayerNodes[u].computeOutput();
                              }
                              for (int u = 0; u <
NUMBER_OF_HIDDEN_LAYER_NODES; u++) {
                                    outputLayer.input[u] =
HiddenLayerNodes[u].output;
                              }
                              outputLayer.computeSum();
                              outputLayer.computeError(epithimito,
outputLayer.sum);
                              out2.write(team1 + " - " + team2 +
"\n");
                              out2.write("Sum = " + outputLayer.sum);
                              out2.write("\nEpi8imito = " +
epithimito);
                              out2.write("\nError " +
outputLayer.error);
                              out2.write("\n");
                              out2.write("\n");

      mainClass.computeStatistics(outputLayer.sum, epithimito);

                        }
                  }
                  out2.write("\n\n\n");
                  out2.write("correct decisions :" +
mainClass.correctDecision);
                  out2.write("\nwrong decisions :" +
mainClass.wrongDecision);
                  out2.write("\ncorrect MORE :" +
mainClass.correctMORE);
                  out2.write("\ncorrect LESS :" +
mainClass.correctLESS);
                  out2.write("\nwrong MORE :" + mainClass.wrongMORE);
                  out2.write("\nwrong LESS :" + mainClass.wrongLESS);
                  // Close the output stream
                  out2.close();
                  for (int g = 0; g < 46; g++) {
                        System.out
```

```
      .print(mainClass.s1.LeaqueTable[0].matchTable[g].homeTeam);
                        System.out.print(" - ");
                        System.out

      .print(mainClass.s1.LeaqueTable[0].matchTable[g].awayTeam
                                                + " ");
                        System.out

      .print(mainClass.s1.LeaqueTable[0].matchTable[g].goalsHomeTeam
                                                + " - ");
                        System.out

      .print(mainClass.s1.LeaqueTable[0].matchTable[g].goalsAwayTeam
                                                + "  ");
                        System.out

      .println(mainClass.s1.LeaqueTable[0].matchTable[g].epithimito);
                  }
            } catch (Exception e) {// Catch exception if any
                  System.err.println("Error3: " + e.getMessage());
            }
            System.out.println(mainClass.correctDecision + "\t"
                        + mainClass.wrongDecision);

            /*
             *
==========================================================================
             * ==
             *
==========================================================================
             */

      }

      void saveCentreVectors(HiddenLayerNode[] HiddenLayerNodes) {
            try {
                  FileWriter fstream2 = new
FileWriter("NetworkStructure.txt");
                  BufferedWriter out2 = new BufferedWriter(fstream2);

                  for (int i = 0; i < NUMBER_OF_HIDDEN_LAYER_NODES;
i++) {
                        out2.write("\n\nHidden Layer " + i + "\n");
                        out2.write("===============" + "\n\n");
                        out2.write("Sigma\t" + HiddenLayerNodes[i].S +
"\n");
                        out2.write("Coefficient\t" +
HiddenLayerNodes[i].C + "\n\n");
                        out2.write("Centres\t");
                        for (int j = 0; j < 46; j++) {
                              if ((HiddenLayerNodes[i].CentreVector[j]
< 0.001)
                                                &&
(HiddenLayerNodes[i].CentreVector[j] > -0.001)) {
                                    out2.write("0   ");
                              } else {

      out2.write(HiddenLayerNodes[i].CentreVector[j] + "   ");
                              }
```

```
                        }
                  }

                  out2.close();

            } catch (Exception e) {// Catch exception if any
                  System.err.println("Error2: " + e.getMessage());
            }

      }

      void read(String fileName, String actualOutput) throws
IOException {
            int tableRows = 4;
            int numberOfData = 2;
            this.dataTable = new double[tableRows][numberOfData];
            this.epi8imito = new double[(this.NUMBEROFTEAMS - 1) * 2];

            Reader r = new BufferedReader(new FileReader(fileName));
            StreamTokenizer stok = new StreamTokenizer(r);

            stok.parseNumbers();
            stok.nextToken();
            for (int i = 0; i < 4; i++) {
                  stok.nextToken();
                  for (int j = 0; j < 2; j++) {
                        if (stok.ttype == StreamTokenizer.TT_NUMBER)
                              this.dataTable[i][j] = (int) stok.nval;
                        stok.nextToken();
                  }
            }

            Reader r2 = new BufferedReader(new
FileReader(actualOutput));
            StreamTokenizer stok2 = new StreamTokenizer(r2);

            stok.parseNumbers();
            stok.nextToken();
            int z = 0;
            for (int i = 0; i < 4; i++) {
                  for (int j = i; j < 4; j++) {
                        if (stok2.ttype == StreamTokenizer.TT_NUMBER)
{
                              this.epi8imito[z] = (int) stok2.nval;
                              z++;
                        }
                        stok2.nextToken();
                  }
            }

