Ατομική Διπλωματική Εργασία

Πρόβλεψη Ποδοσφαιρικών Αγώνων: Εφαρμογή του Αλγορίθμου Hessian-Free Optimization για την Εκτίμηση Over/Under 2,5 Τερμάτων

Μιχάλης Ξυδιάς

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2025

Περιεχόμενο

Περίληψη

Η Διπλωματική μου εργασία έχει ως σκοπό την υλοποίηση ενός τεχνητού νευρωνικού δικτύου στο οποίο θα προβλέπεται ο αριθμός των τερμάτων που θα επιτευχθούν σε ένα ποδοσφαιρικό αγώνα, με επακόλουθο το κέρδος έπειτα από στοιχηματική δραστηριότητα.

Δηλαδή, θα λαμβάνονται, ως είσοδος σε αυτό, δεδομένα και στατιστικά από προηγούμενους αγώνες για τις δύο διαγωνιζόμενες ομάδες, όπως παραδείγματος χάριν ο μέσος όρος τερμάτων υπέρ και κατά, το ποσοστό αγώνων διατήρησης ανέπαφης εστίας, το ποσοστό αγώνων που σημειώθηκαν άνω των δυόμιση τερμάτων κ.α. Τα στατιστικά αυτά θα συνυπολογίζονται και θα προβλέπεται το κατά πόσον ο αριθμός των τερμάτων που θα επιτευχθούν σε έκαστο, μελλοντικό, ποδοσφαιρικό αγώνα θα υπερβαίνουν ή όχι τον αριθμό δυόμιση. Η τιμή αυτή αποτελεί σημείο στοιχήματος στα πρακτορεία, το λεγόμενο under-over.

Διπλωματικές εργασίες παρόμοιες με την δική μου έγιναν και από άλλους προπτυχιακούς φοιτητές με τον καθένα να προσθέτει, μέσω των πορισμάτων της εργασίας του, την δική του γνώση η οποία με βοήθησε να αναπτύξω την δική μου διπλωματική.

Για την υλοποίηση και τον τρόπο επίλυσης του προβλήματος αποφασίστηκε, σε συνεργασία με τον επιβλέποντα καθηγητή, να αξιοποιηθούν σύγχρονες τεχνικές μηχανικής μάθησης, καθώς έχουν αποδειχθεί ιδιαίτερα αποτελεσματικές στην επίλυση σύνθετων προβλημάτων πρόβλεψης. Η επιλογή της χρήσης νευρωνικών δικτύων βασίστηκε στην ικανότητά τους να αντλούν και να μαθαίνουν μοτίβα από δεδομένα με μεγάλη ποικιλομορφία και πολυπλοκότητα, όπως αυτά που συναντώνται στον χώρο του ποδοσφαίρου. Τα συστήματα αυτά μπορούν να προσαρμόζονται σε νέες συνθήκες και να παράγουν εκτιμήσεις, ακόμη και όταν δεν υπάρχει σαφής ή προφανής γραμμική συσχέτιση στα δεδομένα. Με δεδομένη την πρόκληση που παρουσιάζει το συγκεκριμένο πρόβλημα αφού η πρόβλεψη του αριθμού των τερμάτων ή της πιθανότητας για over/under δεν είναι εύκολη υπόθεση κρίθηκε απαραίτητη η χρήση αλγορίθμων που προσφέρουν υψηλό βαθμό ευελιξίας και ικανότητα μάθησης από τα δεδομένα, προκειμένου να επιτευχθεί η καλύτερη δυνατή απόδοση.

Οι τεχνικές που εφαρμόστηκαν στην παρούσα εργασία είναι τα τεχνητά νευρωνικά δίκτυα με επιβλεπόμενη μάθηση, όπως το Multilayer Perceptron (MLP) με backpropagation, τα Radial Basis Function (RBF) δίκτυα με κινητά κέντρα, καθώς και το Hessian Free Optimization (HFO). Σε κάθε περίπτωση, η εκπαίδευση των μοντέλων πραγματοποιείται μέσω επιβλεπόμενης μάθησης (supervised learning), κατά την οποία το δίκτυο λαμβάνει ως είσοδο τα στατιστικά των ομάδων πριν από κάθε αγώνα, ενώ ταυτόχρονα του παρέχεται και η επιθυμητή έξοδος (target value), που αντιστοιχεί στον συνολικό αριθμό των τερμάτων του εκάστοτε αγώνα. Ο στόχος του δικτύου είναι να μάθει να προσεγγίζει όσο το δυνατόν πιο αποτελεσματικά αυτή την έξοδο, μειώνοντας σιγά-σιγά το σφάλμα πρόβλεψης. Η ακριβής πρόβλεψη του πλήθους των τερμάτων επιτρέπει την εξαγωγή συμπερασμάτων για τον στοιχηματικό τύπο Over/Under 2.5, γεγονός που καθιστά το σύστημα χρήσιμο για στοχευμένες προβλέψεις με πρακτική αξία.

Το πρώτο στάδιο της παρούσας εργασίας αφορούσε την άντληση και επεξεργασία δεδομένων από την ιστοσελίδα www.football-data.co.uk, η οποία παρέχει σε μορφή CSV αποτελέσματα και στατιστικά για το αγγλικό πρωτάθλημα Championship (Πρωτάθλημα 2ης κατηγορίας του Αγγλικού ποδοσφαίρου) για κάθε αγωνιστική περίοδο. Η επεξεργασία των δεδομένων πραγματοποιήθηκε στη γλώσσα Java, όπου δημιουργήθηκε ένα αυτόματο σύστημα ανάγνωσης των αρχείων, με στόχο την εξαγωγή χρήσιμων στατιστικών χαρακτηριστικών για κάθε ομάδα και κάθε αγώνα. Συγκεκριμένα, για κάθε αγωνιστική και κάθε ομάδα δημιουργείται ένας πίνακας με τα στατιστικά μέχρι εκείνο το σημείο της σεζόν, όπως μέσος όρος τερμάτων υπέρ και κατά, ποσοστά over/under, αριθμός συνεχόμενων αγώνων με συγκεκριμένα αποτελέσματα κ.ά.

Τα αποτελέσματα της εργασίας μου κρίνονται ιδιαίτερα ενθαρρυντικά, καθώς και οι τρεις υλοποιημένοι αλγόριθμοι (MLP με backpropagation, RBF με κινητά κέντρα και HFO) παρουσίασαν ικανοποιητική απόδοση. Ιδιαίτερα ο αλγόριθμος Hessian Free Optimization (HFO) ξεχώρισε, καθώς εμφάνισε πολύ ικανοποιητικά αποτελέσματα στην πρόβλεψη αποτελεσμάτων αγώνων σε δεδομένα που δεν είχε επεξεργαστεί κατά την εκπαίδευση. Η συμπεριφορά του HFO αποδείχθηκε πιο σταθερή και αξιόπιστη σε σχέση με τους υπόλοιπους αλγορίθμους της εργασίας και έδειξε ανώτερη ικανότητα γενίκευσης. Επιπλέον, συγκρινόμενος με προηγούμενες προσεγγίσεις που έχουν υλοποιηθεί στην ίδια θεματική, ο HFO φάνηκε να προσαρμόζεται καλύτερα στο συγκεκριμένο πρόβλημα και να προσφέρει πιο συνεπείς προβλέψεις.

Κάθε μοντέλο εκπαιδεύτηκε με δεδομένα από τις αγωνιστικές περιόδους 2006 έως και 2023, ενώ για τον έλεγχο της ακρίβειας των προβλέψεων χρησιμοποιήθηκε η σεζόν 2024. Το πολυεπίπεδο perceptron με backpropagation (MLP) παρουσίασε ακρίβεια της τάξης του 84% στα δεδομένα εκπαίδευσης και 70% στα δεδομένα ελέγχου, δείχνοντας καλή ικανότητα γενίκευσης. Το RBF δίκτυο, χρησιμοποιώντας κινητά κέντρα, κατέγραψε ακρίβεια 73% στην εκπαίδευση και 71% στα δεδομένα ελέγχου. Τέλος, το δίκτυο με Hessian-Free Optimization (HFO) επέδειξε την πιο σταθερή και αξιόπιστη συμπεριφορά, φτάνοντας ακρίβεια 81% κατά την εκπαίδευση και 76% στον έλεγχο, επιβεβαιώνοντας την αποτελεσματικότητά του συγκριτικά με τις άλλες προσεγγίσεις και προηγούμενες εργασίες στο ίδιο πεδίο.

Κεφάλαιο 1

Εισαγωγή

1.1 Στόχος

Σκοπός της παρούσας εργασίας είναι η υλοποίηση και σύγκριση τριών διαφορετικών τεχνικών νευρωνικών δικτύων με στόχο την πρόβλεψη του πλήθους των τερμάτων που θα σημειωθούν σε έναν ποδοσφαιρικό αγώνα στο πλαίσιο του αγγλικού πρωταθλήματος Championship. Η πρόβλεψη αυτή αφορά την εκτίμηση του αν ο συνολικός αριθμός των τερμάτων θα υπερβεί ή όχι ένα προκαθορισμένο όριο (over/under 2.5 goals), στοιχείο που αποτελεί βασικό σημείο στοιχηματισμού.

Στην εργασία αυτή αξιοποιήθηκαν τρεις αρχιτεκτονικές: ένα πολυεπίπεδο perceptron (MLP) εκπαιδευόμενο με backpropagation, ένα δίκτυο RBF με κινητά κέντρα, καθώς και ένα δίκτυο βασισμένο στη μέθοδο Hessian-Free Optimization (HFO). Η επιλογή αυτών των μοντέλων αποσκοπεί στη σύγκριση των αποτελεσμάτων τους και στην αξιολόγηση της ικανότητάς τους να γενικεύουν σε νέα, μη γνωστά δεδομένα.

Κύριος στόχος της διπλωματικής μου είναι η δημιουργία ενός αξιόπιστου εργαλείου πρόβλεψης, το οποίο θα μπορούσε να αξιοποιηθεί για την αύξηση του ποσοστού επιτυχίας παικτών που επιλέγουν στοιχήματα βάσει στατιστικής ανάλυσης, προσφέροντας μια ένδειξη εμπιστοσύνης μέσα από την ακρίβεια των προβλέψεων του συστήματος.

1.2 Προηγούμενες προσπάθειες

Πριν ξεκινήσει η υλοποίηση της δικής μου πρότασης, κρίθηκε απαραίτητο να προηγηθεί μια αναλυτική μελέτη των προσπαθειών που έχουν προηγηθεί στον ίδιο ή παρόμοιο ερευνητικό άξονα. Αυτό μου επέτρεψε να κατανοήσω ποιες τεχνικές έχουν εφαρμοστεί, τι ποσοστά επιτυχίας είχαν, ποια μεθοδολογία ακολουθήθηκε στην επιλογή και επεξεργασία των δεδομένων, καθώς και τι προκλήσεις αντιμετωπίστηκαν. Επιπλέον, αποτέλεσε βάση για να συγκρίνω αργότερα τα δικά μου αποτελέσματα με προηγούμενες προσεγγίσεις και να εντοπίσω τις δικές μου συνεισφορές.

Η πρώτη τεκμηριωμένη προσπάθεια εντοπίζεται το 2001 από τον Matt Jackson του πανεπιστημίου Birkbeck του Λονδίνου. Ο Jackson προσπάθησε να προβλέψει το τελικό αποτέλεσμα ενός ποδοσφαιρικού αγώνα (νίκη-ισοπαλία-ήττα) χρησιμοποιώντας Τεχνητά Νευρωνικά Δίκτυα και συγκεκριμένα τεχνικές όπως το Kohonen SOM (για ομαδοποίηση) και Back Propagation (για μάθηση). Τα αποτελέσματα που σημείωσε ήταν 47% για το Back Propagation και 57% για το Kohonen. Ωστόσο, τα αποτελέσματα αυτά οφείλονταν κυρίως στην πρόβλεψη νίκης του γηπεδούχου, η οποία στατιστικά είναι η πιο συχνή έκβαση, αλλά στοιχηματικά η λιγότερο αποδοτική, καθώς οι αποδόσεις είναι χαμηλές [1].

Ακολούθως, το 2004, ο Kevin Davies, επίσης του Birkbeck, επιχείρησε να συνεχίσει και να βελτιώσει τα αποτελέσματα της προηγούμενης εργασίας. Η βασική διαφορά στην προσέγγισή του ήταν η μεγαλύτερη έμφαση στην προεπεξεργασία των δεδομένων, ώστε να αποκτήσουν πιο «ευανάγνωστη» μορφή για το δίκτυο. Παρόλο που και ο ίδιος χρησιμοποίησε Back Propagation, τα ποσοστά επιτυχίας του δεν διαφοροποιήθηκαν σημαντικά από εκείνα του Jackson. Η αύξηση στο στοιχηματικό κέρδος παρέμεινε οριακή, με το σύστημα να αντιμετωπίζει προβλήματα γενίκευσης [2].

Η Sian Turner, το 2005, προχώρησε σε μια πιο ενδιαφέρουσα διαφοροποίηση: εγκατέλειψε την πρόβλεψη του τελικού αποτελέσματος και επικεντρώθηκε στο στοίχημα τύπου “More-Less”, δηλαδή αν θα σημειωθούν πάνω ή κάτω από 2.5 γκολ σε έναν αγώνα. Χρησιμοποιώντας επίσης Back Propagation και δίνοντας ιδιαίτερη σημασία στην επεξεργασία των δεδομένων, πέτυχε ποσοστό πρόβλεψης 57%. Αν και το ποσοστό αυτό

δεν εντυπωσιάζει, αποτέλεσε ένδειξη ότι η στροφή προς το More-Less είναι ελπιδοφόρα [3].

Το 2007, ο Θεοφάνης Νεοκλέους, φοιτητής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου, υλοποίησε ένα πλήρες σύστημα πρόβλεψης More-Less χρησιμοποιώντας πάλι Back Propagation. Μέσα από πολλούς πειραματισμούς και δοκιμές στην επιλογή χαρακτηριστικών και στη ρύθμιση του δικτύου, έφτασε σε ακρίβεια πρόβλεψης 62% και κατάφερε να πετύχει μέχρι και 23% αύξηση στο υποθετικό κέρδος από στοιχήματα, επιβεβαιώνοντας την αποτελεσματικότητα της τεχνικής [4].

Η επόμενη και πιο εντυπωσιακή προσπάθεια μέχρι τότε πραγματοποιήθηκε από τον Ανδρέα Ιακώβου. Ο Ιακώβου ήταν ο πρώτος που εφάρμοσε Radial Basis Function (RBF) Νευρωνικά Δίκτυα στο πρόβλημα πρόβλεψης ποδοσφαιρικών αγώνων. Με στόχο επίσης την κατηγοριοποίηση ως προς το More-Less, το RBF δίκτυο του πέτυχε ακρίβεια 68%, ενώ παράλληλα χρησιμοποίησε και Support Vector Machines (SVM), όπου η απόδοση έφτασε το 70%. Το ποσοστό αυτό θεωρήθηκε εξαιρετικά ικανοποιητικό και με σημαντικές προοπτικές για κέρδος, εφόσον αξιοποιηθεί στοιχηματικά [5].

Το 2016, ο φοιτητής Αντρέας Φράγκου ασχολήθηκε με την πρόβλεψη του στοιχήματος over/under 2.5 σε αγώνες ποδοσφαίρου, χρησιμοποιώντας Recurrent Neural Networks (RNN). Κατά την εκπαίδευση, τα στατιστικά κάθε αγώνα συνοδεύονταν από την επιθυμητή έξοδο (over ή under), και το RNN μάθαινε να προβλέπει βάσει αυτών. Η ακρίβεια των προβλέψεων του πρώτου μοντέλου RNN ήταν 57%, ενώ το δεύτερο έφτασε έως και το 70% στα δεδομένα εκπαίδευσης και 60% στα δεδομένα ελέγχου. Η προσέγγισή του αποτέλεσε σημαντικό βήμα προς τη χρήση χρονοσειρών και επαναλαμβανόμενων δομών στο πρόβλημα της στοιχηματικής πρόβλεψης [6].

Πέρα από τις πιο παλιές ακαδημαϊκές προσπάθειες και τις προγενέστερες διπλωματικές εργασίες, υπάρχουν και αρκετές πρόσφατες επιστημονικές δημοσιεύσεις που ασχολούνται με τη μοντελοποίηση και πρόβλεψη αποτελεσμάτων ποδοσφαιρικών αγώνων μέσω τεχνικών μηχανικής μάθησης και στατιστικής.

Ο Constantinou και ο Fenton (2018) [7] ανέπτυξαν ένα Bayesian δίκτυο που συνδυάζει τη στατιστική μοντελοποίηση με μεθόδους υπολογισμού αβεβαιότητας, επιτρέποντας όχι μόνο την πρόβλεψη αποτελέσματος αλλά και την αξιολόγηση της αξιοπιστίας της πρόβλεψης. Το μοντέλο εφαρμόστηκε σε αγώνες ποδοσφαίρου και παρουσίασε ανώτερη απόδοση σε σχέση με πιο κλασικά προγνωστικά συστήματα, καθιστώντας το ιδανικό για περιβάλλοντα με υψηλή μεταβλητότητα, όπως ο στοιχηματισμός. Το σύστημα επικεντρωνόταν στην πρόβλεψη του τελικού αποτελέσματος (νίκη γηπεδούχου, ισοπαλία ή νίκη φιλοξενούμενου), αποδίδοντας πιθανότητες σε κάθε έκβαση βάσει των διαθέσιμων δεδομένων και της σχετικής αβεβαιότητας.

Οι Groll et al. (2022) [8] πραγματοποίησαν μία συγκριτική αξιολόγηση μοντέλων μηχανικής μάθησης για πρόβλεψη ποδοσφαιρικών αποτελεσμάτων, χρησιμοποιώντας δεδομένα από διεθνή τουρνουά (π.χ. EURO, Παγκόσμιο Κύπελλο). Εφάρμοσαν μοντέλα όπως random forests, lasso regression και Poisson regression για την πρόβλεψη σκορ και κατάρτισαν σενάρια για την εξέλιξη ολόκληρων διοργανώσεων. Η εργασία προσφέρει μια πολύπλευρη οπτική στην πρακτική εφαρμογή μοντέλων πρόβλεψης σε μεγάλης κλίμακας πρωταθλήματα. Ένα από τα βασικά ευρήματα ήταν ότι τα μοντέλα Poisson regression απέδωσαν καλύτερα στην πρόβλεψη του τελικού σκορ, ενώ τα δέντρα αποφάσεων (random forests) ήταν πιο ευέλικτα στην ανίχνευση πολύπλοκων σχέσεων μεταξύ μεταβλητών. Επιπλέον, η μελέτη ανέδειξε τη σημασία της σωστής επιλογής χαρακτηριστικών και της ενσωμάτωσης εξωτερικών παραγόντων (π.χ. βαθμολογικής σημασίας αγώνα) για τη βελτίωση της ακρίβειας.