      }

      public void initializeCentreVectors(HiddenLayerNode[]
HiddenLayerNodes) {
            Random randomGenerator = new Random();
            for (int i = 0; i < NUMBER_OF_HIDDEN_LAYER_NODES; i++)
                  for (int j = 0; j < 46; j++)
                        HiddenLayerNodes[i].CentreVector[j] =
randomGenerator
                                    .nextDouble();
```

```
	}

	int findTeamNumber(String teamName) {
		for (int i = 0; i < NUMBEROFTEAMS; i++) {
			if
(teamName.equals(this.s1.LeaqueTable[i].teamName))
				return i;

		}
		return 0;
	}

	void computeStatistics(double sum, double epi8ymito) {

		if ((sum > 2.5) && (epi8ymito > 2.5)) {
			this.correctDecision++;
			this.correctMORE++;
		} else if ((sum < 2.5) && (epi8ymito < 2.5)) {
			this.correctDecision++;
			this.correctLESS++;
		} else if ((sum > 2.5) && (epi8ymito < 2.5)) {
			this.wrongDecision++;
			this.wrongMORE++;
		} else if ((sum < 2.5) && (epi8ymito > 2.5)) {
			this.wrongDecision++;
			this.wrongLESS++;
		}

	}

	void UpdateGames() {
		int t1;
		String team1;
		String team2;
		double epithimito;
		for (int i = 0; i < 46 - 6; i++) {
			for (int j = 0; j < 12; j++) {
				team1 = this.s1.Games[i][j].homeTeam;
				team2 = this.s1.Games[i][j].awayTeam;
				t1 = findTeamNumber(team1);
				for (int b = 0; b < 46; b++) {
					if
(this.s1.LeaqueTable[t1].matchTable[b].awayTeam
							.equals(team2)) {
						epithimito =
this.s1.LeaqueTable[t1].epithimito[b];
						this.s1.Games[i][j].epithimito =
epithimito;
						break;
					}
				}
			}
		}

	}

}
```

# Κλάση QuickLauncher.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;
import java.io.StreamTokenizer;

public class QuickLauncher {

	public static final int NUMBER_OF_HIDDEN_LAYER_NODES = 30;
	int NUMBEROFTEAMS = 24;
	Season s1;

	int round;

	QuickLauncher(int currentround) {
		this.round=currentround;
	}

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

		HiddenLayerNode[] HiddenLayerNodes = new
HiddenLayerNode[30];
		OutputNode outputLayer = new OutputNode();
		double N = 0.2;
		//QuickLauncher Q1 = new
QuickLauncher(numberofrounds,currentround-1);
		QuickLauncher Q1 = new QuickLauncher(46-1);
		FileWriter fstream2;
		BufferedWriter out2;
		String team1;
		String team2;
		int t1;
		int t2;

		HiddenLayerNodes[0] = new HiddenLayerNode(4.5, 0.10, N);
		HiddenLayerNodes[1] = new HiddenLayerNode(5.5, 0.15, N);
		HiddenLayerNodes[2] = new HiddenLayerNode(4, 0.18, N);
		HiddenLayerNodes[3] = new HiddenLayerNode(5, 0.20, N);
		HiddenLayerNodes[4] = new HiddenLayerNode(6.5, 0.11, N);
		HiddenLayerNodes[5] = new HiddenLayerNode(2.5, 0.45, N);
		HiddenLayerNodes[6] = new HiddenLayerNode(2, 0.48, N);
		HiddenLayerNodes[7] = new HiddenLayerNode(5, 0.22, N);
		HiddenLayerNodes[8] = new HiddenLayerNode(4.5, 0.12, N);
		HiddenLayerNodes[9] = new HiddenLayerNode(5.1, 0.14, N);
		HiddenLayerNodes[10] = new HiddenLayerNode(4, 0.18, N);
		HiddenLayerNodes[11] = new HiddenLayerNode(3.2, 0.20, N);
		HiddenLayerNodes[12] = new HiddenLayerNode(4, 0.18, N);
		HiddenLayerNodes[13] = new HiddenLayerNode(3.5, 0.20, N);
		HiddenLayerNodes[14] = new HiddenLayerNode(6.1, 0.10, N);
		HiddenLayerNodes[15] = new HiddenLayerNode(2.5, 0.45, N);
		HiddenLayerNodes[16] = new HiddenLayerNode(2.3, 0.48, N);
		HiddenLayerNodes[17] = new HiddenLayerNode(1.3, 0.20, N);
		HiddenLayerNodes[18] = new HiddenLayerNode(4.5, 0.12, N);
		HiddenLayerNodes[19] = new HiddenLayerNode(8.1, 0.14, N);
```

```
            HiddenLayerNodes[20] = new HiddenLayerNode(1.1, 0.18, N);
            HiddenLayerNodes[21] = new HiddenLayerNode(2.2, 0.20, N);
            HiddenLayerNodes[22] = new HiddenLayerNode(4, 0.18, N);
            HiddenLayerNodes[23] = new HiddenLayerNode(3.5, 0.20, N);
            HiddenLayerNodes[24] = new HiddenLayerNode(6.1, 0.10, N);
            HiddenLayerNodes[25] = new HiddenLayerNode(2.5, 0.45, N);
            HiddenLayerNodes[26] = new HiddenLayerNode(2.3, 0.48, N);
            HiddenLayerNodes[27] = new HiddenLayerNode(1.3, 0.20, N);
            HiddenLayerNodes[28] = new HiddenLayerNode(4.5, 0.12, N);
            HiddenLayerNodes[29] = new HiddenLayerNode(8.1, 0.14, N);

            Q1.LoadCentreVectors(HiddenLayerNodes);

            try {
                  Q1.s1 = new Season(24, "2010");
                  Q1.s1.initializeTable(24,Q1.round);
                  Q1.s1.initializeGames(Q1.round);
                  //Q1.UpdateGames(Q1.round);

                  // Create file
                  fstream2 = new FileWriter("predictions.txt");
                  out2 = new BufferedWriter(fstream2);

                        Q1.s1.recalculateStatistics(Q1.round);

                        Q1.s1.LeaqueTable[0].printStatisticsTable();

                        for (int q = 0; q < 12; q++) {
                              team1 = Q1.s1.Games[Q1.round-
7][q].homeTeam;
                              team2 = Q1.s1.Games[Q1.round-
7][q].awayTeam;

                              t1 = Q1.findTeamNumber(team1);
                              t2 = Q1.findTeamNumber(team2);

                              for (int i = 0; i <
NUMBER_OF_HIDDEN_LAYER_NODES; i++) {
                                    for (int j = 0; j < 23; j++) {

      HiddenLayerNodes[i].InputVector[j] =
Q1.s1.LeaqueTable[t1].statistics[j];
                                    }
                                    for (int j = 23; j < 46; j++) {

      HiddenLayerNodes[i].InputVector[j] =
Q1.s1.LeaqueTable[t2].statistics[j - 23];
                                    }
                              }


                              for (int u = 0; u <
NUMBER_OF_HIDDEN_LAYER_NODES; u++) {

      HiddenLayerNodes[u].computeEucleidianDistance();

      HiddenLayerNodes[u].GaussianFunction();

      HiddenLayerNodes[u].computeOutput();
                              }
```

```
                              for (int u = 0; u <
NUMBER_OF_HIDDEN_LAYER_NODES; u++) {
                                    outputLayer.input[u] =
HiddenLayerNodes[u].output;
                              }
                              outputLayer.computeSum();
                              // outputLayer.computeError(epithimito,
outputLayer.sum);
                              out2.write(team1 + " - " + team2 +
"\n");
                              out2.write("Sum = " + outputLayer.sum);
                              // out2.write("\nEpi8imito = " +
epithimito);
                              // out2.write("\nError " +
outputLayer.error);
                              out2.write("\n");
                              out2.write("\n");
                              // Q1.computeStatistics(outputLayer.sum,
epithimito);

                        }

                  out2.close();
            } catch (Exception e) {// Catch exception if any
                  System.err.println("Error3: " + e.getMessage());
            }

      }

      public void LoadCentreVectors(HiddenLayerNode[]
HiddenLayerNodes)
                  throws IOException {

            String s3 = "";
            try {
                  s3 = new java.io.File(".").getCanonicalPath();
            } catch (IOException e) {
                  e.printStackTrace();
            }
            s3 = s3 +
"\\data\\NetworkStructure\\NetworkStructure.txt";

            Reader r = new BufferedReader(new FileReader(s3));
            StreamTokenizer stok = new StreamTokenizer(r);



            for (int i = 0; i < NUMBER_OF_HIDDEN_LAYER_NODES; i++) {
                  for (int t = 0; t < 20; t++)
                        stok.nextToken();

                  //System.out.println("sigma= " + stok.nval);
                  HiddenLayerNodes[i].S=stok.nval;
                  //System.out.println("S = "+ HiddenLayerNodes[i].S);
                  stok.nextToken();
                  stok.nextToken();
                  HiddenLayerNodes[i].C=stok.nval;
                  //System.out.println("C = "+ HiddenLayerNodes[i].C);
                  stok.nextToken();
```

```
                  for (int j = 0; j < 46; j++) {
                        stok.nextToken();
                        HiddenLayerNodes[i].CentreVector[j]=stok.nval;


      //System.out.println(HiddenLayerNodes[i].CentreVector[j]);
                        //System.out.println("centre " + j + "= " +
stok.nval);

                  }
            }

      }

      void UpdateGames(int numberOfRounds) {
            int t1;
            String team1;
            String team2;
            double epithimito;
            for (int i = 0; i < numberOfRounds - 6; i++) {
                  for (int j = 0; j < 12; j++) {
                        team1 = this.s1.Games[i][j].homeTeam;
                        team2 = this.s1.Games[i][j].awayTeam;
                        t1 = findTeamNumber(team1);
                        for (int b = 0; b < 46; b++) {
                              if
(this.s1.LeaqueTable[t1].matchTable[b].awayTeam
                                          .equals(team2)) {
                                    epithimito =
this.s1.LeaqueTable[t1].epithimito[b];
                                    this.s1.Games[i][j].epithimito =
epithimito;
                                    break;
                              }
                        }
                  }
            }

      }

      int findTeamNumber(String teamName) {
            for (int i = 0; i < NUMBEROFTEAMS; i++) {
                  if
(teamName.equals(this.s1.LeaqueTable[i].teamName))
                        return i;

            }
            return 0;
      }

}
```