Οι Rischard et al. (2024) [9] παρουσίασαν ένα σύστημα βασισμένο σε meta-learning (μετα-μάθηση), το οποίο επιλέγει δυναμικά τον πιο κατάλληλο αλγόριθμο πρόβλεψης ανάλογα με τα χαρακτηριστικά του κάθε dataset. Το σύστημα εφαρμόστηκε στην πρόβλεψη αποτελεσμάτων ποδοσφαιρικών αγώνων, με στόχο τη βελτιστοποίηση της ακρίβειας μέσω αυτόματης προσαρμογής σε διαφορετικού τύπου δεδομένα (π.χ. στατιστικά ομάδων, ιστορικά αποτελέσματα, βαθμολογική θέση). Η εργασία αυτή είναι σημαντική γιατί δεν χρησιμοποιεί μόνο ένα μοντέλο πρόβλεψης, αλλά προσαρμόζεται κάθε φορά επιλέγοντας το καταλληλότερο, ανάλογα με τα δεδομένα. Με αυτόν τον

τρόπο, μπορεί να δίνει πιο αξιόπιστες προβλέψεις, ακόμα και όταν τα δεδομένα προέρχονται από διαφορετικές διοργανώσεις ή έχουν μεγάλες διαφορές μεταξύ τους.

Ακολουθώντας την πορεία όλων αυτών των εργασιών, η παρούσα διπλωματική εργασία στοχεύει όχι μόνο να επιβεβαιώσει τα αποτελέσματα προηγούμενων ερευνών αλλά και να προτείνει μια πιο σύγχρονη και πολυδιάστατη προσέγγιση. Για τον σκοπό αυτό υλοποιήθηκαν και συγκρίθηκαν τρεις διαφορετικοί αλγόριθμοι νευρωνικών δικτύων.

Συνοψίζοντας, οι προηγούμενες προσπάθειες έθεσαν τα θεμέλια για την κατανόηση της πρόβλεψης αγώνων μέσω νευρωνικών δικτύων, δοκιμάζοντας διάφορες τεχνικές και αναδεικνύοντας τη σημασία της σωστής αναπαράστασης των στατιστικών. Η παρούσα διπλωματική εργασία υλοποίησε και συνέκρινε τρεις αλγορίθμους νευρωνικών δικτύων για την πρόβλεψη του over/under 2.5. Συγκεκριμένα, εφαρμόστηκε το MLP με backpropagation, RBF με κινητά που παρουσίασαν ικανοποιητική απόδοση και σταθερότητα, και τέλος, ο αλγόριθμος Hessian-Free Optimization (HFO), που επέδειξε τη σταθερότερη και πιο αξιόπιστη συμπεριφορά στο σύνολο των δοκιμών, ξεπερνώντας σε απόδοση και προηγούμενες προσεγγίσεις που έχουν υλοποιηθεί για το ίδιο πρόβλημα.

Κεφάλαιο 2

Υπόβαθρο

2.1 Νευρωνικά δίκτυα – Ιστορική Αναδρομή

Τα νευρωνικά δίκτυα παρουσιάστηκαν για πρώτη φορά ως θεωρητική έννοια το 1943 από τους McCullock και Pitts. Οι δύο ερευνητές υποστήριξαν ότι ο ανθρώπινος εγκέφαλος αποτελείται από έναν μεγάλο αριθμό απλών, διασυνδεδεμένων νευρώνων, οι οποίοι συνεργάζονται μεταξύ τους με τρόπο που μπορεί να προσομοιωθεί από ένα σύστημα παρόμοιο με ηλεκτρικό κύκλωμα, με στόχο την αποτύπωση και αναπαράσταση ανθρώπινων λειτουργιών.[10]

Οι McCullock και Pitts, ένας νευροφυσιολόγος και ένας μαθηματικός αντίστοιχα, άνοιξαν τον δρόμο για την ανάπτυξη των τεχνητών νευρωνικών δικτύων παρουσιάζοντας ένα πρότυπο όπου κάθε νευρώνας έχει τη δυνατότητα να δέχεται πολλαπλές εισόδους και να παράγει μία έξοδο με δυαδική μορφή (ενεργός ή ανενεργός). Το μοντέλο τους βασιζόταν στη λογική ότι οι είσοδοι ενός νευρώνα, εφόσον ξεπερνούν ένα ορισμένο όριο, προκαλούν ενεργοποίηση και αποστολή σήματος στους επόμενους νευρώνες του δικτύου. Με αυτόν τον τρόπο, οι έξοδοι των νευρώνων τροφοδοτούν ως είσοδοι τους

επόμενους, διαμορφώνοντας ένα πολύπλοκο σύστημα πληροφορικής ροής. Η ιδέα τους αυτή αποτέλεσε τη βάση για την προσομοίωση της εγκεφαλικής λειτουργίας μέσω ηλεκτρονικών κυκλωμάτων και τεχνητών νευρωνικών δομών, καθορίζοντας την πορεία της μελλοντικής έρευνας στον τομέα της υπολογιστικής νοημοσύνης.[11][12]

Όταν η πληροφορία φτάσει στην τελική έξοδο ενός νευρωνικού δικτύου, ενεργοποιείται ένας μηχανισμός ανάδρασης, κατά τον οποίο η έξοδος ενός νευρώνα επηρεάζει ξανά τις εισόδους του, μέσω της ενίσχυσης ή αποδυνάμωσης των μεταξύ τους συνδέσεων. Η διαδικασία αυτή μεταβάλλει τη «δύναμη» ή αλλιώς το βάρος που κάθε είσοδος ασκεί στο τελικό αποτέλεσμα του νευρώνα. Με αυτόν τον τρόπο, το δίκτυο μαθαίνει από την εμπειρία και αποθηκεύει πληροφορία, καθώς αυτή η δυναμική ρύθμιση των βαρών καθορίζει τη μνήμη και τη συμπεριφορά του δικτύου σε μελλοντικές εισόδους.

Ωστόσο, αυτή η σημαντική θεωρητική πρόοδος ίσως να μην είχε ποτέ πρακτική εφαρμογή, αν ο J. Von Neumann δεν είχε τη διορατικότητα να συνδέσει την αρχιτεκτονική των βιολογικών νευρωνικών συστημάτων με την ταχύτατα εξελισσόμενη τεχνολογία των ηλεκτρονικών υπολογιστών της δεκαετίας του 1950. Ήταν η δική του οπτική που ενέπνευσε την ιδέα ότι η λειτουργία του ανθρώπινου εγκεφάλου θα μπορούσε να προσομοιωθεί από μηχανές, οδηγώντας έτσι στα πρώτα τεχνητά νευρωνικά δίκτυα. Τα δίκτυα αυτά αποτέλεσαν μια απόπειρα να μεταφερθούν οι αρχές της βιολογικής μάθησης και επεξεργασίας πληροφορίας στο πεδίο των υπολογιστικών συστημάτων.[13]

Με την εξάπλωση της ιδέας των τεχνητών νευρωνικών δικτύων, άρχισαν να κάνουν την εμφάνισή τους τα πρώτα θεμελιώδη έργα που εξακολουθούν να αποτελούν τη βάση για τη μελέτη και την ανάπτυξη των σύγχρονων συστημάτων. Ένα από τα σημαντικότερα ήταν το βιβλίο του D. Hebb, με τίτλο *"The Organization of Behavior"* το 1949, το οποίο διατύπωσε μια καθοριστική θεωρία για το πώς δημιουργούνται και ενισχύονται οι συνδέσεις μεταξύ των νευρώνων κατά την εκπαίδευση ενός δικτύου. Μέσα από πειράματα στη νευροφυσιολογία, ο Hebb κατέληξε στο συμπέρασμα ότι οι συνάψεις ενισχύονται όταν μια είσοδος συμβάλλει σταθερά στην ενεργοποίηση ενός νευρώνα. Με άλλα λόγια, η σύνδεση μεταξύ ενός ενεργού νευρώνα και μιας εισόδου που τον πυροδοτεί ενισχύεται με την επανάληψη, κάτι που αποτελεί τη βάση της διαδικασίας μάθησης και μνήμης στα τεχνητά νευρωνικά δίκτυα. Αυτή η αρχή είναι σήμερα γνωστή ως “Hebbian learning” και θεωρείται θεμέλιος λίθος στη θεωρία της εκπαίδευσης των δικτύων.[14]

Καθώς η έρευνα στα τεχνητά νευρωνικά δίκτυα προχωρούσε, άρχισαν να εμφανίζονται και τα πρώτα λειτουργικά μοντέλα. Το 1957, ο Frank Rosenblatt παρουσίασε το πρώτο πρακτικό νευρωνικό δίκτυο, γνωστό ως Perceptron, το οποίο υλοποιήθηκε σε πραγματικό hardware. Η δυνατότητά του να επιλύει διάφορα προβλήματα δημιούργησε ενθουσιασμό στην επιστημονική κοινότητα, καθώς πολλοί πίστεψαν πως επρόκειτο για

την αρχή μιας νέας εποχής στην τεχνητή νοημοσύνη. Ωστόσο, αυτή η αισιοδοξία αποδείχθηκε πρόωρη. Μεταγενέστερες μελέτες αποκάλυψαν σημαντικούς περιορισμούς του Perceptron, όπως η αδυναμία του να επιλύει μη γραμμικά διαχωρίσιμα προβλήματα, κάτι που περιόρισε αισθητά τις πρακτικές του εφαρμογές και ανέκοψε την αρχική του δυναμική.[15]

Μια πιο σε βάθος ανάλυση του perceptron πραγματοποιήθηκε από τους Marvin Minsky και Seymour Papert στο βιβλίο τους με τίτλο "Perceptrons". Οι δύο ερευνητές παρουσίασαν τόσο τις δυνατότητες όσο και τους περιορισμούς του συγκεκριμένου μοντέλου, τεκμηριώνοντάς τους με αυστηρή μαθηματική απόδειξη. Με την εργασία τους ανέδειξαν το περιορισμένο φάσμα προβλημάτων που μπορούσε να επιλύσει το perceptron, γεγονός που οδήγησε τη διεθνή ερευνητική κοινότητα να απομακρυνθεί προσωρινά από την ενασχόληση με τα νευρωνικά δίκτυα και να στραφεί περισσότερο προς την ανάπτυξη συστημάτων τεχνητής νοημοσύνης, σε μια προσπάθεια να επιλυθούν πιο πολύπλοκα προβλήματα με άλλες μεθόδους.[16]

Μια σημαντική πρόοδος στον τομέα των τεχνητών νευρωνικών δικτύων καταγράφηκε το 1982, όταν ο φυσικός John Hopfield παρουσίασε ένα έργο που αναζωπύρωσε το επιστημονικό ενδιαφέρον για την περιοχή. Σε μια σύντομη αλλά ιδιαίτερα επιδραστική εργασία, απέδειξε πως ένα νευρωνικό δίκτυο μπορεί να λειτουργήσει όχι μόνο ως μηχανισμός επεξεργασίας πληροφορίας, αλλά και ως αποθηκευτικός χώρος μνήμης, με τη δυνατότητα να ανακατασκευάσει ένα πλήρες σύνολο δεδομένων ακόμη και από μερική πληροφορία. Η εργασία αυτή αποτέλεσε το έναυσμα για ένα νέο κύμα ερευνών και ιδεών, εδραιώνοντας την έννοια των αναδρομικών νευρωνικών δικτύων και των δυναμικών συστημάτων στον χώρο της τεχνητής νοημοσύνης.[17]

Το 1986 αποτέλεσε έτος-ορόσημο για την εξέλιξη των τεχνητών νευρωνικών δικτύων, καθώς οι McClelland και Rumelhart δημοσίευσαν την εργασία τους με τίτλο "Parallel Distributed Processing", εισάγοντας σημαντικές έννοιες που διαμόρφωσαν την πορεία της έρευνας. Κεντρική τους ιδέα ήταν πως ένα νευρωνικό δίκτυο μπορεί να λειτουργεί ως ένας παράλληλος επεξεργαστής, αξιοποιώντας την ταυτόχρονη επεξεργασία πληροφοριών από διαφορετικά επίπεδα νευρώνων. Μέσα από τη μελέτη τους καθιερώθηκε η δυνατότητα ύπαρξης πολλών ενδιάμεσων επιπέδων (hidden layers) σε ένα δίκτυο και θεμελιώθηκε η μέθοδος της οπισθοδιάδοσης σφάλματος (backpropagation), η οποία επιτρέπει την προσαρμογή των βαρών του δικτύου με στόχο την ελαχιστοποίηση του σφάλματος. Η τεχνική αυτή αναδείχθηκε σε μία από τις πιο καθοριστικές και ευρέως χρησιμοποιούμενες μεθόδους εκπαίδευσης έως σήμερα.[18]

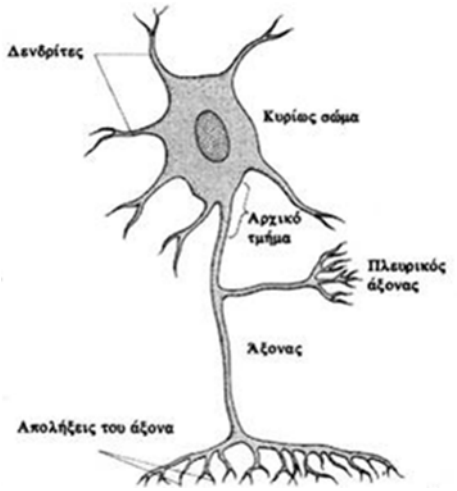
Μετά τις σημαντικές εξελίξεις της δεκαετίας του 1980, τα τεχνητά νευρωνικά δίκτυα απέκτησαν έναν αυτόνομο και δυναμικά αναπτυσσόμενο τομέα εντός της επιστημονικής κοινότητας. Άρχισαν να αντιμετωπίζονται όχι απλώς ως μία υποενότητα της τεχνητής νοημοσύνης, αλλά ως ένα ιδιαίτερο επιστημονικό πεδίο με δικά του χαρακτηριστικά,

θεωρίες και εφαρμογές. Πολλοί ερευνητές αφιέρωσαν συστηματικά τις προσπάθειές τους στην κατανόηση και βελτίωση αυτών των συστημάτων, γεγονός που οδήγησε σε ραγδαία πρόοδο. Ιδρύθηκαν επιστημονικά συνέδρια και περιοδικά αποκλειστικά αφιερωμένα στη μελέτη των νευρωνικών δικτύων, ενώ σε σταθερή βάση δημοσιεύονται νέες έρευνες, μελέτες και προσεγγίσεις, επιβεβαιώνοντας το ισχυρό και διαρκώς αυξανόμενο ενδιαφέρον του ακαδημαϊκού και τεχνολογικού κόσμου.

Συνοψίζοντας, η ανάπτυξη των τεχνητών νευρωνικών δικτύων προήλθε σε μεγάλο βαθμό από την ανθρώπινη επιθυμία να κατανοήσει τη λειτουργία του ίδιου του εγκεφάλου και, παράλληλα, από τη φιλοδοξία να δημιουργήσει μηχανές ικανές να προσομοιώσουν τη σύνθετη σκέψη και συμπεριφορά του ανθρώπου. Τα ιστορικά επιτεύγματα που προηγήθηκαν έθεσαν τις βάσεις γι' αυτήν την εξέλιξη, ωστόσο οι στόχοι αυτοί παραμένουν σε μεγάλο βαθμό ανεκπλήρωτοι. Η πλήρης κατανόηση του εγκεφάλου και η δημιουργία μιας μηχανής με παρόμοιες ικανότητες εξακολουθούν να αποτελούν προκλήσεις που η επιστήμη και η τεχνολογία προσπαθούν ακόμα να κατακτήσουν.

2.2 Νευρώνες του εγκεφάλου

Ο νευρώνας αποτελεί το θεμέλιο του νευρικού συστήματος, καθώς είναι το κύτταρο που ευθύνεται για τη δημιουργία και μεταφορά ηλεκτρικών σημάτων εντός του ίδιου ή μεταξύ άλλων νευρικών κυττάρων. Η μελέτη των εκατομμυρίων αυτών νευρικών μονάδων έχει απασχολήσει εντατικά την επιστημονική κοινότητα για περισσότερο από έναν αιώνα, καθώς η πολύπλοκη αλληλεπίδρασή τους παραμένει ένα αινιγματικό πεδίο έρευνας. Παρά τις σημαντικές προόδους που έχουν σημειωθεί στη νευροεπιστήμη, ο πλήρης μηχανισμός με τον οποίο ο εγκέφαλος λειτουργεί ως ενιαίο σύστημα δεν έχει ακόμη κατανοηθεί σε βάθος.[19]



Εικόνα 2.1 Νευρώνας εγκεφάλου 1

- Οι δενδρίτες διακλαδώνονται, προς τα έξω του κυρίως σώματος του κυττάρου, και λαμβάνουν πληροφορίες-ερεθύσματα από το περιβάλλον, τα οποία αποθηκεύουν και μεταβιβάζουν στο κυρίως σώμα του.
- Το κυρίως σώμα τώρα περιέχει τον πυρήνα του κυττάρου όπως και τον μηχανισμό παραγωγής πρωτεϊνών.
- Ο άξονας ή νευρική ίνα είναι η προέκταση του κυτταρικού σώματος όπου μεταφέρει τα ερεθύσματα μέσω ηλεκτρικών σημάτων από τον νευρώνα στις απολήξεις με μεγάλες ταχύτητες, σχεδόν ταυτόχρονα.
- Τα σήματα αυτά καταλήγουν στις νευρικές απολήξεις που περιέχουν συναπτικά κυστίδια τα οποία απελευθερώνουν μια χημική ουσία τον νευροδιαβιβαστή, σε

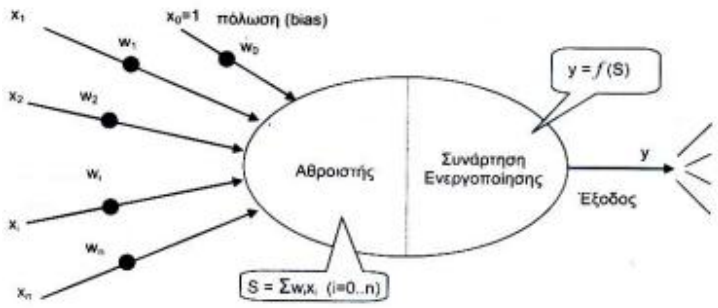
περίπτωση που το κύτταρο υποστεί διέγερση. Ο νευροδιαβιβαστής λειτουργεί ως αγγελιαφόρος προς τους νευρώνες που είναι ενωμένοι μαζί με τον εν λόγω νευρώνα και μεταφέρει το σήμα του προς αυτούς. Έτσι η μεταφορά των ερεθισμάτων από νευρώνα σε νευρώνα καθιστά δυνατή την ένωση τους και όλοι μαζί καθιστούν ένα νευρωνικό δίκτυο, τον εγκέφαλο[20].

Το δίκτυο των νευρώνων που επικοινωνούν μεταξύ τους μέσω συνάψεων αποτελεί τον θεμέλιο λίθο για την ανάπτυξη της μάθησης, της μνήμης και της συναισθηματικής εμπειρίας στον άνθρωπο. Όσο αυξάνεται η ποσότητα των εξωτερικών ερεθισμάτων που λαμβάνει ο εγκέφαλος, τόσο αυξάνεται και ο αριθμός των νέων συνάψεων που σχηματίζονται. Η διαδικασία αυτή συμβάλλει καθοριστικά στην ενίσχυση των γνωστικών ικανοτήτων, της νοημοσύνης και της συναισθηματικής ωριμότητας του ατόμου.

2.3 Τεχνητά νευρωνικά δίκτυα

Τα τεχνητά νευρωνικά δίκτυα δημιουργήθηκαν ως απάντηση στην αδυναμία των παραδοσιακών υπολογιστών να προσομοιώσουν τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου. Στόχος τους είναι να αξιοποιήσουν τη γνώση που έχουμε αποκτήσει από τη μελέτη των βιολογικών νευρώνων και να την ενσωματώσουν σε υπολογιστικά μοντέλα. Με αυτόν τον τρόπο επιχειρούν να μιμηθούν τη νοητική επεξεργασία και τη μαθησιακή ικανότητα του εγκεφάλου, φέρνοντας τους υπολογιστές ένα βήμα πιο κοντά στην ανθρώπινη σκέψη.

Με αυτόν τον τρόπο διαμορφώνεται ένα σύνολο κόμβων, οι οποίοι λειτουργούν ως τεχνητά κύτταρα, ενώ οι συνδέσεις μεταξύ τους προσομοιώνουν τις βιολογικές συνάψεις. Κατά τη διαδικασία της εκπαίδευσης του δικτύου, οι τιμές αυτών των συνδέσεων (βάρη) προσαρμόζονται δυναμικά, ώστε να καταλήξουν σε μια τελική μορφή που αποθηκεύει τη «γνώση» του συστήματος. Αυτή η ικανότητα προσαρμογής και αποθήκευσης επιτρέπει στο δίκτυο να ανταποκρίνεται σωστά σε μελλοντικές εισόδους, υιοθετώντας έναν τρόπο λειτουργίας που μιμείται σε μεγάλο βαθμό τις γνωσιακές λειτουργίες του ανθρώπινου εγκεφάλου.



Εικόνα 2.2 Τεχνητός Νευρώνας

Στο σχήμα παρατηρούμε πως οι είσοδοι του νευρωνικού δικτύου, συμβολισμένες ως x_1, x_2, ..., x_n, κατευθύνονται από τα αριστερά προς τα δεξιά προς έναν τεχνητό νευρώνα. Κάθε είσοδος συνοδεύεται από ένα αντίστοιχο βάρος w_i, το οποίο καθορίζει την επίδρασή της στον νευρώνα. Τα βάρη αυτά παίζουν καθοριστικό ρόλο, καθώς

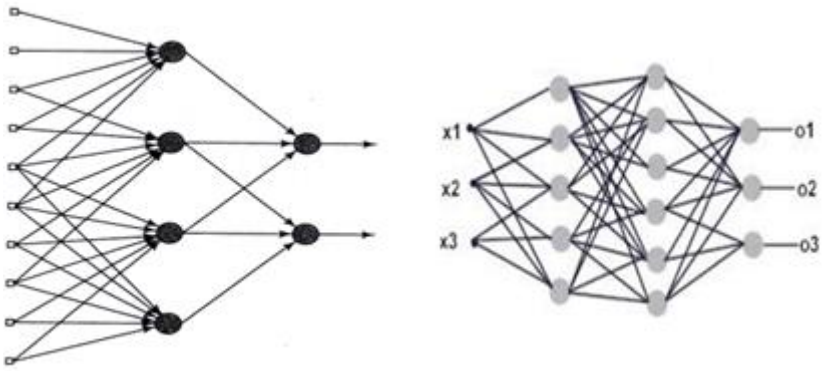
προσδιορίζουν τη συμβολή της κάθε εισόδου στην έξοδο του νευρώνα, δηλαδή πόσο επηρεάζει η κάθε μεταβλητή εισόδου την τελική ενεργοποίηση του νευρώνα.

Ο αθροιστής αποτελεί βασικό τμήμα της λειτουργίας ενός τεχνητού νευρώνα, καθώς συλλέγει όλες τις εισόδους του νευρώνα, τις πολλαπλασιάζει με τα αντίστοιχα βάρη τους και υπολογίζει το αθροιστικό τους αποτέλεσμα. Αυτό το συνολικό άθροισμα στη συνέχεια περνάει από μια συνάρτηση ενεργοποίησης, η οποία καθορίζει αν ο νευρώνας θα ενεργοποιηθεί ή όχι. Ανάλογα με την τιμή που επιστρέφει η συνάρτηση, ο νευρώνας χαρακτηρίζεται ως ενεργός ή ανενεργός για το συγκεκριμένο σετ εισόδων, επηρεάζοντας έτσι την περαιτέρω ροή πληροφορίας στο δίκτυο.[21]

2.4 Αρχιτεκτονικές Νευρωνικών Δικτύων

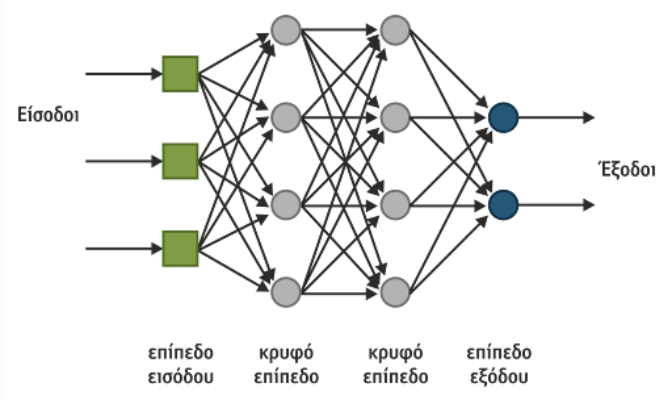
Η αρχιτεκτονική ενός νευρωνικού δικτύου καθορίζεται από τον τρόπο με τον οποίο συνδέονται μεταξύ τους οι νευρώνες, επηρεάζοντας άμεσα τόσο τη διαδικασία εκπαίδευσης όσο και την ποιότητα των αποτελεσμάτων. Η επιλογή της κατάλληλης αρχιτεκτονικής είναι κρίσιμο στάδιο στον σχεδιασμό ενός συστήματος, καθώς πρέπει να λαμβάνεται υπόψη το είδος του προβλήματος προς επίλυση. Ο σχεδιαστής του δικτύου οφείλει, πριν την υλοποίηση, να αξιολογήσει ποια διάταξη θα προσφέρει την πιο αποδοτική επίδοση μετά την εκπαίδευση. Επιπλέον, είναι σημαντικό να εξεταστούν και οι κατάλληλοι αλγόριθμοι εκπαίδευσης που υποστηρίζουν την εκάστοτε αρχιτεκτονική, καθώς κάθε μορφή δικτύου μπορεί να απαιτεί διαφορετικές τεχνικές μάθησης.[22]

Η πρώτη βασική κατηγορία νευρωνικών δικτύων είναι τα δίκτυα εμπρόσθιας τροφοδότησης που διαθέτουν ένα μόνο κρυφό επίπεδο. Σε αυτού του τύπου την αρχιτεκτονική, οι συνδέσεις μεταξύ των νευρώνων κινούνται αποκλειστικά προς τα εμπρός, δηλαδή από τους νευρώνες εισόδου προς τους νευρώνες του κρυφού επιπέδου και έπειτα στους νευρώνες εξόδου, χωρίς να δημιουργούνται κυκλικές διαδρομές ή επιστροφές προς προηγούμενα επίπεδα ή ακόμη και προς τον ίδιο νευρώνα. Οι είσοδοι επεξεργάζονται στο ενδιάμεσο αυτό επίπεδο, όπου λαμβάνουν χώρα οι κύριοι υπολογισμοί, και στη συνέχεια το αποτέλεσμα μεταφέρεται στην έξοδο του δικτύου.



Εικόνα 2.3 Αριστερά ένα μερικώς και δεξιά ένα πλήρως συνδεδεμένο δίκτυο

Η δεύτερη κατηγορία αρχιτεκτονικής περιλαμβάνει τα δίκτυα πρόσθιας τροφοδότησης με περισσότερα από ένα κρυφά επίπεδα. Η κύρια διαφορά σε σχέση με την προηγούμενη κατηγορία έγκειται στην προσθήκη επιπλέον ενδιάμεσων επιπέδων, γεγονός που ενισχύει τη δυνατότητα του δικτύου να επεξεργάζεται και να αναπαριστά πιο σύνθετες και μη γραμμικά διαχωρίσιμες σχέσεις στα δεδομένα. Καθώς αυξάνεται ο αριθμός των επιπέδων και των νευρώνων, το δίκτυο αποκτά μεγαλύτερη υπολογιστική ισχύ και ικανότητα γενίκευσης, όμως αυτή η πολυπλοκότητα συνοδεύεται από κινδύνους όπως ο υπερβολικός υπολογιστικός φόρτος και η πιθανότητα υπερπροσαρμογής, ειδικά όταν τα δεδομένα εκπαίδευσης είναι περιορισμένα.

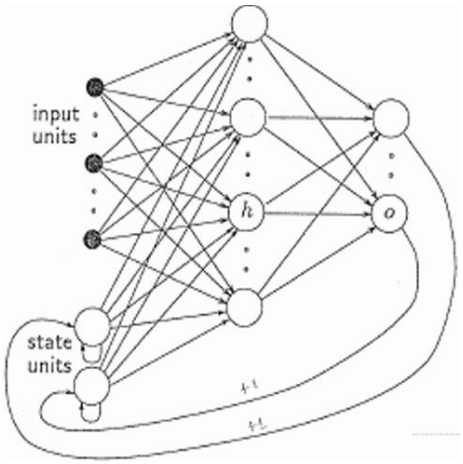


Εικόνα 2.4 Αρχιτεκτονική εμπρόσθιας τροφοδότησης

με δυο κρυφά επίπεδα νευρώνων

Η τρίτη και πιο σύνθετη μορφή αρχιτεκτονικής είναι τα επαναληπτικά ή αναδρομικά νευρωνικά δίκτυα. Σε αυτά, τουλάχιστον μία σύνδεση λειτουργεί αναδρομικά, δημιουργώντας βρόχους στο δίκτυο, μέσω των οποίων η έξοδος ενός νευρώνα επανεισάγεται ως είσοδος σε επόμενη χρονική στιγμή. Οι βρόχοι αυτοί επιτρέπουν στο δίκτυο να «θυμάται» παλαιότερες καταστάσεις, καθιστώντας το κατάλληλο για προβλήματα που περιλαμβάνουν χρονική εξάρτηση. Οι αναδρομικές συνδέσεις μπορούν να προέρχονται είτε από το κρυφό επίπεδο είτε από το επίπεδο εξόδου, ανάλογα με τη δομή του δικτύου. Παρότι η δυνατότητα ενσωμάτωσης μνήμης καθιστά αυτά τα δίκτυα εξαιρετικά ισχυρά για εφαρμογές όπως η επεξεργασία ακολουθιών, η εκπαίδευσή τους

είναι ιδιαίτερα απαιτητική και επηρεάζεται σημαντικά από τη σύνθετη ροή της πληροφορίας λόγω των αναδρομικών στοιχείων.[23]



Εικόνα 2.5 Δίκτυο αναδρομικής αρχιτεκτονικής.

2.5 MLP με Backpropagation

Εισαγωγή και Σκοπός του MLP με Backpropagation

Τα Πολυεπίπεδα Δίκτυα Αντίληψης (Multilayer Perceptron - MLP) αποτελούν την πιο διαδεδομένη μορφή τεχνητών νευρωνικών δικτύων, που χρησιμοποιείται ευρέως για την επίλυση πολύπλοκων προβλημάτων, ιδιαίτερα αυτών που δεν μπορούν να λυθούν μέσω γραμμικού διαχωρισμού. Το βασικό χαρακτηριστικό των δικτύων αυτών είναι η παρουσία πολλαπλών στρωμάτων νευρώνων (τουλάχιστον ένα κρυφό επίπεδο), τα οποία συνεργάζονται μεταξύ τους προκειμένου να αναγνωρίσουν σύνθετα πρότυπα και να πραγματοποιήσουν προβλέψεις.

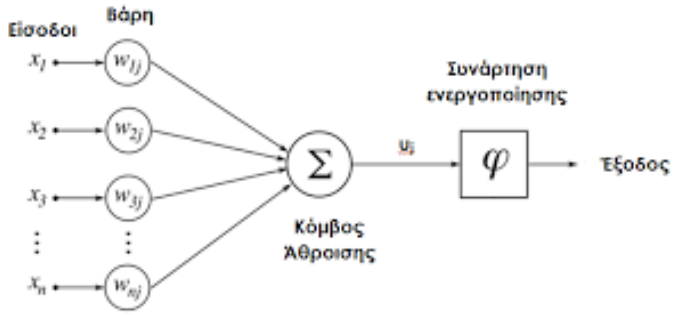
Η εκπαίδευση ενός MLP γίνεται κυρίως με τη μέθοδο της οπισθοδιάδοσης σφάλματος (Backpropagation), η οποία εισήχθη από τους Rumelhart, Hinton και Williams το 1986. Η μέθοδος αυτή επιτρέπει στα δίκτυα να «μάθουν» και να βελτιώνουν συνεχώς την απόδοσή τους μέσω της τροποποίησης των βαρών σύνδεσης μεταξύ των νευρώνων, βασιζόμενη στη μείωση του σφάλματος πρόβλεψης.[24]

Δομή ενός MLP

Ένα πολυεπίπεδο perceptron αποτελείται από τρεις τύπους επιπέδων:

- Επίπεδο εισόδου (Input layer): Δέχεται τις αρχικές πληροφορίες και δεδομένα, όπως στατιστικά χαρακτηριστικά ή μετρήσεις.
- Κρυφά επίπεδα (Hidden layers): Ενδιάμεσα στρώματα νευρώνων που επεξεργάζονται τα δεδομένα, μετατρέποντάς τα σε μία πιο περίπλοκη και εκλεπτυσμένη μορφή.
- Επίπεδο εξόδου (Output layer): Παράγει την τελική πρόβλεψη ή την ταξινόμηση των δεδομένων.

Κάθε νευρώνας σε αυτά τα επίπεδα συνδέεται πλήρως με όλους τους νευρώνες του επόμενου επιπέδου, δημιουργώντας μία πυκνή και ολοκληρωμένη αρχιτεκτονική. Οι συνδέσεις μεταξύ των νευρώνων διαθέτουν «βάρη» (weights), που αντιπροσωπεύουν τη σημασία της κάθε εισόδου στον προσδιορισμό της τελικής εξόδου.



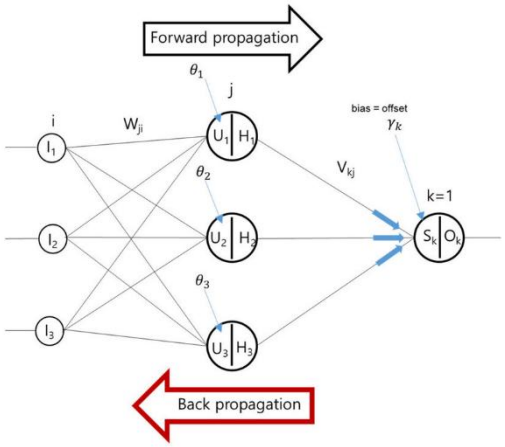
Εικόνα 2.6 Διάγραμμα αρχιτεκτονικής ενός MLP δικτύου

Αλγόριθμος Οπισθοδιάδοσης (Backpropagation Algorithm)

Η οπισθοδιάδοση αποτελεί έναν εποπτευόμενο αλγόριθμο εκπαίδευσης που λειτουργεί επαναληπτικά, χρησιμοποιώντας δύο βασικές φάσεις:

- Φάση εμπρόσθιας τροφοδότησης (Forward Propagation)
  Κατά τη διάρκεια αυτής της φάσης, τα δεδομένα εισόδου περνούν μέσα από το δίκτυο. Κάθε νευρώνας πολλαπλασιάζει τις εισόδους του με τα αντίστοιχα βάρη, αθροίζει τα αποτελέσματα και ενεργοποιεί μια συνάρτηση ενεργοποίησης (συνήθως sigmoid ή ReLU), παράγοντας μια έξοδο που μεταφέρεται στο επόμενο επίπεδο.
- Φάση οπισθοδιάδοσης (Backward Propagation)
  Εδώ υπολογίζεται το σφάλμα της τελικής εξόδου, συγκρίνοντάς την με την επιθυμητή τιμή (target). Το σφάλμα αυτό μεταδίδεται προς τα πίσω στο δίκτυο, από το επίπεδο εξόδου προς το επίπεδο εισόδου, ώστε να τροποποιηθούν τα βάρη και να μειωθεί το σφάλμα στην επόμενη επανάληψη.

Η διαδικασία αυτή επαναλαμβάνεται σε πολλαπλές «εποχές» (epochs) μέχρι να επιτευχθεί η επιθυμητή απόδοση.



Εικόνα 2.7 Διάγραμμα ροής της διαδικασίας Forward και Backward Propagation

Μαθηματική περιγραφή του Backpropagation

Το σφάλμα E που θέλουμε να ελαχιστοποιήσουμε ορίζεται συνήθως ως:

$$E = \frac{1}{2}\sum_p (t_p - o_p)^2$$

όπου:

- tp η επιθυμητή έξοδος,
- op η πραγματική έξοδος του δικτύου.

Η βασική ιδέα του αλγορίθμου οπισθοδιάδοσης (Backpropagation) είναι η ενημέρωση των βαρών των συνδέσεων με σκοπό την ελαχιστοποίηση του σφάλματος που δημιουργείται κατά την παραγωγή της εξόδου του δικτύου. Η ενημέρωση των βαρών ξεκινάει από το στρώμα εξόδου και προχωρά προς τα πίσω στα προηγούμενα στρώματα, ακολουθώντας τις εξής σχέσεις:

Η ενημέρωση των βαρών μεταξύ του νευρώνα i (στο προηγούμενο στρώμα) και του νευρώνα j (στο επόμενο στρώμα) γίνεται σύμφωνα με τη σχέση:

$$Wij\ (t + 1) = Wij(t) + \eta * \delta pj\ opj$$

όπου:

- Wij(t) : το τρέχον βάρος της σύνδεσης μεταξύ των νευρώνων.
- η: ο ρυθμός εκμάθησης (learning rate).
- δpj : ο όρος σφάλματος για τον νευρώνα j.
- opi : η έξοδος του νευρώνα i στο προηγούμενο στρώμα.

Η μορφή του όρου σφάλματος *δpj\delta_{pj}*δpj διαφέρει ανάλογα με το αν ο νευρώνας είναι στο στρώμα εξόδου ή σε κρυφό στρώμα:

Για νευρώνες του στρώματος εξόδου, ο όρος σφάλματος υπολογίζεται από τη σχέση:

$$\delta pj = \alpha * opj * (1-opj) * (tpj - opj)$$

όπου:

- opj : η έξοδος του νευρώνα *jj*j στο στρώμα εξόδου.
- tpj : η επιθυμητή έξοδος του νευρώνα.
- α: μια σταθερά που σχετίζεται με τη συνάρτηση ενεργοποίησης.

Για νευρώνες κρυφών στρωμάτων, ο όρος σφάλματος είναι:

$$\delta pj = \alpha * opj * (1-opj) \Sigma(\delta pk * wjk)$$

όπου το άθροισμα λαμβάνει χώρα πάνω από όλους τους νευρώνες k του επόμενου στρώματος, και αφορά την επίδραση του σφάλματος του κάθε τέτοιου νευρώνα.

Με αυτές τις μαθηματικές σχέσεις, ο αλγόριθμος backpropagation προσαρμόζει σταδιακά τα βάρη του δικτύου, ώστε να μειωθεί το σφάλμα πρόβλεψης σε κάθε επανάληψη, βελτιώνοντας συνεχώς την απόδοση και την ακρίβεια των προβλέψεων.

```
Repeat
{
    for (pno=0;pno<N_Patterns;pno++)
    {
        /* forward pass */
        Apply Input[pno] to input units;
        Compute Y[pno] at output units ;
        /* Backward pass */
        For each layer, starting at output
        {
            For each unit in this layer
            {
                Compute the error at this unit
                For each weight to this unit
                {
                        Compute Δw
                        Apply Δw
                }
            }
        }
    }
    Increment epoch counter
    Compute total error
}
until (total error small enough or
       epoch count exceeded)
```

Εικόνα 2.8 Backpropagation Algorithm

Συναρτήσεις Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης αποτελούν ένα θεμελιώδες στοιχείο στη δομή και την εκπαίδευση των τεχνητών νευρωνικών δικτύων, αφού καθορίζουν τον τρόπο με τον οποίο το δίκτυο αποφασίζει πώς θα ενεργοποιηθεί ή θα απενεργοποιηθεί ένας νευρώνας, καθώς και πώς διαδίδεται το σήμα από το ένα επίπεδο στο επόμενο. Συνεπώς, είναι κρίσιμες για την εκμάθηση σύνθετων μοτίβων και την επίλυση μη γραμμικών προβλημάτων.