# Παράρτημα Β - Παράδειγμα διαδικασίας εκπαίδευσης

**BIRMINGHAM - BLACKPOOL**
**Sum = 1.7590684390090623**
**Epi8imito = 1.0**
**Error -0.7590684390090623**

Hidden Layer 0
==========

Sigma 2.4737564791221462
Coefficient 25.353560310259

Centres 2.384652218687835 -0.08285148196247057
0.6097060272924336 2.0193128620807705 -0.2927603444676971
-0.9873644179794169 1.6562625677526288 0.5982827018174
0.3535727779873702 2.2900665066205828 2.7343172498987673
2.30524836916381 2.4619040922140356 0.4308663078745849 -
0.45463570119714763 -1.027025770017995 1.9767962799161283
1.3788040029664947 0.9728968549651632 0.9477120043118161
0.6719250197212621 -0.1406267488083821 -0.15592426511197002
0.4295921559182886 -0.180015763178168 -0.2557011807530624
0.40526709312303427 3.463837860272738 0.294627150393097
1.5849551870624043 -0.9445988042173599 0.4243200790815518
0.47875761846614034 0.17623042589443394 -0.6683933411114361
1.4127542566075268 -0.882881942385184 1.5935367682753587 -
0.2579023108961863 0.13929141381085966 0.7413517505924628
0.9090157331981362 -0.6260668691952894 3.250661796294799
1.4421076052012216 1.305124918266113

Hidden Layer 1
==========

Sigma 4.412371185172986
Coefficient 25.40356031025911

Centres 3.5912179664249266 2.042265655719807
1.8967941751953337 1.0527686575200883 -0.41805208834382823
0.940177518637327 -1.1826362410626934 0.6765448506561665
3.418457867045231 -0.10507820428217879 -1.1172014651113784
-0.7540746325041107 -0.4400317365961809 -1.100964675325662
1.689500756989701 2.224475449234374 -0.8926524677651102
0.4783763576071249 -0.049263038793369275 0.02860550764939165
3.835500495562967 1.7327620684654554 0.2841088420353823 -
0.8951904034674446 0.3403010382512476 -0.3989850416064113
1.511837849185764 1.728754233800554 1.8679113971947603
4.1765031266304184 2.6918939244064353 0.04201790272464731
2.959931800738533 1.0308736317884708 0.5053033803264825
1.0106979410891033 0.6266794734216171 -0.23550919689624997
1.5006526881558657 -0.818264455520722 -0.4159165322365632
0.8304745124646803 0.03628841312240291 4.340170816987147
1.5128093434128824 0.3120369907521272

Hidden Layer 2
==========

Sigma 4.301894817861381
Coefficient 25.433560310259118

Centres 3.3280669209164917 1.9579229408476473
1.8215693091532283 1.4132873154640344 -1.3456915943583754
0.6755289470142098 -1.5085735655714065 0.8067258963634216
3.449398357031792 -0.3671839671886388 -1.1220420196338743
-0.7834398210700748 -0.55844524685087 -1.0556113844298765
1.5253140500114382 1.9052547064371108 -0.5914525848729819
0.778366509977519 -0.30871074740888393 -0.038004362276525164
3.8359786131519065 1.6116308468643583 0.11050026081213647
-0.9061604330189328 0.5302883723055496 -1.3644814294962524
1.352805021272844 1.4338472159621867 1.8462519686437882
3.914263796931134 2.569129256060563 -0.21480706672580754
2.913663802632889 0.9810729609262302 0.656343825941518
1.1388511697385122 0.9456784174924447 -0.4563016649358785
1.1282102604858804 -0.7656595161131582 -0.5025778339378643
0.6554578096594769 -0.36593081909926933 4.360691792282341
1.3589090271243507 -0.007031068086465091