Οι πιο κοινές συναρτήσεις ενεργοποίησης που χρησιμοποιούνται είναι οι εξής:

1. Συνάρτηση Sigmoid

Η Sigmoid συνάρτηση ενεργοποίησης περιγράφεται από τη σχέση:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Η Sigmoid λαμβάνει οποιονδήποτε πραγματικό αριθμό και τον μετατρέπει σε τιμή στο διάστημα (0, 1). Έχει το πλεονέκτημα ότι είναι διαφορίσιμη, γεγονός που διευκολύνει τη διαδικασία εκπαίδευσης του δικτύου μέσω του αλγορίθμου Backpropagation. Συχνά χρησιμοποιείται για προβλήματα ταξινόμησης ή για προβλήματα όπου επιθυμείται η έξοδος να είναι πιθανότητα. Ωστόσο, έχει το μειονέκτημα ότι μπορεί να προκαλέσει πρόβλημα κορεσμού (saturation), λόγω της μηδενικής κλίσης στις ακραίες περιοχές, με αποτέλεσμα να επιβραδύνεται η διαδικασία της εκπαίδευσης του δικτύου.

2. Συνάρτηση Tanh (Υπερβολική Εφαπτομένη)

Η Tanh περιγράφεται από τη σχέση:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Η Tanh λαμβάνει οποιοδήποτε πραγματικό αριθμό και τον μετατρέπει σε τιμή στο διάστημα (-1, 1). Όπως η Sigmoid, είναι επίσης διαφορίσιμη, αλλά σε αντίθεση με τη Sigmoid, η μέση τιμή της Tanh είναι το 0. Αυτό το χαρακτηριστικό της την καθιστά πιο κατάλληλη για χρήση σε προβλήματα όπου οι τιμές εισόδου του δικτύου είναι θετικές και αρνητικές. Παρουσιάζει παρόμοιο μειονέκτημα κορεσμού με τη Sigmoid, με αποτέλεσμα να δημιουργεί δυσκολίες κατά την εκπαίδευση.

3. Συνάρτηση ReLU (Rectified Linear Unit)

Η συνάρτηση ReLU δίνεται από τη σχέση:

$$f(x) = max(0, x)$$

Η συνάρτηση ReLU είναι μια από τις πιο δημοφιλείς συναρτήσεις ενεργοποίησης στα σύγχρονα δίκτυα. Η έξοδος της είναι μηδενική για αρνητικές τιμές και ίση με την τιμή εισόδου όταν αυτή είναι θετική. Η συνάρτηση ReLU είναι απλή, έχει πολύ καλή

υπολογιστική απόδοση και επιλύει σε μεγάλο βαθμό το πρόβλημα κορεσμού που παρατηρείται σε Sigmoid και Tanh. Ωστόσο, η ReLU μπορεί να αντιμετωπίσει το λεγόμενο πρόβλημα "dying ReLU", κατά το οποίο ορισμένοι νευρώνες σταματούν να μαθαίνουν, καθώς η έξοδος παραμένει σταθερά μηδενική, ειδικά εάν λαμβάνουν συχνά αρνητικές εισόδους.

Η επιλογή της κατάλληλης συνάρτησης ενεργοποίησης εξαρτάται άμεσα από το πρόβλημα που προσπαθεί να επιλύσει κανείς, τα δεδομένα εισόδου, την αρχιτεκτονική του δικτύου και την επιθυμητή συμπεριφορά στην έξοδο. Σε ορισμένες περιπτώσεις μπορεί να χρησιμοποιηθεί συνδυασμός συναρτήσεων ενεργοποίησης σε διαφορετικά στρώματα του ίδιου δικτύου, για να αξιοποιηθούν τα πλεονεκτήματα της κάθε συνάρτησης.

Χρήσεις και Εφαρμογές

Τα πολυεπίπεδα Perceptron (MLP) που εκπαιδεύονται μέσω του αλγορίθμου Backpropagation αποτελούν μια ιδιαίτερα δημοφιλή και αποτελεσματική προσέγγιση για την επίλυση μιας πληθώρας προβλημάτων σε διάφορους τομείς, λόγω της ικανότητάς τους να μοντελοποιούν πολύπλοκες μη γραμμικές σχέσεις. Οι σημαντικότερες χρήσεις και εφαρμογές τους είναι:

1. Κατηγοριοποίηση δεδομένων (Classification)

Η ταξινόμηση δεδομένων αποτελεί μία από τις βασικότερες εφαρμογές των MLP. Τα δίκτυα αυτά λαμβάνουν ως είσοδο δεδομένα (για παράδειγμα, χαρακτηριστικά αντικειμένων, ιατρικές μετρήσεις ή οικονομικούς δείκτες) και τα κατηγοριοποιούν σε διακριτές κλάσεις. Παραδείγματα εφαρμογών ταξινόμησης αποτελούν η αναγνώριση ανεπιθύμητων μηνυμάτων ηλεκτρονικού ταχυδρομείου (spam detection), η ιατρική

διάγνωση ασθενειών βάσει ιατρικών δεδομένων, και η ταξινόμηση πιστοληπτικής ικανότητας πελατών σε τραπεζικά συστήματα.

2. Πρόβλεψη χρονοσειρών (Time Series Prediction)

Τα MLP είναι ιδιαίτερα δημοφιλή στη μοντελοποίηση και πρόβλεψη χρονοσειρών, όπου ο στόχος είναι η πρόβλεψη μελλοντικών τιμών με βάση παρελθοντικά δεδομένα. Τέτοιες εφαρμογές βρίσκονται κυρίως σε πεδία όπως η οικονομία (πρόβλεψη μετοχών, συναλλαγματικών ισοτιμιών), οι ενεργειακές αγορές (πρόβλεψη ζήτησης ηλεκτρικής ενέργειας), οι μετεωρολογικές προβλέψεις και η πρόβλεψη πωλήσεων στον τομέα του εμπορίου. Τα MLP είναι χρήσιμα για την αναγνώριση σύνθετων μοτίβων και τάσεων σε δεδομένα χρονοσειρών, καθιστώντας δυνατές ακριβείς προβλέψεις. Στη φάση προεπεξεργασίας των χρονοσειρών, συχνά χρησιμοποιείται η τεχνική κινητού παραθύρου (sliding window): ένα παράθυρο σταθερού μεγέθους k μετακινείται κατά μία θέση κάθε φορά πάνω στη σειρά, και οι τιμές t–k,...,t–1t–k, ... , t–1t–k,...,t–1 μετατρέπονται σε διανύσματα εισόδου, ενώ η τιμή t γίνεται ο στόχος εξόδου. Με αυτό τον τρόπο το MLP “βλέπει” πάντα ένα σταθερό ιστορικό k τιμών και μαθαίνει να προβλέπει την επόμενη, αναγνωρίζοντας πολύπλοκα μοτίβα και τάσεις. Η προσέγγιση αυτή επιτρέπει αξιόπιστες προβλέψεις, ακόμα και όταν οι σχέσεις στο παρελθόν είναι μη γραμμικές.

3. Βελτιστοποίηση και Λήψη Αποφάσεων (Optimization and Decision Making)

Τα MLP μπορούν επίσης να χρησιμοποιηθούν σε συστήματα βελτιστοποίησης και λήψης αποφάσεων, όπου μαθαίνουν να εκτιμούν την αποτελεσματικότητα πιθανών αποφάσεων και στρατηγικών. Χρησιμοποιούνται ευρέως σε χρηματοοικονομικά συστήματα, στον σχεδιασμό εφοδιαστικών αλυσίδων (logistics), και σε συστήματα αυτόματης διαπραγμάτευσης (trading algorithms), όπου η λήψη αποφάσεων πρέπει να είναι γρήγορη, ακριβής και προσαρμοστική.

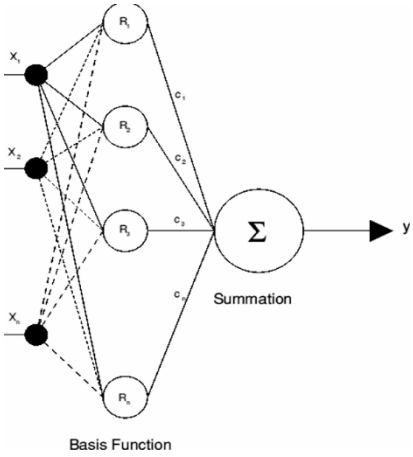
Συμπέρασμα

Συνοψίζοντας, το MLP με Backpropagation αποτελεί έναν ισχυρό, αποτελεσματικό και ευρέως διαδεδομένο αλγόριθμο, με πληθώρα εφαρμογών, που επιτρέπει τη μάθηση πολύπλοκων και μη γραμμικών σχέσεων σε δεδομένα. Η σωστή κατανόηση των επιμέρους διαδικασιών και η προσεκτική παραμετροποίηση των δικτύων αυτών αποτελούν το κλειδί για την επιτυχημένη εφαρμογή τους σε πληθώρα προβλημάτων.

2.6 Radial Basis Function Networks

Τα δίκτυα Radial Basis Function (RBF) ανήκουν στα μοντέλα επιβλεπόμενης μάθησης, καθώς στη φάση της εκπαίδευσης είναι γνωστά και χρησιμοποιούνται τα επιθυμητά αποτελέσματα. Η δομή τους αποτελείται από τρία επίπεδα: το επίπεδο εισόδου, το κρυφό επίπεδο και το επίπεδο εξόδου. Οι είσοδοι τροφοδοτούνται σε όλους τους νευρώνες του κρυφού επιπέδου, που ονομάζονται κέντρα, και εκεί υπολογίζεται η ευκλείδεια απόσταση ανάμεσα στο διάνυσμα εισόδου και το διάνυσμα του αντίστοιχου κέντρου. Έπειτα, η έξοδος του κάθε νευρώνα υπολογίζεται μέσω μιας Γκαουσιανής συνάρτησης και προωθείται στο επίπεδο εξόδου.[25]

Η αποτελεσματικότητα αυτής της διαδικασίας βασίζεται στο θεώρημα του Cover σχετικά με τη διαχωρισιμότητα προτύπων. Σύμφωνα με αυτό το θεώρημα, ένα περίπλοκο πρόβλημα κατηγοριοποίησης προτύπων γίνεται πιο πιθανό να διαχωριστεί γραμμικά όταν μετασχηματιστεί σε έναν χώρο μεγαλύτερης διαστατικότητας, αντί να εξεταστεί στον αρχικό χώρο μικρότερης διάστασης. Η μετάβαση σε έναν υψηλότερης διαστατικότητας χώρο και η μετατροπή ενός μη γραμμικά διαχωρίσιμου προβλήματος σε γραμμικά διαχωρίσιμο μπορεί να επιτευχθεί είτε με την εφαρμογή Γκαουσιανών συναρτήσεων στα δεδομένα, είτε με την αύξηση του αριθμού των νευρώνων στο κρυφό επίπεδο του δικτύου.



Εικόνα 2.9 Δομή RBF δικτύου

Τα δίκτυα RBF εκπαιδεύονται χρησιμοποιώντας δύο διαφορετικές προσεγγίσεις: με σταθερά και με μεταβλητά κέντρα.

Στην εκπαίδευση με σταθερά κέντρα, αρχικά καθορίζονται τα κέντρα, τα οποία δεν μεταβάλλονται κατά τη διάρκεια της εκπαίδευσης, ενώ προσαρμόζονται μόνο τα βάρη (coefficients). Με κάθε ζεύγος εισόδου και επιθυμητής εξόδου προκύπτει μία εξίσωση, και η επίλυση του συστήματος αυτών των εξισώσεων καθορίζει τα βάρη του δικτύου. Κρίσιμης σημασίας είναι η σωστή επιλογή των κέντρων, του πλάτους της Γκαουσιανής συνάρτησης (σ), καθώς επίσης και η αρχική τιμή των βαρών. Αυτή η μέθοδος χρησιμοποιείται κυρίως σε προβλήματα που απαιτούν ακρίβεια και ειδικά για παρεμβολή δεδομένων, όπου ο αριθμός των κέντρων ισούται με τον αριθμό των διαθέσιμων δεδομένων εισόδου.

Η εκπαίδευση με μεταβλητά κέντρα βασίζεται στη μέθοδο της καθόδου κλίσης, στην οποία προσαρμόζονται τόσο τα κέντρα, όσο και οι διασπορές και τα βάρη. Σε αυτή τη μέθοδο, αρχικά επιλέγονται τα κέντρα και οι τιμές των διασπορών και γίνεται αρχικοποίηση των βαρών με μικρές τυχαίες τιμές. Έπειτα, υπολογίζεται η πραγματική έξοδος και μέσω της καθόδου κλίσης προσαρμόζονται διαδοχικά κέντρα, διασπορές και βάρη. Η εκπαίδευση ολοκληρώνεται όταν επιτευχθεί ικανοποιητική σύγκλιση. Αυτή η προσέγγιση είναι πιο απλή συγκριτικά με τη μέθοδο Backpropagation, καθώς εδώ υπάρχει μόνο ένα επίπεδο προσαρμόσιμων βαρών. 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.

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.

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

1. Choose appropriate centres ***R***
2. Choose appropriate σ
3. Initialise the weights/coefficients ***c*** to small random values
4. Calculate the output $\underline{y} = \underline{A} \times \underline{c} = \Phi(x_n, \underline{R}) \times \underline{c}$
5. Find the optimum weight values with $\underline{c} = \underline{y} \times \underline{A}^{-1}$

Εικόνα 2.10 Αλγόριθμος RBF με σταθερά κέντρα

Στη μέθοδο μεταβλητών κέντρων χρησιμοποιούμε ένα στοχαστικό αλγόριθμο ταχυτέρας καθόδου. 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 :

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

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 :

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 :

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

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

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

Όπου :

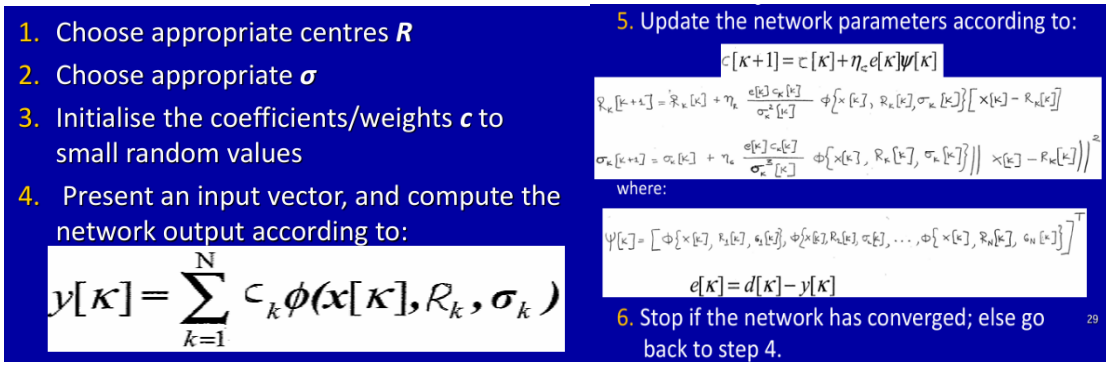
$c_k[k+1], R_k[k+1], \sigma_k[k+1]$ είναι τα νέα βάρη, κέντρα, διασπορές

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

n είναι ο ρυθμός μάθησης

$n_c \frac{\partial J[k]}{\partial c_k}, n_R \frac{\partial J[k]}{\partial R_k}, n_\sigma \frac{\partial J[k]}{\partial \sigma_k}$ 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.



Εικόνα 2.11 Αλγόριθμος RBF με μεταβλητά κέντρα

2.7 Hessian Free Optimization

Hessian Free Optimization (HFO), όπως εισήχθη από τον Martens το 2010 [26], αποτελεί μια τεχνική βελτιστοποίησης δεύτερης τάξης για συναρτήσεις στόχου με πραγματικές τιμές. Αποτελεί παραλλαγή της κλασικής μεθόδου του Newton και βασίζεται στη χρήση τοπικών τετραγωνικών προσεγγίσεων για την πρόταση νέων τιμών των παραμέτρων. Σε προβλήματα με μεγάλο αριθμό διαστάσεων – όπως σε βαθιά νευρωνικά δίκτυα με πολλά κρυφά επίπεδα – οι μέθοδοι πρώτης τάξης, όπως η Gradient Descent, συχνά είναι πολύ αργές και αναποτελεσματικές. Αυτό οφείλεται στο φαινόμενο του «vanishing gradient» (εξασθένιση της παραγώγου), όπου το μέγεθος του παραγώγου της συνάρτησης σφάλματος μειώνεται κάθε φορά που μεταδίδεται προς τα πίσω στα επίπεδα του δικτύου. Ως αποτέλεσμα, τα αρχικά επίπεδα λαμβάνουν σχεδόν μηδενική πληροφόρηση για την ενημέρωση των βαρών τους, καθιστώντας την εκπαίδευση πολύ αργή ή εντελώς ανεπαρκή.

Το πλεονέκτημα της χρήσης αλγορίθμων βελτιστοποίησης δεύτερης τάξης, όπως η μέθοδος του Newton ή η Hessian Free Optimization (HFO), έγκειται στο γεγονός ότι αυτοί λαμβάνουν υπόψη την καμπυλότητα της επιφάνειας σφάλματος (μέσω του πίνακα Hessian) κατά τη διαδικασία ενημέρωσης των παραμέτρων. Αυτό οδηγεί σε σημαντικά πιο αποτελεσματικές και στοχευμένες ενημερώσεις. Σε αντίθεση με τις μεθόδους πρώτης τάξης, που προσεγγίζουν την επιφάνεια σφάλματος ως επίπεδο και πραγματοποιούν σταδιακά βήματα προς το ελάχιστο, οι μέθοδοι δεύτερης τάξης σχηματίζουν μια πιο ακριβή παραβολική προσέγγιση γύρω από το τρέχον σημείο και εντοπίζουν απευθείας το τοπικό ελάχιστο αυτής της καμπύλης. Το αποτέλεσμα είναι πολύ αποδοτικότερη και ταχύτερη σύγκλιση. Επιπλέον, ο αλγόριθμος HFO έχει ακόμη ένα σημαντικό πλεονέκτημα έναντι των υπολοίπων αλγορίθμων πρώτης τάξης, αξιοποιώντας τη δεύτερη παράγωγο (Hessian), μπορεί να κατευθυνθεί απευθείας προς τα ακρότατα σημεία της συνάρτησης σφάλματος χωρίς να απαιτεί μικρά και συχνά αναποτελεσματικά βήματα, όπως συμβαίνει με τον απλό Gradient Descent. Έτσι, επιτυγχάνει ταχύτερη προσέγγιση της βέλτιστης λύσης, μειώνοντας σημαντικά τον συνολικό χρόνο εκπαίδευσης του δικτύου.