Hidden Layer 3
==========

Sigma 4.578283242911713
Coefficient 25.45356031025912

Centres 3.531891238275793 1.983938659595617
2.036018838771631 0.8191722387299769 0.18195869373514942
1.2770094192200752 -0.5425063902062843 0.6379625954692772
3.3996952610784357 0.5810977516612897 -0.9643640301617211
-0.6390132119841658 -0.06454591285316365 -
0.11005464115379018 1.422996538056972 2.2816503201841702 -
1.3425572396979084 -0.10995453083152555 0.4154126560784002
-0.04881116566342269 4.059010758969382 2.0499404694403567
0.4275986768791301 -1.104247110045677 0.2336887710959846
0.4696294766716212 1.6323290634255083 2.0490831442001154
1.945743119866583 4.154406692535701 3.1293543832493307
0.6926315052600795 3.198574021312649 1.495130748744967
0.8347611944865092 0.2895640258069434 -0.14510153053204453
0.13810277080913017 1.5002530290361116 -1.0752215078907055
-0.4416072862946749 0.8954847328983035 -0.32815782928098935
4.2696001611621055 1.9871892439447127 0.5074490898966645

Hidden Layer 4
==========
Sigma 2.621279108084706
Coefficient 25.36356031025891

Centres 0.8140811734506777 1.3768674717636662
1.5319948972946777 2.039702669130082 0.9978631735366212
1.446621450573956 1.6104495040193423 0.43493972854250235
0.9866131107746947 1.9159373944176037 1.5531635253782352
0.9077981886037093 1.7875476625478577 1.1439691301277763
1.0485021084213353 1.0202947301411116 2.629890977405219
0.0522208927110585 1 -0.8509794812041662 0.9518424337807223
1.4753110954748438 0.784954399246679 1.595559926360113
1.91362898358199 -1.0193516250229409 -1.2333636917082567
2.3826838249010422 1.1097001314730008 0.20247181800281955
2.495276670161161 1.434546394942662 0.7868591227001042
1.1950328292572279 2.4532799526561426 -1.0072155532006748
-0.4064184705451967 -1.9192229146930762 2.154921422474241
-0.4967604419439098 0.8417846684157199 0.5124909910338522
-1.027803480682404 -0.175571310173457 0.9942449858396711
3.319150509152954 2.0347463024678167

**BRISTOLCITY - DONCASTER**
**Sum = 2.3367831434231907**
**Epi8imito = 5.0**
**Error 2.6632168565768093**

Hidden Layer 0
==========

Sigma 4.453260560210334
Coefficient 25.433560310259118

Centres 3.5511187725580706 2.0136054839801143
1.990794958554315 1.0282814680956374 -0.05671985786101156
1.0927623662857082 -0.7666095288081762 0.6913773368660721
3.416540616154604 0.34946577348003166 -0.9817129984803211
-0.5420094358715101 -0.1529387303417691 -0.25425001738349934
1.4679207080183565 2.1227344411224327 -1.1174801008132853
0.20834211169675845 0.35851368089971275 0.04500900785186656
3.9690539285869164 1.946325142464398 0.4546816883707607 -
1.1459732880591216 0.18117183866722325 0.2888752292972411
1.6337219342972478 1.811114204927584 1.8195146164683842
4.179052466723951 2.9627442779106774 0.45241960703021344
3.028489957123097 1.2138311272051348 0.6490175055950494
0.7535113593948749 0.33619369400290083 0.004980329081796204
1.4334491652032573 -0.8565559939534678 -0.37775442837337464
0.9950515589193066 -0.21140444497053582 4.220691535708584
1.770685447499199 0.49409701081228513

```
Hidden Layer 1
===============

Sigma 3.8975965003441964
Coefficient      25.45356031025912

Centres    1.8264081577109486   1.5656678006465043
0.8133618971969753   1.9212256259868448   -3.05594143937827
0.5382626653647556   -2.2934596480896743   1.1723704691813828
2.57627597971911   -0.3006496754260639   -0.6925137733115216   -
1.0486551376962279   -0.3704816210667024   -0.08568687964727699
0.8451951997030429   1.1542899612246391   -0.09757056692872294
0.8996917812803047   -1.1902844837404019   -0.18506458543334703
3.540502258065171   1.228103425008355   -0.35270052016173165   -
1.6592550403519086   0.5785393672512315   -2.9555012727389354
0.9245818638006548   0.8142800337650842   1.86614220806942
2.0440235827126485   2.1294381463916903   0.14895147873216144
2.6058443540736866   0.6924307530085743   0.6134747623935423
0.5621477411214106   0.708190262181104   0.0780771925643691
0.6939219649644429   -0.7177461604814368   -0.8358259675138897
0.12967084901752834   -1.3354513021532557   3.8824340182510895
1.4012108515885402   -0.4749418861363014

Hidden Layer 2
===============

Sigma 3.9237118502072437
Coefficient      25.433560310259118

Centres    1.8977255291122872   1.6241355974465819
0.7034628398381149   1.9247922046297765   -2.5075879793044544
0.8023476902798433   -1.9771972718050395   1.1532130133619023
2.6084497590910125   -0.03677223328780892   -0.5941285419738229
-0.9076091028301521   -0.46981551553001993   -
0.18680352382767998   0.953921643031691   1.3762047945906801   -
0.1660994885396595   0.6476742946096978   -1.2855398701460685
0.015581607905343874   3.757857545803354   1.3974371440817335
-0.341690582659748   -1.8988648559269055   0.4676530703499619
-2.543443004414081   1.0367381556386166   0.9881321604686107
2.1600353186735584   2.0553486303530804   2.2287852351904953
0.43365114860942205   2.621801539438346   0.7812192210104604
0.6422044032323397   0.4801652940035705   0.5784196561419079
0.3068745209567694   1.0806400428721357   -0.7718917199380206
-0.7314623797780844   0.17849492264404931   -1.1398030017654959
3.8856370959390114   1.6739078266089369   -0.15496653079502198
```