Ωστόσο, ο υπολογισμός του πίνακα Hessian για ένα μεγάλο τεχνητό νευρωνικό δίκτυο που περιλαμβάνει χιλιάδες ή και εκατομμύρια παραμέτρους είναι πρακτικά αδύνατος λόγω των τεράστιων απαιτήσεων σε μνήμη που απαιτούνται για την αποθήκευσή του. Γι' αυτό τον λόγο, παρόλο που έχουν αναπτυχθεί διάφορες παραλλαγές της μεθόδου Newton, όπως οι Newton-CG, CG-Steihaug, Newton-Lanczos [27] και Truncated Newton [28], καμία από αυτές δεν έχει εφαρμοστεί με επιτυχία στον τομέα της μηχανικής μάθησης ή στα νευρωνικά δίκτυα — ή τουλάχιστον οι εφαρμογές τους είναι εξαιρετικά περιορισμένες, όπως επισημαίνουν οι Martens και Sutskever [29].

Η μέθοδος Hessian Free (HFO) προσφέρει πρακτικές λύσεις στα προβλήματα μνήμης που δημιουργούνται από τη χρήση του πίνακα Hessian στην εκπαίδευση νευρωνικών δικτύων. Αντί να υπολογίζει και να αποθηκεύει ολόκληρο τον πίνακα Hessian (H), η HFO υπολογίζει μόνο το γινόμενο του Hessian με κάποιο αυθαίρετο διάνυσμα (Hu), χρησιμοποιώντας μαθηματικές τεχνικές όπως οι πεπερασμένες διαφορές (finite differences), οι οποίες έχουν παρόμοιο υπολογιστικό κόστος με τον υπολογισμό μιας απλής παραγώγου. Αυτή η προσέγγιση καθιστά τον αλγόριθμο ιδιαίτερα αποδοτικό, καθώς αποφεύγεται η άμεση χρήση του ίδιου του Hessian και αντ’ αυτού βασίζεται σε επαναλαμβανόμενα γινόμενα με διανύσματα.

Επιπρόσθετα, τα τοπικά τετραγωνικά μοντέλα σφάλματος που χρησιμοποιούνται από τις μεθόδους δεύτερης τάξης, μπορούν να ελαχιστοποιηθούν αποτελεσματικά με την τεχνική της συζυγούς καθοδικής πορείας (conjugate gradient - CG), η οποία λειτουργεί ως αντιστάθμισμα για την απουσία του πλήρους Hessian που απαιτείται στη μέθοδο Newton. Αν και θεωρητικά η CG απαιτεί N επαναλήψεις για σύγκλιση (όπου N είναι ο αριθμός παραμέτρων του δικτύου), στην πράξη χρησιμοποιούνται διάφορα κριτήρια τερματισμού που σταματούν την επανάληψη μόλις σημειωθεί ικανοποιητική πρόοδος. Αυτό είναι ιδιαίτερα κρίσιμο, καθώς δεν είναι πρακτικό να περιμένει κανείς την πλήρη σύγκλιση σε κάθε βήμα, ειδικά όταν η επιπλέον μείωση στο σφάλμα είναι ελάχιστη.

Αξίζει να σημειωθεί ότι, παρόλο που στη μέθοδο Hessian Free Optimization (HFO) δεν υπολογίζεται άμεσα ολόκληρος ο πίνακας Hessian, αυτό δεν σημαίνει ότι γίνεται κάποια αριθμητική προσέγγιση στον υπολογισμό — το γινόμενο του Hessian με ένα αυθαίρετο διάνυσμα (Hu) υπολογίζεται με ακρίβεια. Η βασική διαφορά μεταξύ της HFO και της

παραδοσιακής μεθόδου του Newton είναι ότι η τελευταία εκτελεί πλήρη βελτιστοποίηση πάνω στην τετραγωνική προσέγγιση της συνάρτησης σφάλματος, ενώ η HFO χρησιμοποιεί μια μη πλήρως συγκλίνουσα διαδικασία συζυγούς καθοδικής πορείας (CG). Ωστόσο, τα πλεονεκτήματα που προκύπτουν από την αποφυγή του πλήρους υπολογισμού και της αντιστροφής του πίνακα Hessian είναι σημαντικά, και αντισταθμίζουν με το παραπάνω την ελάχιστη απώλεια ακρίβειας που μπορεί να προκύψει από την μη πλήρη σύγκλιση της CG.

Παρόλο που το γινόμενο του Hessian με ένα διάνυσμα (Hu) μπορεί να υπολογιστεί με ακρίβεια και αποδοτικά, στη Hessian Free Optimization δεν χρησιμοποιείται κατά κύριο λόγο. Αντί αυτού, προτιμάται το γινόμενο Gu, όπου G είναι ο πίνακας Gauss-Newton, ο οποίος αποτελεί μια προσέγγιση του Hessian. Αν και θα μπορούσε κανείς να θεωρήσει περιττή τη χρήση μιας προσέγγισης αντί του πραγματικού πίνακα καμπυλότητας, ειδικά όταν δεν τίθεται θέμα απόδοσης, ο Gauss-Newton αποφεύγει συγκεκριμένα αριθμητικά προβλήματα που ενδέχεται να προκύψουν με τον Hessian και να οδηγήσουν την εκπαίδευση σε αποτυχία. Επιπλέον, ακόμη και όταν τέτοια προβλήματα δεν υφίστανται, η χρήση του G παρέχει πιο αποδοτικές κατευθύνσεις ελαχιστοποίησης, ενώ καταναλώνει λιγότερη μνήμη και επιταχύνει σημαντικά τη διαδικασία σε σχέση με τη χρήση του Hessian.

Η μέθοδος Hessian Free Optimization (HFO) αποτελεί έναν αλγόριθμο ελαχιστοποίησης για συναρτήσεις στόχου που είναι δύο φορές παραγωγίσιμες, δηλαδή διαθέτουν δεύτερη παράγωγο. Εστιάζει στη βελτιστοποίηση ενός διανύσματος παραμέτρων $w \in R$, χρησιμοποιώντας μια προσέγγιση παρόμοια με αυτή της μεθόδου Newton. Αντί να υπολογίζει άμεσα το ελάχιστο της συνάρτησης, η HFO εφαρμόζει επαναληπτικά τοπικές παραβολικές προσεγγίσεις, προκειμένου να υπολογίζει διορθώσεις στις παραμέτρους.

Σε κάθε επανάληψη, ξεκινώντας από τις προηγούμενες τιμές των παραμέτρων W_{t-1}, υπολογίζεται μια νέα προσέγγιση Wt μέσω ελαχιστοποίησης μιας παραβολικής συνάρτησης Mt−1(δ), η οποία προσεγγίζει τοπικά τη συμπεριφορά της συνάρτησης στόχου γύρω από το σημείο Wt-1.

Με τον τρόπο αυτό, η HFO καθορίζει μια κατεύθυνση και ένταση ενημέρωσης για τις παραμέτρους, με στόχο τη γρήγορη και αποδοτική σύγκλιση προς το ελάχιστο της συνάρτησης.

Η επίλυση του συστήματος της εξίσωσης 2.1 για την εύρεση του ελαχίστου δ* όπως προτείνεται από τη βασική μέθοδο του Newton είναι υπολογιστικά μη πρακτική, και σε ορισμένα δίκτυα ακόμη και αδύνατη, αν λάβουμε υπόψη την πολυπλοκότητα που απαιτείται της τάξης $O(n^3)$ [29]. Για να αποφευχθεί αυτό, χρησιμοποιείται η γραμμική μέθοδος Συζυγών Κατευθύνσεων Conjugate Gradient, η οποία προσεγγίζει μερικώς τη βελτιστοποίηση της παραβολικής προσέγγισης Μ. Ο προκύπτων προσεγγιστικός ελαχιστοποιητής δ* χρησιμοποιείται στη συνέχεια για την ενημέρωση των βαρών w.

$$f(w_{t-1} + \delta) \cong M_{t-t}(\delta) = f(w_{t-1}) + \nabla f(w_{t-1})^T \delta + \frac{1}{2}\delta^T B_{t-1}\delta$$

Εξίσωση 2.1: Η τοπική παραβολική προσέγγιση της αντικειμενικής συνάρτησης f(wt-1+δ), όπου Bt-1 είναι ο πίνακας καμπυλότητας, ο οποίος συνήθως είναι ο πίνακας Hessian (H)

Η διαδικασία ελαχιστοποίησης της τοπικής παραβολικής προσέγγισης οδηγεί στον εντοπισμό της καλύτερης δυνατής κατεύθυνσης δ* , η οποία χρησιμοποιείται για να υπολογιστεί η νέα τιμή των παραμέτρων, όπως περιγράφεται στην εξίσωση 2.2.

$$w_t = w_{t-1} + a\delta_t^*$$

Εξίσωση 2.2: Η ανανέωση των βαρών βασίζεται στη βέλτιστη τιμή δ*t , η οποία αντιστοιχεί στην κατεύθυνση που ελαχιστοποιεί το παραβολικό μοντέλο της εξίσωσης 2.1. Το τελικό βήμα καθορίζεται από έναν συντελεστή α, που βρίσκεται εντός του διαστήματος [0, 1] και επιλέγεται μέσω τεχνικής αναζήτησης βήματος(Line search)

Line Search

Η τεχνική Line Search αποτελεί έναν από τους βασικότερους μηχανισμούς στις μεθόδους βελτιστοποίησης και χρησιμοποιείται για να καθορίσει το κατάλληλο μέγεθος του βήματος (step size) κατά την ενημέρωση των παραμέτρων ενός δικτύου. Συγκεκριμένα, αφού προσδιοριστεί η κατεύθυνση βελτιστοποίησης (search direction) — συνήθως μέσω υπολογισμών που περιλαμβάνουν το gradient ή σε πιο εξελιγμένες μεθόδους, το

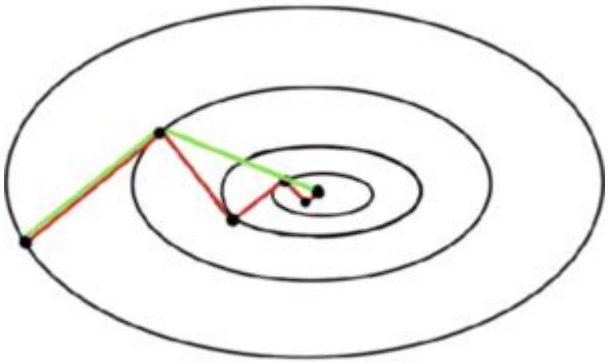
Hessian — το Line Search προσπαθεί να εντοπίσει την τιμή του βήματος $\alpha \in [0,1]$ που θα επιφέρει τη μεγαλύτερη δυνατή μείωση στη συνάρτηση κόστους. Μια απλοϊκή προσέγγιση για τον καθορισμό του μεγέθους του βήματος είναι να κινούμαστε κατά μήκος της κατεύθυνσης αναζήτησης με μικρά διαδοχικά βήματα, αξιολογώντας τη συνάρτηση σφάλματος σε κάθε σημείο, έως ότου παρατηρηθεί αύξηση του σφάλματος. Το Line Search εξασφαλίζει ότι η εκπαίδευση προχωρά σταθερά προς το τοπικό ελάχιστο της συνάρτησης κόστους χωρίς να κάνει υπερβολικά μεγάλα βήματα που ενδέχεται να προκαλέσουν αστάθεια ή πολύ μικρά βήματα που καθυστερούν τη σύγκλιση. Με αυτόν τον τρόπο, επιτυγχάνεται μια ισορροπία μεταξύ ταχύτητας και αξιοπιστίας κατά την εκπαίδευση του μοντέλου.[30]

Η εξίσωση 2.2 παρουσιάζει τα βασικά στοιχεία για τον υπολογισμό της επόμενης επανάληψης της τιμής x , όπου το δ είναι η κατεύθυνση αναζήτησης (search direction) και το α είναι το μέγεθος βήματος (step size), που καθορίζει πόσο μακριά θα κινηθεί το *xxx* προς αυτή την κατεύθυνση. Στην απλή μέθοδο της καθοδικής κλίσης (gradient descent), η κατεύθυνση αναζήτησης είναι το αρνητικό της παραγώγου της συνάρτησης σφάλματος και το μέγεθος βήματος είναι μια προκαθορισμένη τιμή (learning rate). Αν αυτό το βήμα είναι υπερβολικά μεγάλο, υπάρχει κίνδυνος η συνάρτηση κόστους να απομακρυνθεί πολύ από το ελάχιστό της. Αντίθετα, αν το βήμα είναι πολύ μικρό, τότε οι αλλαγές γίνονται ανεπαίσθητες, γεγονός που είτε καθυστερεί υπερβολικά τη διαδικασία εκπαίδευσης είτε "παγιδεύει" τη συνάρτηση σε κάποιο τοπικό ελάχιστο. Επομένως, είναι ζωτικής σημασίας να καθορίζεται ένα βέλτιστο μέγεθος βήματος σε κάθε επανάληψη και προς κάθε κατεύθυνση. Γι’ αυτόν τον λόγο, η τεχνική του line search στοχεύει στην εύρεση του ιδανικού μεγέθους βήματος που θα ελαχιστοποιεί τη συνάρτηση στόχου προς την τρέχουσα κατεύθυνση βελτιστοποίησης.

Conjugate Gradient

Για να εφαρμοστεί η μέθοδος line search με σκοπό τη βελτιστοποίηση του μεγέθους του βήματος και την ελαχιστοποίηση της συνάρτησης σφάλματος σε κάθε επανάληψη, είναι απαραίτητο πρώτα να καθοριστεί μια κατεύθυνση καθοδικής αναζήτησης (descent direction). Στην περίπτωση του Gradient Descent, η κατεύθυνση αυτή αντιστοιχεί στο

αρνητικό του παραγώγου της συνάρτησης σφάλματος στη συγκεκριμένη θέση. Ωστόσο, αυτή η επιλογή γενικά δεν θεωρείται ιδανική. Διαδοχικές κατευθύνσεις βασισμένες στο απλό παραγώγιμο, μπορεί να οδηγήσουν στο φαινόμενο που απεικονίζεται στην εικόνα (κόκκινη καμπύλη), κατά το οποίο τα βάρη ταλαντώνονται γύρω από την επιθυμητή λύση χωρίς να πλησιάζουν ουσιαστικά το ελάχιστο, καθιστώντας τη σύγκλιση αργή ή και αναποτελεσματική. [31]



Εικόνα 2.12 Gradient Descent (Κόκκινο) και Conjugate Gradient (Πράσινο)

Η μέθοδος Συζυγών Κατευθύνσεων (Conjugate Gradient - CG), στη θεωρία, μπορεί να φτάσει σε πλήρη σύγκλιση σε N βήματα, όπου N είναι ο αριθμός των παραμέτρων του δικτύου. Παρ' όλα αυτά, στην πράξη, η μέθοδος συχνά καταλήγει σε λύση πολύ πιο γρήγορα, ειδικά όταν η δομή του πίνακα καμπυλότητας B είναι κατάλληλη. Ακόμη και όταν δεν φτάσει σε πλήρη σύγκλιση, συνήθως προσφέρει σημαντική πρόοδο προς την κατεύθυνση της ελαχιστοποίησης. Μια τεχνική που χρησιμοποιείται για να ενισχύσει αυτή τη διαδικασία είναι η προπροσαρμογή (preconditioning), κατά την οποία εφαρμόζεται ένας πίνακας P ώστε να μετασχηματιστεί το σύστημα συντεταγμένων και να επιταχυνθεί η απόδοση του CG. Ο τρόπος εφαρμογής του CG με αυτή την τεχνική παρουσιάζεται στον αλγόριθμο της εικόνας 2.13.

Για να επιτευχθεί ο κατάλληλος τερματισμός του αλγορίθμου Συζυγών Κατευθύνσεων (CG), εφαρμόζονται ορισμένα κριτήρια τα οποία επιδιώκουν να βρουν μια ισορροπία ανάμεσα στην ακρίβεια της λύσης και στον αριθμό των επαναλήψεων που απαιτούνται

για την επίτευξή της. Ο Martens εισήγαγε μια μέθοδο αξιολόγησης της προόδου στη βελτιστοποίηση της παραμετρικής συνάρτησης, βασισμένη στη σχετική βελτίωση της τιμής της, όπως περιγράφεται στην εξίσωση 2.3. Οταν αυτη η τιμή s γίνει μικρότερη απο μια σταθερή μεταβλητή ο CG τερματίζεται.

$$s_j = \frac{M(x_j) - M(x_{j-k})}{M(x_j)}$$

Εξίσωση 2.3: Η μέτρηση της προόδου που προτάθηκε από τον Martens, όπου Xj είναι η j-οστή επανάληψη του αλγορίθμου Συζυγών Κατευθύνσεων (CG) και k είναι το μέγεθος του παραθύρου στο οποίο υπολογίζεται η πρόοδος.

inputs: b, A, x_0, P
$r_0 \leftarrow Ax_0 - b$
$y_0 \leftarrow$ solution of $Py = r_0$
$p_0 \leftarrow -y_0$
$i \leftarrow 0$
while termination conditions do not apply **do**
  $\alpha_i \leftarrow \frac{r_i^\top y_i}{p_i^\top A p_i}$
  $x_{i+1} \leftarrow x_i + \alpha_i p_i$
  $r_{i+1} \leftarrow r_i + \alpha_i A p_i$
  $y_{i+1} \leftarrow$ solution of $Py = r_{i+1}$
  $\beta_{i+1} \leftarrow \frac{r_{i+1}^\top y_{i+1}}{r_i^\top y_i}$
  $p_{i+1} \leftarrow -y_{i+1} + \beta_{i+1} p_i$
  $i \leftarrow i + 1$
end while
output: x_i

Εικόνα 2.13: Ο προρρυθμισμένος αλγόριθμος Συζυγών Κατευθύνσεων (Preconditioned Conjugate Gradient – CG). Σημειώνουμε ότι για την ελαχιστοποίηση της παραβολικής συνάρτησης στην HFO, ισχύει ότι x = δ, A = Bt−1 και b = ∇f(wt−1), P το preconditioning matrix

Damping

Ο αλγόριθμος συζυγών βαθμίδων (CG) απαιτεί ο πίνακας καμπυλότητας B να είναι θετικά ορισμένος για να λειτουργήσει σωστά. Ωστόσο, στα νευρωνικά δίκτυα η συνάρτηση στόχος είναι συνήθως μη κυρτή, κάτι που σημαίνει ότι ο B μπορεί να μην πληροί αυτή την προϋπόθεση. Σε μια τέτοια περίπτωση, η παραβολική προσέγγιση M ενδέχεται να μην έχει πραγματικό ελάχιστο και η μέθοδος CG να μην μπορεί να εφαρμοστεί. Επιπρόσθετα, στις πρώτες φάσεις της εκπαίδευσης, η προτεινόμενη διόρθωση $\delta*$ μπορεί να είναι πολύ μεγάλη και υπερβολικά «επιθετική», φτάνοντας έξω από το εύρος στο οποίο η παραβολική προσέγγιση παραμένει ακριβής. Αυτές οι δυσκολίες είναι χαρακτηριστικές των μεθόδων δεύτερης τάξης και συνήθως αντιμετωπίζονται με την τεχνική της απόσβεσης (damping), η οποία σταθεροποιεί τη διαδικασία βελτιστοποίησης. Ουσιαστικά, περιορίζουν τη βελτιστοποίηση της M σε μια «περιοχή εμπιστοσύνης» (trust region) προσθέτοντας στη M όρους ποινής. Υπάρχουν αρκετές μέθοδοι απόσβεσης (damping) που έχουν προταθεί από τον Martens και μπορούν να εφαρμοστούν στη Hessian Free Optimization (HFO).[26] Ωστόσο, σε αυτή τη διπλωματική εργασία χρησιμοποιείται η τεχνική απόσβεσης Tikhonov(εξίσωση 2.4) σε συνδυασμό με τον ευρετικό κανόνα Levenberg-Marquardt(εξίσωση 2.5).[32]

$$\hat{M}(\delta) \equiv M(\delta) + \frac{\lambda}{2}\delta^\top\delta = f(\theta) + \nabla f(\theta)^\top\delta + \frac{1}{2}\delta^\top\hat{\mathrm{B}}\delta$$

Εξίσωση 2.4: Η νέα αποσβεσμένη παραβολική προσέγγιση ορίζεται ως $\hat{B} = B + \lambda I$, όπου το $\hat{B}$ είναι ο τροποποιημένος πίνακας καμπυλότητας, το B είναι ο αρχικός πίνακας καμπυλότητας, το I είναι ο μοναδιαίος πίνακας, και το λ είναι μια μη αρνητική σταθερά ($\lambda \geq 0$) η οποία καθορίζει τη «δύναμη» της απόσβεσης.