Hidden Layer 3
==========

Sigma 4.4095529909712265
Coefficient 25.45356031025912

Centres 3.3980629079681934 1.9980131275009954
1.714247760055516 1.098873902969313 -0.676050692432261
1.0541465320547108 -1.1366889571620171 0.659189368753377
3.3750287940772723 0.12118222271585899 -0.9638058284046002
-0.6143190504217512 -0.37888367326706646 -0.698756579589313
1.4601525040262033 2.0260744067399474 -1.0091903576185632
0.17460065432967609 -0.07880892124821 -0.019169809916357687
3.9270395744168907 1.7993701843825196 0.19645345308880421
-1.317513965319494 0.2785378759240438 -0.5842614313948241
1.4728332899867653 1.6258848351437694 1.9625295066602053
3.9289466301486877 2.87817838961223 0.22281597015429516
3.0806193939688704 1.1488544257224218 0.6487729208677168
0.6806767814908057 0.47478806996457784 -0.15942293717507167
1.1996134593553398 -0.9551868340521052 -0.5396465111811819
0.7966536407197037 -0.5572671576885783 4.202617643328921
1.6857011174867664 0.2241017979625276

Hidden Layer 4
==========

Sigma 4.390795039554745
Coefficient 25.353560310259

Centres 3.6201571908022525 2.0927787052059235
1.6946636865888596 0.8176351447784088 -0.07624528811877326
1.0242578465008239 -0.7891141353343923 0.6711230089604711
3.334695730010024 0.4131854331672619 -0.8499564019622601 -
0.40965867477115314 -0.036004416141241884 -
0.15836690982047022 1.2677852157106848 1.8378430266824648
-1.17899827258101 0.2282530664582321 0.343061789338451 -
0.05664668713333279 3.845592263508936 1.9467797192975291
0.4281536195655987 -1.4672845427632633 -0.044963779989955516
0.42216713331798833 1.643585209729353 1.6905601856541268
1.682687423989057 3.992096228098085 2.989045350927051
0.5149612565819057 2.9044323174161084 1.0755592843985513
0.4697064924808662 0.7137275705331176 0.3227265496374817
0.1449825998330294 1.3844952495025205 -1.01178256354201 -
0.5586177462527487 0.9872964732862566 -0.5786148647595007
3.9612300678609635 1.7486741072730991 0.3784101764063876

**COVENTRY - Q.P.R**
**Sum = 2.0263311851958004**
**Epi8imito = 1.0**
**Error -1.0263311851958004**

Hidden Layer 0
==========

Sigma 4.40834953570788
Coefficient 25.433560310259118

Centres 3.341292370764118 1.9332225713110256
1.9773523450483241 1.1895492142793842 -0.5987729415706025
0.9286619800984849 -1.019123853403279 0.7602042811669321
3.4497183926414805 0.09334484723321923 -1.103835548440259
-0.7651930027051411 -0.2926841604211218 -0.5903141061711199
1.509000664594094 2.0923409653719 -0.9081850418037536
0.4437909834980885 0.04811773612870268 0.05350758795868911
3.922141543865332 1.8422248979093134 0.3182496009249688 -
0.9859234952577043 0.39539386943879345 -0.39277394287519213
1.4873423719037242 1.7817517621354821 1.93738960889244
4.028729157019016 2.8524910639703998 0.29270332769888807
3.051368765790504 1.21536676890249 0.7439300797207279
0.8231075607344231 0.5316050996808613 -0.11229880597676546
1.388319872205739 -0.8206591505823159 -0.35713369196316236
0.9026440623408708 -0.1648416534752467 4.383107863383307
1.6896926959729597 0.3779087457375522

Hidden Layer 1
==========

Sigma 1.3197475799779508
Coefficient 25.45356031025912

Centres 0.6297239507830342 0.42835236229255513
0.9444935802735406 -0.01953593012443489 0.5399254935115104
0.1447960199296238 0.39874432039761837 -0.2340561344572179
1.3379644873731589 -0.07955084524925753 0.6266005595551628
-0.4786528631840314 1.261279019426829 0.7409671958748433
1.3061961589217452 0.18678755773407987 0.24418147703264498
0.47268833734049137 0.22612076961843064 0.3807159933226807
0.5920552683797461 0.30026522462382044 0.8880280233635283
0.4113064107993184 0.146144528666855 -0.08460981496663085
0.8988311926302486 1.3524163732283274 0.2911061384995695
1.4569730607239275 1.1877286821643196 -0.08741051559720321
1.3492112862613141 0.7460559632520016 0.6812731996614843
0.7885193466572439 -0.30313035181458176 1.314671292011567
0.84515565041006 1.2472212862986 -0.012280434484089359
0.24899391944426907 1.1985113464922792 0.08408742870618675
0.7610173484505659 0.01655522164513621