Η επιλογή μιας καλής τιμής για το λ είναι καθοριστική για την επιτυχία της απόσβεσης Tikhonov. Η προσαρμογή της τιμής του λ κατά τη διάρκεια της διαδικασίας βελτιστοποίησης είναι ιδιαίτερα σημαντική, ώστε να μπορεί να ανταποκρίνεται συνεχώς στις μεταβαλλόμενες τοπικές καμπυλότητες της συνάρτησης στόχου f. Μια αποτελεσματική μέθοδος για την αντιμετώπιση αυτού του ζητήματος είναι ο ευριστικός αλγόριθμος Levenberg-Marquardt. Όταν η τιμή του ρ είναι πολύ μικρότερη από το 1,

αυτό σημαίνει πως το τετραγωνικό μοντέλο υπερεκτιμά τη μείωση της συνάρτησης στόχου, άρα η τιμή του λ πρέπει να αυξηθεί για πιο συντηρητικές ενημερώσεις. Αντίθετα, όταν το ρ πλησιάζει το 1, η προσέγγιση είναι αξιόπιστη και το λ μπορεί να μειωθεί για πιο τολμηρές ενημερώσεις.

Η ευρετική Levenberg-Marquardt προτείνει:

1. Αν ρ > 0.75 τότε λ γίνεται 2/3λ
2. Αν ρ < 0.25 τότε λ γίνεται 3/2λ
   Αλλιώς, το λ παραμένει ίδιο.

$$\rho \equiv \frac{f(\theta_{k-1} + \delta_k) - f(\theta_{k-1})}{M_{k-1}(\delta_k)}$$

Εξίσωση 2.5: Ο λόγος μείωσης μετρά τη μείωση της συνάρτησης στόχου σε σύγκριση με την τετραγωνική προσέγγιση.

Hv και Gv multiplications

Αντί να υπολογίζεται ολόκληρος ο πίνακας Hessian, γίνεται χρήση των εσωτερικών γινομένων του με τυχαία διανύσματα v∈R, κάτι που έχει υπολογιστικό κόστος αντίστοιχο με έναν υπολογισμό παραγώγου. Αν θεωρήσουμε τον πίνακα Hessian ως τον Ιακωβιανό πίνακα (δηλαδή τον πίνακα των πρώτων παραγώγων) της παραγώγου της αντικειμενικής συνάρτησης, τότε το γινόμενο H(w)v αντιπροσωπεύει τη κατευθυνόμενη παράγωγο του διανύσματος του gradient ∇f(w) προς την κατεύθυνση του διανύσματος v.

$$H(w)v = \lim_{\varepsilon \to 0} \frac{\nabla f(w + \varepsilon v) - \nabla f(w)}{\varepsilon}$$

Εξίσωση 2.6:Το H(w)v αντιπροσωπεύει το πόσο αλλάζει το gradient όταν κινούμαστε προς την κατεύθυνση του v.

Ένα σημαντικό πρόβλημα που σχετίζεται με τη χρήση του πίνακα Hessian ως πίνακα καμπυλότητας είναι ότι δεν είναι πάντα θετικά ορισμένος, κάτι που καθιστά αδύνατη την εφαρμογή της μεθόδου συζυγών βαθμίδων (CG) στο τετραγωνικό μοντέλο. Αν και οι

τεχνικές απόσβεσης (damping) προσπαθούν να αντιμετωπίσουν εν μέρει το πρόβλημα αυτό, υπάρχει μια πιο άμεση και αποτελεσματική λύση. Αντί να χρησιμοποιείται ο Hessian, μπορεί να χρησιμοποιηθεί ένας άλλος πίνακας – ο γενικευμένος πίνακας Gauss-Newton – ο οποίος αποτελεί προσέγγιση του Hessian και έχει το πλεονέκτημα ότι είναι πάντοτε θετικά ημι-ορισμένος. Πέρα από αυτό, στην πράξη έχει αποδειχθεί ότι ο πίνακας Gauss-Newton παρουσιάζει καλύτερη αποδοτικότητα και συνολική απόδοση σε σχέση με τον Hessian . Η εικόνα 2.13 δίχνει τον αλγόριθμο για υπολογισμο του Hv και η εικόνα 2.14 δίχνει τον αλγόριθμο για υπολογισμό του Gv. [33]

Forward Differentiation

Αν και η παραπάνω περιγραφή θα μπορούσε να παραπέμπει σε μια μέθοδο υπολογισμού του γινομένου Hv μέσω πεπερασμένων διαφορών (finite-differences), με κόστος αντίστοιχο ενός μόνο υπολογισμού του gradient, στην πράξη αυτή η προσέγγιση παρουσιάζει αριθμητικά σφάλματα, τα οποία είναι ιδιαίτερα ανεπιθύμητα κατά την εκπαίδευση νευρωνικών δικτύων. Γι' αυτόν τον λόγο εφαρμόζεται μια μέθοδος η οποία αποφεύγει αυτά τα αριθμητικά προβλήματα. Η μέθοδος αυτή ονομάζεται Forward Differentiation. Η βασική ιδέα του Forward Differentiation είναι η επαναλαμβανόμενη εφαρμογή του κανόνα αλυσίδας (chain rule) σε κάθε κόμβο του δικτύου, με τρόπο παρόμοιο με αυτόν που περιγράφεται στον αλγόριθμο backpropagation της ενότητας 2.5. Πιο συγκεκριμένα, ορίζεται ένας τελεστής Rv(X), ο οποίος δηλώνει την κατευθυνόμενη παράγωγο της μεταβλητής X προς την κατεύθυνση του διανύσματος v.

$$\mathrm{R}_v X = \lim_{\varepsilon \to 0} \frac{X(\mathrm{w} + \varepsilon v) - X(\theta)}{\varepsilon} = \frac{\partial X}{\partial \mathrm{w}} v$$

$$\begin{aligned} \mathrm{R}_v(X+Y) &= \mathrm{R}_v X + \mathrm{R}_v Y && \text{linearity} \\ \mathrm{R}_v(XY) &= (\mathrm{R}_v X)Y + X\mathrm{R}_v Y && \text{product rule} \\ \mathrm{R}_v(h(X)) &= (\mathrm{R}_v X)h'(X) && \text{chain rule} \end{aligned}$$

Εξίσωση 2.7: Rv(X), ο οποίος δηλώνει την κατευθυνόμενη παράγωγο της μεταβλητής X προς την κατεύθυνση του διανύσματος v.

input: v mapped to $(\mathrm{R}W_1, \ldots, \mathrm{R}W_{\ell-1}, \mathrm{R}b_1, \ldots, \mathrm{R}b_{\ell-1})$
$\mathrm{R}y_0 \leftarrow 0$ (since y_0 is not a function of the parameters)
for all i **from** 0 **to** $\ell - 1$ **do**
  $\mathrm{R}x_{i+1} \leftarrow \mathrm{R}W_i y_i + W_i \mathrm{R}y_i + \mathrm{R}b_i$ (product rule)
  $\mathrm{R}y_{i+1} \leftarrow \mathrm{R}x_{i+1} s'_{i+1}(x_{i+1})$ (chain rule)
end for
$\mathrm{R}dy_\ell \leftarrow \mathrm{R}\left(\frac{\partial L(y_\ell; t_\ell)}{\partial y_\ell}\right) = \frac{\partial\{\partial L(y_\ell; t_\ell)/\partial y_\ell\}}{\partial y_\ell}\mathrm{R}y_\ell = \frac{\partial^2 L(y_\ell; t_\ell)}{\partial y_\ell^2}\mathrm{R}y_\ell$
for all i **from** $\ell - 1$ **downto** 0 **do**
  $\mathrm{R}dx_{i+1} \leftarrow \mathrm{R}dy_{i+1} s'_{i+1}(x_{i+1}) + dy_{i+1} R\left\{s'_{i+1}(x_{i+1})\right\}$ (product rule)
  $= dy_{i+1} s''_{i+1}(x_{i+1}) \mathrm{R}x_{i+1}$ (chain rule)
  $\mathrm{R}dW_i \leftarrow \mathrm{R}dx_{i+1} y_i^\top + dx_{i+1} \mathrm{R}y_i^\top$ (product rule)
  $\mathrm{R}db_i \leftarrow \mathrm{R}dy_i$
  $\mathrm{R}dy_i \leftarrow \mathrm{R}W_i^\top dx_{i+1} + W_i^\top \mathrm{R}dx_{i+1}$ (product rule)
end for
output: $H(\mathrm{w})v$ as mapped from $(\mathrm{R}dW_1, \ldots, \mathrm{R}dW_{\ell-1}, \mathrm{R}db_1, \ldots, Rdb_{\ell-1})$.

Εικόνα 2.14: Αλγόριθμος για τον υπολογισμό του Hv product όπου L ειναι μία συνάρτηση σφάλματος

input: $\mathrm{R}W_1, \ldots, \mathrm{R}W_{\ell-1}, \mathrm{R}b_1, \ldots, \mathrm{R}b_{\ell-1}$.
$\mathrm{R}y_0 \leftarrow 0$ (y_0 is not a function of the parameters)
for all i **from** 1 **to** $\ell - 1$ **do**
  $\mathrm{R}x_{i+1} \leftarrow \mathrm{R}W_i y_i + W_i \mathrm{R}y_i + \mathrm{R}b_i$ (product rule)
  $\mathrm{R}y_{i+1} \leftarrow \mathrm{R}x_{i+1} s'_{i+1}(x_{i+1})$
end for
$\mathrm{R}dy_\ell \leftarrow \frac{\partial^2 L(y_\ell; t_\ell)}{\partial y_\ell^2}\mathrm{R}y_\ell$
for all i **from** $\ell - 1$ **downto** 1 **do**
  $\mathrm{R}dx_{i+1} \leftarrow \mathrm{R}dy_{i+1} s'_{i+1}(x_{i+1})$
  $\mathrm{R}dW_i \leftarrow \mathrm{R}dx_{i+1} y_i^\top$
  $\mathrm{R}db_i \leftarrow \mathrm{R}dx_{i+1}$
  $\mathrm{R}dy_i \leftarrow \mathrm{R}W_i^\top dx_{i+1}$
end for
output: $(\mathrm{R}dW_1, \ldots, \mathrm{R}dW_{\ell-1}, \mathrm{R}b_1, \ldots, \mathrm{R}b_{\ell-1})$.

Εικόνα 2.15: Αλγόριθμος για τον υπολογισμό του Gv product όπου L ειναι μία συνάρτηση σφάλματος

Για την υλοποίηση του αλγορίθμου Hessian Free Optimization (HFO) αξιοποιήθηκε ως βάση ανοιχτού κώδικα η βιβλιοθήκη που είναι διαθέσιμη στο GitHub του Rasmussen D [34], η οποία είναι γραμμένη σε γλώσσα C. Αν και η υλοποίηση αυτή προσφέρει ένα ολοκληρωμένο πλαίσιο για την εκπαίδευση νευρωνικών δικτύων με χρήση της συγκεκριμένης τεχνικής, κρίθηκε απαραίτητο να προσαρμοστεί στις ανάγκες της παρούσας εργασίας. Για το σκοπό αυτό, μετατράπηκαν και παραμετροποιήθηκαν οι μέθοδοι που ήταν απαραίτητες, σε γλώσσα Java, ώστε να είναι συμβατές με την υπόλοιπη δομή του συστήματος. Οι τεχνικές λεπτομέρειες και η εξειδικευμένη περιγραφή των μεθόδων που υλοποιήθηκαν παρουσιάζονται αναλυτικά στην Ενότητα 3.6 της παρούσας εργασίας.

Κεφάλαιο 3

Σχεδιασμός και Υλοποίηση

3.1 Δεδομένα εισόδου

Σε αυτό το στάδιο της διπλωματικής μου εργασίας, παρουσιάζονται τα δεδομένα εισόδου που χρησιμοποιούνται για την εκπαίδευση των τριών δικτύων μου Radial Basis Function (RBF), MLP με Backpropagation και HFO, με σκοπό την πρόβλεψη του αριθμού των γκολ που ενδέχεται να σημειωθούν σε κάθε ποδοσφαιρικό αγώνα.

Η βάση των δεδομένων αντλήθηκε από την αγγλική Championship για τις περιόδους 2006 έως 2023, και η επιλογή των χαρακτηριστικών έγινε με βάση προηγούμενες μελέτες που είχαν εστιάσει στο ίδιο πρόβλημα πρόβλεψης γκολ, ακολουθώντας διαφορετικές μεθοδολογίες. Παρότι υπάρχουν πολυάριθμες μεταβλητές που θεωρητικά θα μπορούσαν να χρησιμοποιηθούν, η εμπειρία από προηγούμενες εργασίες, όπως της Turner, του Νεοκλέους, του Ιακώβου και του Φράγκου, δείχνει πως τα πιο αξιόπιστα χαρακτηριστικά είναι αυτά που σχετίζονται άμεσα με τη σκοράρισμα και την αμυντική επίδοση των ομάδων. Δηλαδή, αριθμοί και ποσοστά που αφορούν τα γκολ που πετυχαίνουν και δέχονται οι ομάδες, τόσο σε εντός όσο και εκτός έδρας αγώνες.

Αντίθετα, μεταβλητές όπως οι απουσίες παικτών, η προϊστορία, το κίνητρο ή οι καιρικές συνθήκες, παρότι ίσως φαίνονται σημαντικές, δεν αποδείχθηκαν ωφέλιμες. Αντίθετα, προσέθεταν πολυπλοκότητα και θόρυβο στο μοντέλο, επιβαρύνοντας την ακρίβεια της εκπαίδευσης. Για τον λόγο αυτό, στη συγκεκριμένη εργασία έγινε συνειδητή επιλογή να συμπεριληφθούν μόνο στατιστικά που έχουν άμεση επιρροή στον αριθμό των γκολ που αναμένεται να σημειωθούν.

Στην επόμενη ενότητα παρατίθενται αναλυτικά τα χαρακτηριστικά που συγκροτούν το διάνυσμα εισόδου κάθε αγώνα, όπως προκύπτουν από την ανάλυση των ιστορικών στοιχείων κάθε ομάδας.

1. **Μέσος όρος τερμάτων της γηπεδούχου ομάδας στην έδρα της**
   Υπολογίζει τον αριθμό των τερμάτων που σημείωσε η ομάδα στους εντός έδρας αγώνες, διαίρεση με το πλήθος αυτών των αγώνων. Αποτελεί έναν βασικό δείκτη επιθετικής απόδοσης εντός έδρας και βοηθά στην αξιολόγηση της δύναμης της ομάδας όταν αγωνίζεται στο γήπεδό της.

2. **Ποσοστό εντός έδρας αγώνων στους οποίους σκόραρε η γηπεδούχος**
   Εκφράζει σε τι ποσοστό των εντός έδρας αναμετρήσεων η ομάδα κατάφερε να σκοράρει τουλάχιστον ένα γκολ. Αντικατοπτρίζει τη συνέπεια της επιθετικής της λειτουργίας μπροστά στο κοινό της.

3. **Μέσος όρος τερμάτων που δέχτηκε η γηπεδούχος στην έδρα της**
   Μετράει την αμυντική σταθερότητα της ομάδας στο σπίτι της, υπολογίζοντας πόσα γκολ δέχεται κατά μέσο όρο ανά εντός έδρας αγώνα. Δείχνει την τάση για παθητικό σε συνθήκες έδρας.

4. **Ποσοστό εντός έδρας αγώνων όπου δέχτηκε γκολ**
   Ποσοστιαία ένδειξη των αγώνων στους οποίους η ομάδα δέχτηκε τουλάχιστον ένα γκολ εντός έδρας. Αποτελεί δείκτη της αμυντικής της ευθραυστότητας.

5. **Μέσος όρος τερμάτων εκτός έδρας για τη φιλοξενούμενη ομάδα**
   Υπολογίζει τον μέσο αριθμό γκολ που πετυχαίνει η ομάδα όταν αγωνίζεται

μακριά από την έδρα της. Ενδεικτικό του πόσο επικίνδυνη είναι επιθετικά στις εξορμήσεις της.

6. **Ποσοστό εκτός έδρας αγώνων όπου σκόραρε η φιλοξενούμενη ομάδα**
   Αντικατοπτρίζει το ποσοστό των εκτός έδρας αγώνων στους οποίους η ομάδα κατάφερε να σκοράρει, αποκαλύπτοντας την αξιοπιστία της εκτός έδρας στο σκοράρισμα.

7. **Μέσος όρος τερμάτων που δέχτηκε η φιλοξενούμενη ομάδα εκτός έδρας**
   Εκφράζει την αμυντική απόδοση της ομάδας εκτός έδρας, δείχνοντας αν αντιμετωπίζει προβλήματα ανασταλτικά στα ματς που δεν έχει το πλεονέκτημα της έδρας.

8. **Ποσοστό εκτός έδρας αγώνων όπου δέχτηκε γκολ η φιλοξενούμενη ομάδα**
   Πόσοι από τους εκτός έδρας αγώνες συνοδεύτηκαν από παθητικό. Χρησιμοποιείται για να μετρηθεί η συχνότητα με την οποία δέχεται γκολ σε ξένα γήπεδα.

9. **Ποσοστό εντός έδρας αγώνων όπου η γηπεδούχος πέτυχε πάνω από 2 τέρματα**
   Μετράει πόσες φορές η γηπεδούχος είχε "εκρηκτική" επιθετική απόδοση εντός έδρας, σκοράροντας περισσότερα από 2 τέρματα.

10. **Μέσος όρος τερμάτων της γηπεδούχου στα τελευταία 3 εντός έδρας παιχνίδια**
    Αντικατοπτρίζει την πρόσφατη επιθετική φόρμα της ομάδας στο γήπεδό της, παρέχοντας ένδειξη για το αν βρίσκεται σε καλό επιθετικό μομέντουμ.

11. **Ποσοστό αγώνων με γκολ στα τελευταία 3 εντός έδρας παιχνίδια της γηπεδούχου**
    Καταγράφει σε πόσους από τους τελευταίους 3 αγώνες στο γήπεδο της κατάφερε να σκοράρει. Ένδειξη συνέπειας στην τρέχουσα περίοδο.

12. **Μέσος όρος τερμάτων που δέχτηκε στα τελευταία 3 εντός έδρας παιχνίδια**

    Δείχνει την πρόσφατη αμυντική σταθερότητα ή αδυναμία της ομάδας στην έδρα της.

13. **Ποσοστό αγώνων όπου δέχτηκε γκολ στα τελευταία 3 εντός έδρας παιχνίδια**

    Πόσο συχνά δέχεται γκολ στα τελευταία της ματς εντός, δίνοντας αίσθηση του παρόντος αμυντικού ρυθμού.