Hidden Layer 2
==========

Sigma 4.454916857029935
Coefficient 25.433560310259118

Centres 3.418833870544282 2.0451470333808595
1.4108943554095708 0.7128399429360358 -0.7674631009463281
0.8489704381325897 -1.3536024068762007 0.7421454490477672
3.298396376326257 0.08854063288522243 -1.0219853756295856
-0.7912553775063259 -0.21206161087548606 -0.7897670944317188
1.3174756898119 1.742439600344895 -1.0364551899116095
0.48456123648637334 -0.42545291031711097 -0.2404126127842812
3.7098332918532844 1.7885731513872154 0.08999727716607007
-1.6470565537746145 0.028854502784238793 -
0.40542430915052613 1.3909585646167304 1.7711827644553138
1.8790087478611612 3.660143709906204 2.9820066687632836
0.570242488786018 3.077448736933608 1.0074733034500343
0.26754150516318276 0.6501064707442714 0.3881913949310027
0.3746286437604356 1.3702304224042385 -1.2017186003014977
-0.9547884563554209 0.8193343190273937 -0.6382638352841856
4.154633512934489 1.8858754011657632 0.16475538886907481

Hidden Layer 3
==========

Sigma 1.3481199677833722
Coefficient 25.45356031025912

Centres 1.072629293258609 1.1789765016685316
1.2007039719769153 -0.41677644858648416 -0.16820191261991504
1.0276797575053098 0.8720329298167545 -0.138317292602234
0.059275267453993655 -0.20938802674343793 0.9351497958391017
-0.13602778084694456 1.3258292627785297 0.4709270197190676 4
0.9489954280145813 0.28346615001892905 0.2215015915873712
0.9559819014819109 0.7245729726802251 0.39808212537980714
0.1319038016792525 1.1474715944527651 1.2892529391230354
1.5966413630341005 -0.2685104824237938 0.8506689123495386
0.7041711405566634 0.9418808702860111 -0.193778224396292
1.0879083252334616 1.0221294701947081 -0.21091837828901325
1.6317139929619544 0.3498493089800036 -0.2885437360158362
1.115319575110828 -0.5440211836063389 0.9583021900447435
0.15010107809371828 0.48148424638260984 0.20501243719430012
0.2514472654607645 1.4252360095282548 0.31076055840167677
0.414427844642247 -0.17140317580079323

Hidden Layer 4
==========

Sigma 2.59360678743211
Coefficient 25.353560310259

Centres 2.7563469319981273 0.2832840494434107
0.37806175330966546 1.591905959585398 -0.6384756108165696
-0.05945497764741628 0.11878762254278634 0.14880888627474995
0.9859651282788758 0.54476934187467 1.3141472970579284 -
0.06342772740610837 1.5400650977810721 0.1325814632391517
0.9891465996939008 -1.891642785158827 1.474699265977198 -
0.05167676667904806 1.239324854957927 -0.14408524462672015
2.27339223262078 1.7697537151808582 -0.055872432598306404
-1.082394302078447 1.234977536150602 -0.3615380235396655
0.8764688602977486 1.7224181504458465 0.6813336428595752
1.690072078547316 1.5603732061922098 1.659940413263763
2.758516251497857 0.5992612879862499 0.752092267115693
1.3991049059537404 -1.2354630251457137 0.6128670735790176
0.34576849400841236 0.18100758115781126 -0.8865219993184539
1.1928630757482463 -0.5611848011713382 1.7768127956258
2.706491512790024 -0.599555744519818

**CRYSTALPALACE - PLYMOUTH**
**Sum = 1.5452367481483005**
**Epi8imito = 3.0**
**Error 1.4547632518516995**

Hidden Layer 0
==========

Sigma 2.0050545190719213
Coefficient 25.70356031025934

Centres 0.0956096403117049 0.6614433083926738
1.0490146749048601 0.6315130533780499 1.0319673717716307
0.7404051875622386 0.9350755349267129 0.8242286537946731
0.15815829418708335 0.22437634489709088 1.1344851391294795
0.24516296486636138 0.1401485218046668 0.44284992428896974
0.5302226486386058 1.2678822548354771 0.9679866675294656
0.14323252948982454 1.3020312629739763 0.9683740838519505
1.0202483825635165 -0.2735188610337657 0.885222715333881
0.45509149265768256 0.7909406886097557 0.9763405308148426
1.6944957624778574 1.711739391605285 1.6340962559692123
0.6186506089276076 -0.2887705983027391 1.4566702467572419
0.4581115953372245 -1.3912224580053687 -3.78222329112292
0.022509911010818635 1.8995915685566538 2.0544968616491617
0.4518125024990838 0.6184355143595547 -0.4902283873708568
-1.4391184184111703 0.765752524080303 2.129170757182523
1.0799178185013765 1.0816070001569507