14. **Μέσος όρος τερμάτων της φιλοξενούμενης στα τελευταία 3 εκτός έδρας παιχνίδια**

    Καταγράφει την παραγωγικότητα της ομάδας εκτός έδρας στα πιο πρόσφατα ματς.

15. **Ποσοστό αγώνων όπου σκόραρε στα τελευταία 3 εκτός έδρας παιχνίδια**

    Πόσο συχνά σκοράρει η ομάδα μακριά από την έδρα της στην τρέχουσα φάση.

16. **Μέσος όρος τερμάτων που δέχτηκε η φιλοξενούμενη στα τελευταία 3 εκτός έδρας**

    Εκφράζει την αμυντική ευαλωτότητα της ομάδας στους πρόσφατους εκτός έδρας αγώνες.

17. **Ποσοστό αγώνων με παθητικό στα τελευταία 3 εκτός έδρας παιχνίδια**

    Πόσες φορές η φιλοξενούμενη ομάδα δέχτηκε γκολ στα τελευταία τρία εκτός έδρας ματς.

18. **Ποσοστό των τελευταίων 3 εντός έδρας αγώνων με σκορ πάνω από 2.5 τέρματα**

    Αντικατοπτρίζει τη συχνότητα υψηλού σκορ σε πρόσφατους εντός έδρας αγώνες.

19. **Ποσοστό των τελευταίων 3 εκτός έδρας αγώνων με σκορ πάνω από 2.5 τέρματα**

    Καταγράφει πόσο "ανοιχτά" είναι τα εκτός έδρας ματς της ομάδας σε ό,τι αφορά τα τέρματα.

20. **Μέσος όρος τερμάτων σε όλα τα παιχνίδια της ομάδας μέχρι τη συγκεκριμένη αγωνιστική**

    Δίνει τη συνολική επιθετική εικόνα της ομάδας ανεξαρτήτως έδρας, στη διάρκεια της σεζόν.

21. **Ποσοστό όλων των αγώνων που είχαν συνολικά πάνω από 2.5 τέρματα**

    Αντικατοπτρίζει το πόσο "πλούσιο σε γκολ" είναι το γενικό αγωνιστικό προφίλ της ομάδας.

22. **Ποσοστό των τελευταίων 6 αγώνων με σκορ πάνω από 2.5 τέρματα**

    Εστιάζει στην πρόσφατη τάση των αγώνων της ομάδας ως προς το αν παράγονται πολλά τέρματα.

23. **Σερί αγώνων με πάνω από 2.5 γκολ**

    Μετράει συνεχόμενους αγώνες όπου σημειώθηκαν τουλάχιστον 3 γκολ συνολικά, ένδειξη για επίθεση/άμυνα και των δύο ομάδων.

24. **Σερί αγώνων με κάτω από 2.5 γκολ**

    Μετράει συνεχόμενα ματς με περιορισμένο αριθμό τερμάτων, δηλαδή αμυντικά/κλειστά παιχνίδια.

25. **Σερί εντός έδρας αγώνων με πάνω από 2.5 γκολ**

    Διαδοχικά ματς εντός έδρας με σκορ άνω των 2.5, ένδειξη επιθετικής κυριαρχίας εντός.

26. **Σερί εντός έδρας αγώνων με κάτω από 2.5 γκολ**

    Πολλαπλοί εντός έδρας αγώνες που έχουν περιορισμένο αριθμό τερμάτων, πιθανή ένδειξη αμυντικής στρατηγικής ή αδυναμίας στο σκοράρισμα.

27. **Σερί εκτός έδρας αγώνων με πάνω από 2.5 γκολ**

    Σειρά εκτός έδρας αναμετρήσεων με υψηλό αριθμό γκολ, ένδειξη "ανοιχτού" παιχνιδιού εκτός.

28. **Σερί εκτός έδρας αγώνων με κάτω από 2.5 γκολ**

    Διαδοχικά εκτός έδρας παιχνίδια που είχαν λίγα γκολ, δείγμα περιορισμένης επιθετικής παρουσίας ή ισχυρής άμυνας.

29. **Ποσοστό εντός έδρας αγώνων όπου η γηπεδούχος ομάδα σημείωσε πάνω από 2 γκολ.**

    πόσο συχνά η ομάδα είχε εξαιρετικά παραγωγικές εμφανίσεις στην έδρα της, σκοράροντας τουλάχιστον 3 γκολ.

3.2 Το πρωτάθλημα της Championship

Το πρωτάθλημα της Championship, το οποίο αποτελεί τη δεύτερη τη τάξει κατηγορία του αγγλικού ποδοσφαίρου, θεωρείται ένα από τα πιο δύσκολα και ανταγωνιστικά πρωταθλήματα της Ευρώπης. Χαρακτηρίζεται από έντονο ρυθμό, δυναμικό στυλ παιχνιδιού και σημαντικό στοιχηματικό ενδιαφέρον, αφού διαθέτει μεγάλο αριθμό ομάδων με ιστορία και φιλοδοξίες για άνοδο στην Premier League. Ακριβώς λόγω αυτών των χαρακτηριστικών και του όγκου των διαθέσιμων δεδομένων, αποφάσισα να βασίσω τη διπλωματική μου εργασία σε αυτό το πρωτάθλημα, με την ελπίδα ότι η πολυπλοκότητά του θα αναδείξει την αποτελεσματικότητα των μεθόδων που ανέπτυξα.

Στη διοργάνωση αυτή συμμετέχουν κάθε χρόνο 24 ομάδες. Αυτό σημαίνει ότι σε κάθε αγωνιστική διεξάγονται 12 παιχνίδια, με κάθε ομάδα να αντιμετωπίζει όλες τις υπόλοιπες δύο φορές – μία εντός και μία εκτός έδρας. Συνολικά, κάθε ποδοσφαιρική σεζόν περιλαμβάνει 552 παιχνίδια. Ωστόσο, προκειμένου να υπολογιστούν οι στατιστικές μου μεταβλητές, είναι απαραίτητο να υπάρχουν δεδομένα από τουλάχιστον έξι προηγούμενες αγωνιστικές για κάθε ομάδα. Έτσι, οι πρώτες έξι αγωνιστικές δεν

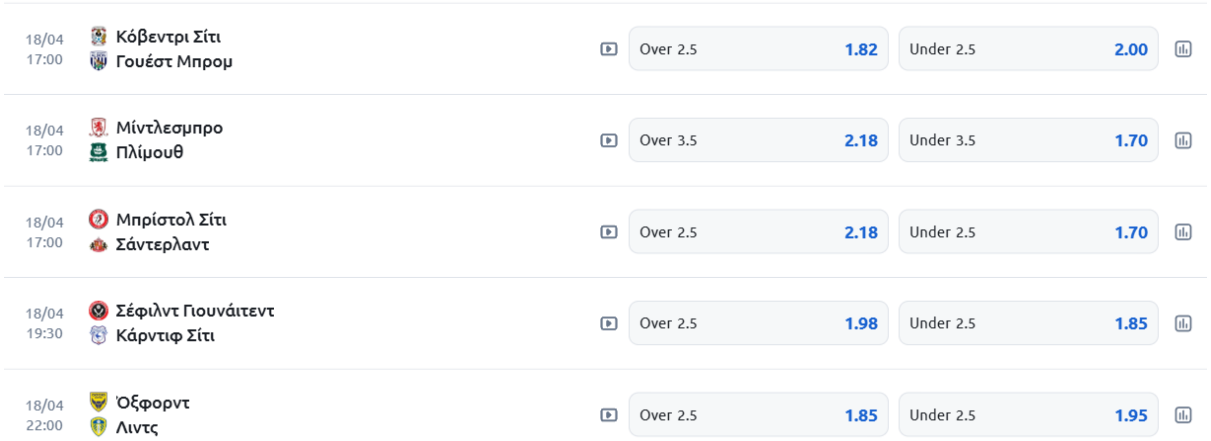
συμπεριλαμβάνονται στα δεδομένα εκπαίδευσης και τελικά για κάθε περίοδο λαμβάνονται περίπου 480 παιχνίδια ως είσοδοι στο σύστημα πρόβλεψης.

#	Team	P	W	D	L	DIFF	Goals	Last 5	PTS
1	Leicester	46	31	4	11	+48	89:41	L W W W L	97
2	Ipswich	46	28	12	6	+35	92:57	D D D W W	96
3	Leeds	46	27	9	10	+38	81:43	D L W L L	90
4	Southampton	46	26	9	11	+24	87:63	W L L L W	87
5	West Brom	46	21	12	13	+23	70:47	W L L L W	75
6	Norwich	46	21	10	15	+15	79:64	D W D D L	73
7	Hull	46	19	13	14	+8	68:60	W D W D L	70
8	Middlesbrough	46	20	9	17	+9	71:62	D D L W W	69
9	Coventry	46	17	13	16	+11	70:59	L L D L L	64
10	Preston	46	18	9	19	-11	56:67	L L L L L	63
11	Bristol City	46	17	11	18	+2	53:51	W D D W L	62
12	Cardiff	46	19	5	22	-17	53:70	W L W L L	62
13	Millwall	46	16	11	19	-10	45:55	W W W W W	59
14	Swansea	46	15	12	19	-6	59:65	W W W D L	57
15	Watford	46	13	17	16	0	61:61	D L D W L	56
16	Sunderland	46	16	8	22	-2	52:54	D W L L L	56
17	Stoke	46	15	11	20	-11	49:60	L D W W W	56
18	QPR	46	15	11	20	-11	47:58	D L W W W	56
19	Blackburn	46	14	11	21	-14	60:74	L W L D W	53
20	Sheffield Wed	46	15	8	23	-24	44:68	D D W W W	53
21	Plymouth	46	13	12	21	-11	59:70	D W L L W	51
22	Birmingham	46	13	11	22	-15	50:65	L W D D W	50
23	Huddersfield	46	9	18	19	-29	48:77	L D L D L	45
24	Rotherham	46	5	12	29	-52	37:89	L L D L W	27

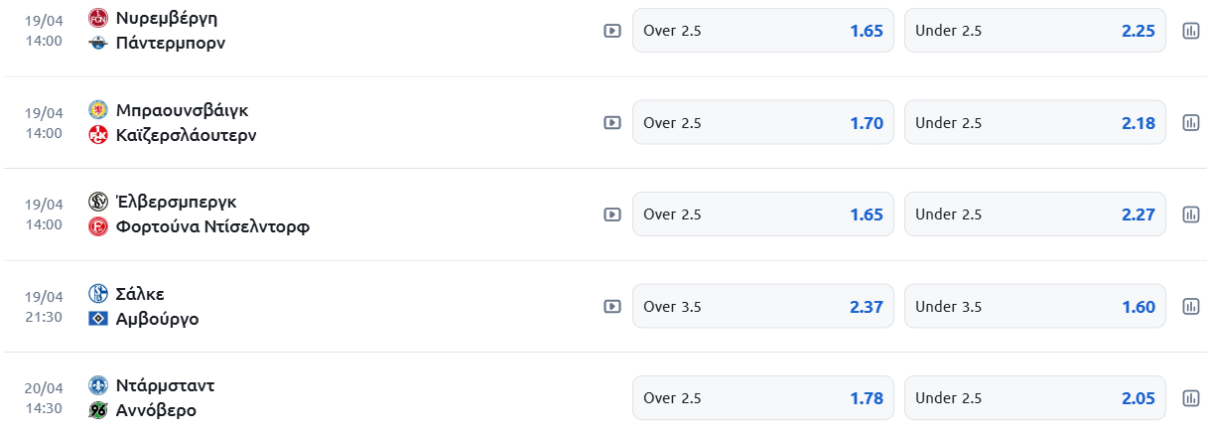
Εικόνα 3.1: Τελική βαθμολογία της championship σεζόν 2023/24

Ένας ακόμη σημαντικός λόγος που επέλεξα να εφαρμόσω την παρούσα μελέτη στο πρωτάθλημα της Championship Αγγλίας, είναι η σχετική ισορροπία που παρατηρείται στις αποδόσεις του στοιχήματος για το over και το under 2.5 γκολ. Όπως φαίνεται και στις ενδεικτικές αποδόσεις που παρουσιάζονται(εικόνα 3.2) σε αρκετά παιχνίδια του αγγλικού πρωταθλήματος, οι τιμές για τις δύο αυτές επιλογές είναι πολύ κοντά μεταξύ τους, γεγονός που υποδηλώνει την αβεβαιότητα των bookers σχετικά με την τελική έκβαση στο σκορ και την ισορροπία στο προφίλ των ομάδων. Αντίθετα, στο πρωτάθλημα της Β' κατηγορίας Γερμανίας (2. Bundesliga), η διαφορά μεταξύ των αποδόσεων του over και του under είναι συχνά μεγάλη(εικόνα 3.3), κάτι που αντικατοπτρίζει έναν πιο ξεκάθαρο προσανατολισμό ως προς την πιθανή έκβαση, μειώνοντας έτσι την πρόκληση της πρόβλεψης και την ανάγκη για εκλεπτυσμένη ανάλυση. Η πιο ισορροπημένη κατανομή τιμών στην Championship ενισχύει τη σημασία της πρόβλεψης, καθώς αποκαλύπτει την

ανάγκη για ισχυρότερα μοντέλα εκτίμησης, δίνοντας έτσι αυξημένη αξία στη χρήση τεχνικών όπως τα νευρωνικά δίκτυα



Εικόνα 3.2: Τυπική αγωνιστική του πρωταθλήματος της Championship



Εικόνα 3.3: Τυπική αγωνιστική του πρωταθλήματος της 2 Bundesliga(2η κατηγορία Γερμανίας)

3.3 Λήψη και επεξεργασία δεδομένων

Η διαδικασία λήψης και επεξεργασίας των δεδομένων αποτελεί ένα από τα πιο κρίσιμα στάδια της διπλωματικής εργασίας, καθώς η ποιότητα και η σωστή αναπαράσταση των δεδομένων επηρεάζουν καθοριστικά την απόδοση των αλγορίθμων πρόβλεψης. Τα δεδομένα της παρούσας υλοποίησης αντλούνται από την αξιόπιστη ιστοσελίδα football-data.co.uk, η οποία παρέχει ιστορικά στατιστικά για το αγγλικό πρωτάθλημα Championship.

Εικόνα 3.4: Σελίδα απο την οποία κατεβάζω τα .csv αρχεία για την κάθε σεζόν

Για κάθε ποδοσφαιρική σεζόν από το 2006 έως και το 2023, γίνεται λήψη του αντίστοιχου CSV αρχείου, στο οποίο περιλαμβάνονται δεδομένα όπως:

- Ονόματα γηπεδούχου και φιλοξενούμενης ομάδας,
- Γκολ που σημειώθηκαν από κάθε ομάδα,
- Η τελική έκβαση του αγώνα (H, A, D),
- Οι αποδόσεις για over/under 2.5 τέρματα. Κτλ

	A	B	C	D	E	F	G	H	I	J	K	L
1	Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	HS
2	E1	5/8/2016	Fulham	Newcastle	1	0	H	1	0	H	S Hooper	5
3	E1	6/8/2016	Birmingham	Cardiff	0	0	D	0	0	D	T Robinson	14
4	E1	6/8/2016	Blackburn	Norwich	1	4	A	0	3	A	T Harringtc	5
5	E1	6/8/2016	Bristol City	Wigan	2	1	H	0	1	A	J Adcock	24
6	E1	6/8/2016	Derby	Brighton	0	0	D	0	0	D	J Simpson	12
7	E1	6/8/2016	Huddersfie	Brentford	2	1	H	0	0	D	G Eltringha	16
8	E1	6/8/2016	Ipswich	Barnsley	4	2	H	0	0	D	K Stroud	9
9	E1	6/8/2016	Nott'm For	Burton	4	3	H	2	2	D	A Woolmer	14
10	E1	6/8/2016	Reading	Preston	1	0	H	1	0	H	A Davies	15
11	E1	6/8/2016	Rotherham	Wolves	2	2	D	2	1	H	S Duncan	6
12	E1	7/8/2016	QPR	Leeds	3	0	H	1	0	H	O Langford	15
13	E1	7/8/2016	Sheffield W	Aston Villa	1	0	H	0	0	D	S Martin	17

Εικόνα 3.5: Μικρο μέρος του .csv για την σεζόν 2017

Για να αξιοποιηθούν αυτά τα δεδομένα, χρησιμοποιείται Java κώδικας που διαβάζει γραμμή προς γραμμή το CSV αρχείο της κάθε σεζόν και εξάγει τις απαραίτητες πληροφορίες. Συγκεκριμένα, για κάθε γραμμή (δηλαδή για κάθε αγώνα), το πρόγραμμα εντοπίζει τις στήλες που περιλαμβάνουν την ομάδα που αγωνίζεται εντός έδρας (Home Team), την ομάδα που αγωνίζεται εκτός έδρας (Away Team), τα τερμάτα που πέτυχε η κάθε ομάδα (FTHG και FTAG), καθώς και τις αποδόσεις για το στοίχημα Over 2.5 και Under 2.5 τέρματα (συνήθως στις στήλες B365>2.5 και B365<2.5). Μόλις συλλεχθούν αυτά τα στοιχεία, δημιουργείται ένα νέο αρχείο .txt για κάθε σεζόν. Η δημιουργία αυτών των `.txt` αρχείων γίνεται αυτόματα με χρήση της κλάσης FileEditor.java, η οποία φροντίζει ώστε τα δεδομένα να είναι καθαρά, σωστά μορφοποιημένα και απαλλαγμένα από ελλείψεις ή ασυνέπειες (π.χ. κενά σκορ ή ελλείπουσες αποδόσεις). Η διαδικασία αυτή επαναλαμβάνεται για κάθε σεζόν ξεχωριστά, με αποτέλεσμα να δημιουργείται ένα .txt αρχείο ανά χρονιά, το οποίο στη συνέχεια μπορεί να χρησιμοποιηθεί για αναφορά, οπτική επιβεβαίωση ή και περαιτέρω επεξεργασία στα μοντέλα πρόβλεψης. Αυτή η δομημένη προσέγγιση στην επεξεργασία των CSV εξασφαλίζει την ομοιογένεια στα δεδομένα και επιτρέπει την εύκολη διαχείριση, ταξινόμηση και ανάλυση τους, τόσο για σκοπούς προεπεξεργασίας, όσο και για την τελική εκπαίδευση των μοντέλων.

```
Fulham-Newcastle-1:0 1.86/1.91
Birmingham-Cardiff-0:0 2.18/1.65
Blackburn-Norwich-1:4 2.23/1.62
Bristol City-Wigan-2:1 2.13/1.68
Derby-Brighton-0:0 2.13/1.68
Huddersfield-Brentford-2:1 1.8/1.96
Ipswich-Barnsley-4:2 2.06/1.72
Nott'm Forest-Burton-4:3 2.17/1.64
Reading-Preston-1:0 2.37/1.55
Rotherham-Wolves-2:2 2.19/1.64
QPR-Leeds-3:0 2.2/1.63
Sheffield Weds-Aston Villa-1:0 2.25/1.6
Brighton-Nott'm Forest-3:0 2.11/1.7
```

Εικόνα 3.6: Μικρό μέρος του .txt αρχείου για την σεζόν 2017

Στο πλαίσιο της διπλωματικής μου εργασίας, ένα από τα σημαντικότερα βήματα στην προεπεξεργασία των δεδομένων είναι η δημιουργία ενός πολυδιάστατου πίνακα στατιστικών για κάθε ομάδα και για κάθε αγωνιστική της σεζόν. Πιο συγκεκριμένα, για κάθε ομάδα που συμμετέχει στο πρωτάθλημα της Championship, κατασκευάζεται ένας πίνακας που αντιστοιχεί σε όλες τις αγωνιστικές της σεζόν και κάθε γραμμή του περιέχει

τα συνολικά στατιστικά της ομάδας μέχρι και την εκάστοτε αγωνιστική. Κάθε γραμμή αποτελεί ένα διάνυσμα 29 χαρακτηριστικών (columns), τα οποία περιλαμβάνουν σημαντικές μεταβλητές που επηρεάζουν άμεσα το αποτέλεσμα ενός αγώνα και, πιο συγκεκριμένα, τον αριθμό των τερμάτων που μπορεί να σημειωθούν.