Hidden Layer 1
==========

Sigma 3.9113203356820505
Coefficient 25.733560310259406

Centres 1.5561090555800676 1.6879887114811805
0.45124803967400173 2.2145670331783247 -3.3472063305718014
0.7670185225760462 -2.494675984450443 1.3144377537510243
2.440726416017404 -0.34970516788496714 -0.3913741687776543
-0.8482983540614532 -0.4467311665606703 -0.05402485102833075
0.6375787002275609 1.044529185553874 0.06118873129113834
0.672777937631802 -1.648582378497454 -0.2586167661603688
3.7026025346071627 1.044119782494339 -0.5871488109926805 -
1.8132714415109785 0.6862489908885427 -3.113942071044145
0.9957355077181174 0.7432109062323496 2.1993804031061357
1.6109904025809085 2.0943886859286986 0.45955598710005924
2.7023166189083545 0.6084435485461421 0.7065478587782847
0.2943284281821744 0.5173274384381719 0.283695489698964
0.7459366333365629 -0.45406428288255524 -0.5483843497905598
-0.040973298564906516 -1.3471210422781121 4.100465465975155
1.5874968917252748 -0.4834809866593325

Hidden Layer 2
==========

Sigma 3.158273519508864
Coefficient 25.45356031025912

Centres 3.021794510068758 2.4804722796916723
1.5996018466622748 2.4573233326195543 0.07399045274181502
1.163231021178977 -0.6652257751831354 0.6677278991777916
2.3493286280608703 -0.44159175711613113 1.5845042471578366
1.8352484700611904 -0.21289647019969501 1.0261442018073221
-0.21965212690620475 -0.736067254651521 -0.6148392543215266
-1.9460529899387202 1.1039945531669981 -1.821754883097143
3.7460516253504585 0.938371424752219 -0.15231570439252676
-0.9011772220270658 -0.041252733442875916 0.5048165011622427
1.7025277146126616 0.18289286316523407 0.3842853810218024
2.4524435029460547 1.9936399856536366 0.30227710529580226
1.3915979609042688 1.3271033743191463 1.4434027706278394
0.8022367893762102 0.35947018337019737 -0.5319672326143063
-2.0309267694173414 0.8637267866433582 0.9627584038628262
1.2186257781407694 -2.040089117026505 2.5355429185690084
0.8033510956112483 0.08462613856428293

Hidden Layer 3
==========

Sigma 4.089199484076662
Coefficient 25.37356031025901

Centres 2.5451673584646963 1.6872786522790668
1.3965141995136914 1.8559216223325372 -3.016792634094758
0.3644474030132752 -2.272793785710891 0.8293057938961513
3.069642940457839 -0.6484154078950858 -1.1113520421896532
-1.2208259668798611 -0.5594272402218772 -0.7789336508076338
1.2337147046584576 1.608754234118987 -0.22012679615215786
1.080223069686581 -1.037228443380992 -0.10496818799859635
3.5638893757461507 1.271748894767473 -0.2802298572574262 -
1.3268715806086118 0.6893003835305049 -3.020235934857408
1.060194528825555 0.7512362572045218 1.6216437922492488
3.0200522196454718 2.1899951476488333 -0.7451579362613338
2.633082600522963 0.6268983698957129 0.6763191413088444
1.1947431312761247 1.48673763225359 -0.6719656484225403
0.5651295023542764 -0.8066737594719803 -0.9632522580870981
0.09816112780871962 -1.0813577250948594 3.9286178335489566
0.9427131339403995 -0.6657464440779364

Hidden Layer 4
==========

Sigma 4.153767002029213
Coefficient 25.39356031025912

Centres 2.9423773699544578 1.7443361801985087
2.2211600475771736 1.3545267427148329 -0.349343079545805
1.0588759955371756 -0.2238136730323879 0.9297460141816257
3.316561104567729 0.46003970565157587 -0.7141727440625635
-0.4244430985768242 -0.16684181120569766 0.12229693784508133
1.3348527918656028 2.26087263051896 -0.6205296376756053
0.35954626522206096 0.6714610133699312 0.29326302436817697
4.069850802529411 1.9360186841370972 0.6467585346100012 -
0.23702177127230104 0.8814671394147418 -0.4183746120643937
1.2496702339271326 1.8548608956776598 2.0059377938206233
3.4018235564762693 2.6656383918832987 0.6338489618058598
2.8091649115790407 1.7778541542860553 1.5612604965768588
0.12626003055057183 -0.2108556544382892 -0.08857281322507808
1.3710001089531436 -0.7012776036656682 0.32540970438349126
0.6538342517126833 -0.2594738306298798 4.076148133444171
1.7514490636270306 0.3571564906865011