Οι 29 αυτές στήλες περιλαμβάνουν ένα μείγμα μέσων όρων, ποσοστών, αλλά και σειρών (streaks) που αναπαριστούν τη φόρμα της ομάδας. Μεταξύ άλλων καταγράφονται: ο μέσος όρος τερμάτων υπέρ και κατά σε εντός και εκτός έδρας αγώνες, ποσοστά σκοραρίσματος και παθητικού σε πρόσφατους αγώνες, επιθετικές και αμυντικές επιδόσεις στους τελευταίους τρεις αγώνες, καθώς και ποσοστά και σερί αγώνων με συνολικό σκορ άνω ή κάτω των 2.5 τερμάτων. Αυτά τα στατιστικά παρέχουν ένα ισχυρό και πολυεπίπεδο προφίλ της ομάδας για την τρέχουσα στιγμή της σεζόν, λαμβάνοντας υπόψη τόσο τα ιστορικά της δεδομένα όσο και τη πρόσφατη φόρμα της.

Η λογική πίσω από τη δημιουργία αυτού του πίνακα είναι να δοθεί στο μοντέλο εκπαίδευσης ένα ρεαλιστικό και συνεχώς μεταβαλλόμενο στιγμιότυπο της κατάστασης κάθε ομάδας, έτσι ώστε οι προβλέψεις του να βασίζονται σε επαρκή πληροφόρηση και όχι σε στατικά ή μεμονωμένα δεδομένα. Ο πίνακας αυτός ενημερώνεται δυναμικά μετά από κάθε αγωνιστική, προσφέροντας έτσι χρονική συνέχεια στην ανάλυση και συμβάλλοντας σημαντικά στην ακρίβεια των προβλέψεων του συστήματος. Η προσέγγιση αυτή ενσωματώνει τη χρονική διάσταση των δεδομένων χωρίς να χρησιμοποιεί ρητά RNN αρχιτεκτονική, ενισχύοντας έτσι τη γενίκευση και την αποτελεσματικότητα των μοντέλων που εκπαιδεύονται με αυτά.

3.4 Κανονικοποίηση Τιμών

Στα τεχνητά νευρωνικά δίκτυα, οι είσοδοι μπορούν θεωρητικά να πάρουν οποιαδήποτε αριθμητική τιμή. Ωστόσο, για να επιτευχθεί πιο αποδοτική και σταθερή εκπαίδευση του δικτύου, είναι κοινή πρακτική οι τιμές εισόδου να κανονικοποιούνται σε ένα προκαθορισμένο εύρος – συνήθως μεταξύ 0 και 1. Αυτή η διαδικασία βοηθά στη διατήρηση των τιμών σε ελεγχόμενα όρια, αποτρέποντας απότομες αποκλίσεις στις εξόδους των νευρώνων και διευκολύνοντας τη σύγκλιση του δικτύου κατά την εκπαίδευση.

Στην παρούσα υλοποίηση, τα δεδομένα εισόδου περιλαμβάνουν στατιστικά όπως μέσους όρους τερμάτων ανά παιχνίδι, που συχνά υπερβαίνουν την τιμή 1. Για παράδειγμα, μια ομάδα μπορεί να σκοράρει κατά μέσο όρο 1.8 ή και 2.5 τέρματα ανά αγώνα, κάτι που ξεπερνά το ανώτατο όριο της τυπικής κλίμακας κανονικοποίησης. Για αυτόν τον λόγο, κρίθηκε αναγκαίο να εφαρμοστεί μια ειδική κανονικοποιητική εξίσωση η οποία φέρνει όλες τις τιμές σε μια ενιαία κλίμακα μεταξύ 0 και 1, χωρίς όμως να παραμορφώνει τις αναλογίες τους. Η εξίσωση αυτή εφαρμόστηκε σε κάθε στατιστικό ξεχωριστά, λαμβάνοντας υπόψη το εκάστοτε μέγιστο και ελάχιστο της συγκεκριμένης μεταβλητής, ώστε να διατηρηθεί η σχετική διαφοροποίηση των δεδομένων κατά την είσοδο στο νευρωνικό δίκτυο.

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

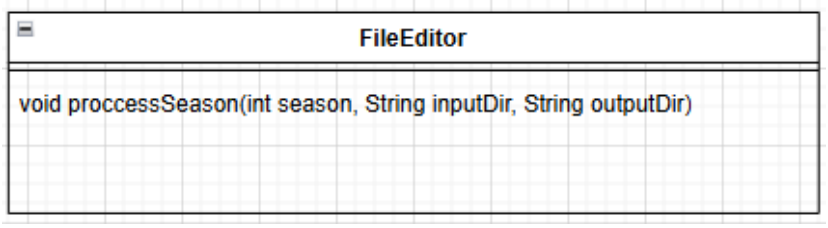
X : αρχική τιμή πριν την κανονικοποίηση.
Xmax: η μεγαλύτερη πιθανή τιμή του συγκεκριμένου στατιστικού πριν την κανονικοποίηση.
Xmin: η μικρότερη πιθανή τιμή του συγκεκριμένου στατιστικού πριν την κανονικοποίηση.
y: η νέα τιμή του στατιστικού.

3.4 Σχεδιασμός και Υλοποίηση του Δυκτίου

Όπως αναφέρθηκε και στην εισαγωγή της παρούσας εργασίας, για την υλοποίηση του νευρωνικού δικτύου επιλέχθηκε η γλώσσα προγραμματισμού Java. Η επιλογή αυτή υπαγορεύει την ανάγκη χρήσης αντικειμενοστραφούς προγραμματιστικού μοντέλου. Στις ενότητες που ακολουθούν παρουσιάζονται αναλυτικά οι βασικές κλάσεις που δημιουργήθηκαν, η λειτουργικότητα που αναλαμβάνει η κάθε μία, τα δεδομένα που διαχειρίζεται (όπως πίνακες στατιστικών, ιστορικά αγώνων, προβλέψεις) καθώς και ο τρόπος με τον οποίο συνεργάζονται μεταξύ τους. Η επικοινωνία και ο συντονισμός αυτών των κλάσεων είναι κρίσιμος, καθώς μαζί συγκροτούν τη συνολική λογική του νευρωνικού δικτύου – από την ανάγνωση των δεδομένων, την επεξεργασία τους και την εκπαίδευση του μοντέλου, μέχρι την πρόβλεψη αποτελεσμάτων και την εξαγωγή συμπερασμάτων.

Κλάση FileEditor

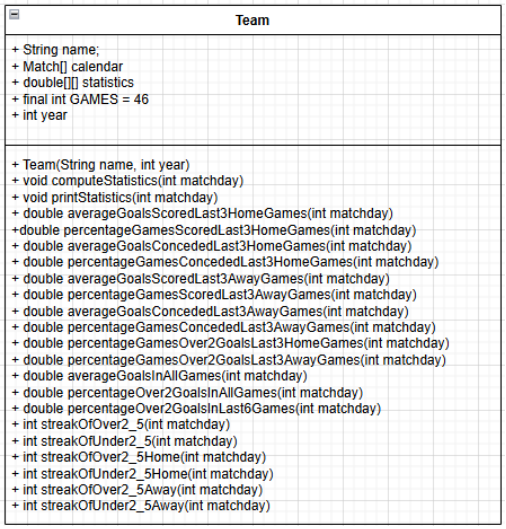
Η κλάση FileEditor (παράρτημα Α) αποτελεί μια βοηθητική κλάση υπεύθυνη για την προεπεξεργασία και μετατροπή αρχείων που περιέχουν δεδομένα αγώνων ποδοσφαίρου, προκειμένου να χρησιμοποιηθούν αργότερα για την εκπαίδευση μοντέλων πρόβλεψης. Περιλαμβάνει μια βασική δημόσια μέθοδο, τη processSeason(), η οποία εκτελεί όλη τη λειτουργικότητα της κλάσης. Πιο συγκεκριμένα, η processSeason() διαβάζει ένα αρχείο .csv που αντιστοιχεί σε μια συγκεκριμένη ποδοσφαιρική σεζόν – για παράδειγμα το E1_2017.csv, που περιλαμβάνει αναλυτικά στοιχεία για κάθε αγώνα της αγωνιστικής περιόδου στην αγγλική Championship. Η μέθοδος αναζητά μέσα στο αρχείο συγκεκριμένες στήλες, όπως η γηπεδούχος και φιλοξενούμενη ομάδα (HomeTeam, AwayTeam), τα τελικά γκολ των δύο ομάδων (FTHG, FTAG) καθώς και τις αποδόσεις για Over 2.5 και Under 2.5 γκολ (BbO.2.5, BbU.2.5). Για κάθε γραμμή του αρχείου δημιουργεί μια νέα μορφοποιημένη εγγραφή που ακολουθεί τη δομή που φαινεται στην εικόνα 3.6 και αποθηκεύεται σε ένα αρχείο `.txt` που φέρει την ονομασία της σεζόν (π.χ. `championship_2017.txt`) και το οποίο αργότερα μπορεί να χρησιμοποιηθεί από άλλες κλάσεις της υλοποίησης για να φορτώσουν και να επεξεργαστούν δεδομένα για σκοπούς εκπαίδευσης νευρωνικών δικτύων ή άλλων μοντέλων μηχανικής μάθησης.



Εικόνα 3.7: UML που δίχνει τα πεδία και τις μεθόδους της FileEditor

Κλάση Team

Η κλάση Team αποτελεί βασική μονάδα της υλοποίησης και είναι υπεύθυνη για την αναπαράσταση μιας ποδοσφαιρικής ομάδας στο σύστημα, καθώς και για τον υπολογισμό όλων των απαραίτητων στατιστικών της κατά τη διάρκεια μιας αγωνιστικής περιόδου. Το μοναδικό της πεδίο είναι το όνομα της ομάδας (name), το οποίο χρησιμοποιείται ως ταυτοποιητής για την ανάκτηση και επεξεργασία των σχετικών δεδομένων. Η κύρια λειτουργικότητα της κλάσης βρίσκεται στη μέθοδο processSeason(Season season), η οποία λαμβάνει ως όρισμα ένα αντικείμενο τύπου Season και επεξεργάζεται τα δεδομένα κάθε αγωνιστικής για την ομάδα. Η μέθοδος αυτή διατρέχει το σύνολο των αγωνιστικών και για κάθε μία υπολογίζει μια σειρά από κρίσιμα στατιστικά που σχετίζονται με την απόδοση της ομάδας, τόσο εντός όσο και εκτός έδρας. Τα στατιστικά αυτά περιλαμβάνουν μέσους όρους τερμάτων υπέρ και κατά, ποσοστά σκοραρίσματος ή παθητικού, ποσοστά αγώνων με πάνω από 2.5 γκολ, αποδόσεις των τελευταίων αγώνων, καθώς και σερί με βάση την έκβαση των σκορ. Τα αποτελέσματα αυτά αποθηκεύονται σε έναν δισδιάστατο πίνακα στατιστικών, όπου κάθε γραμμή αντιστοιχεί σε μια αγωνιστική και κάθε στήλη σε ένα συγκεκριμένο χαρακτηριστικό — συνολικά 29 διαφορετικά χαρακτηριστικά για κάθε ομάδα σε κάθε αγωνιστική. Η Team είναι πλήρως ενσωματωμένη στη λογική του προγράμματος καθώς από την έξοδο αυτής της κλάσης προκύπτουν τα διανύσματα εισόδου που χρησιμοποιούνται για την εκπαίδευση και δοκιμή των μοντέλων πρόβλεψης (όπως τα RBF, MLP ή HFO). Με άλλα λόγια, πρόκειται για την κλάση που γεφυρώνει τα ακατέργαστα δεδομένα της σεζόν με τα πρότυπα μηχανικής μάθησης, επιτρέποντας την αυτόματη παραγωγή χαρακτηριστικών ανά αγωνιστική με βάση τις ιστορικές επιδόσεις της κάθε ομάδας.



Εικόνα 3.8: UML που δίχνει τα πεδία και τις μεθόδους της Team

	0	1	2	3	....	28
0						
1						
2						
3						
...						
45						

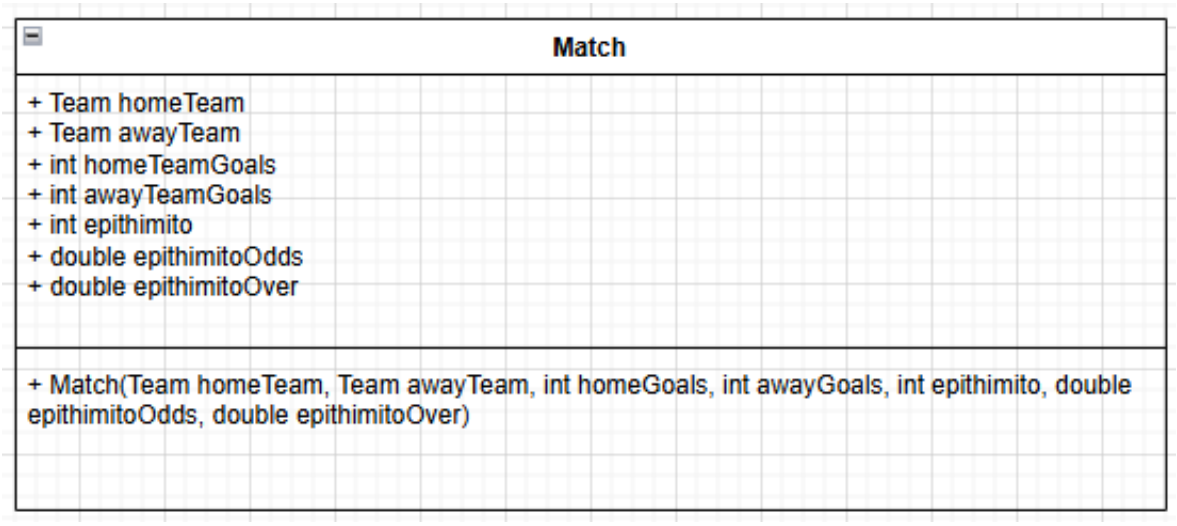
Πίνακας **double statistics[46][29]**. Κάθε στήλη του πίνακα αναπαριστά ένα απο τα 29 διαφορετικά στατιστικά και κάθε γραμμή του πίνακα αναπαριστά για πια αγωνιστική ισχύουν τα στατιστικά.

0	
1	
2	
3	
...	
45	

Πίνακας **Match calendar[46]**. Κάθε γραμμή αναπαριστά τον αγώνα της ομάδας την συγκεκριμένη αγωνιστική

Κλάση Match

Η κλάση Match αποτελεί μια βασική δομή στην εφαρμογή και χρησιμοποιείται για την αναπαράσταση ενός ποδοσφαιρικού αγώνα μεταξύ δύο ομάδων. Συγκεκριμένα, το κάθε αντικείμενο της κλάσης περιέχει πληροφορίες για τις δύο συμμετέχουσες ομάδες, το σκορ του αγώνα καθώς και τις σχετικές αποδόσεις που σχετίζονται με στοιχηματικές επιλογές όπως το Over και το Under. Το πεδίο homeTeam αναφέρεται στην ομάδα που αγωνίζεται στην έδρα της, ενώ το awayTeam στην αντίπαλη φιλοξενούμενη ομάδα. Τα πεδία homeTeamGoals και awayTeamGoals καταγράφουν αντίστοιχα τον αριθμό των τερμάτων που σημείωσαν οι δύο ομάδες σε αυτό το συγκεκριμένο παιχνίδι. Το πεδίο epithimito δηλώνει το επιθυμητό αποτέλεσμα αναφορικά με το πλήθος των τερμάτων (συνολο τερμάτων που σημειώθηκαν στο ματς) , ενώ το epithimitoOdds περιλαμβάνει την αντίστοιχη απόδοση που είχε δοθεί από τον bookmaker για την επιλογή αυτή. Τέλος, το epithimitoOver καταγράφει 1 αν το επιθιμιτό αποτέλεσμα είναι over 2,5 και 0 αν το επιθιμιτο αποτέλεσμα είναι under 2,5. Η κλάση αυτή συνδυάζει τόσο αθλητικά όσο και στοιχηματικά δεδομένα, καθιστώντας την κρίσιμη για τη διαδικασία πρόβλεψης αποτελεσμάτων, την εκπαίδευση νευρωνικών δικτύων και την αξιολόγηση αποδόσεων.



Εικόνα 3.9: UML που δίχνει τα πεδία και τις μεθόδους της Match

Κλάση Season

Η κλάση Season στην Java αναπαριστά τη λειτουργία και τη δομή ενός πρωταθλήματος ποδοσφαίρου για μία συγκεκριμένη αγωνιστική περίοδο. Ο ρόλος της είναι κεντρικός στην αρχιτεκτονική της εφαρμογής, καθώς χειρίζεται την ανάγνωση, αποθήκευση, διαχείριση και προεπεξεργασία των δεδομένων που σχετίζονται με τους αγώνες και τις ομάδες.

Κατά την αρχικοποίηση της κλάσης μέσω του constructor, δίνεται ως είσοδος η χρονιά της σεζόν και ο φάκελος στον οποίο βρίσκεται το αρχείο δεδομένων. Το αρχείο περιλαμβάνει όλες τις αναμετρήσεις της σεζόν, μία ανά γραμμή, στη μορφή Ομάδα1 - Ομάδα2 - Σκορ ΑπόδοσηOver/ΑπόδοσηUnder. Ο constructor διαβάζει το αρχείο αυτό, εξάγει όλα τα ονόματα ομάδων και δημιουργεί έναν πίνακα league από αντικείμενα τύπου Team, διασφαλίζοντας ότι κάθε ομάδα καταχωρείται μόνο μία φορά μέσω ενός Set<String> που ονομάζεται teamSet.

Η κλάση περιλαμβάνει και ένα δισδιάστατο πίνακα games[46][12], ο οποίος αναπαριστά όλα τα παιχνίδια του πρωταθλήματος – 46 αγωνιστικές με 12 παιχνίδια ανά εβδομάδα (σύμφωνα με την Championship Αγγλίας). Κάθε στοιχείο του πίνακα περιέχει ένα αντικείμενο τύπου Match, το οποίο κρατά πληροφορίες για το σκορ, τις αποδόσεις και τις ομάδες που συμμετείχαν.

Η βασική μέθοδος gamesPlayed(int gameweek) είναι υπεύθυνη για τη δημιουργία και αποθήκευση των αγώνων μέχρι και την επιλεγμένη αγωνιστική. Διαβάζει το αρχείο και για κάθε αγώνα βρίσκει τις αντίστοιχες ομάδες, εξάγει το σκορ και τις αποδόσεις, και δημιουργεί αντικείμενα Match. Τα αντικείμενα αυτά αποθηκεύονται τόσο στον πίνακα games όσο και στα calendar[] των αντίστοιχων ομάδων, επιτρέποντας έτσι την αμφίδρομη συσχέτιση των αγώνων με τις ομάδες.

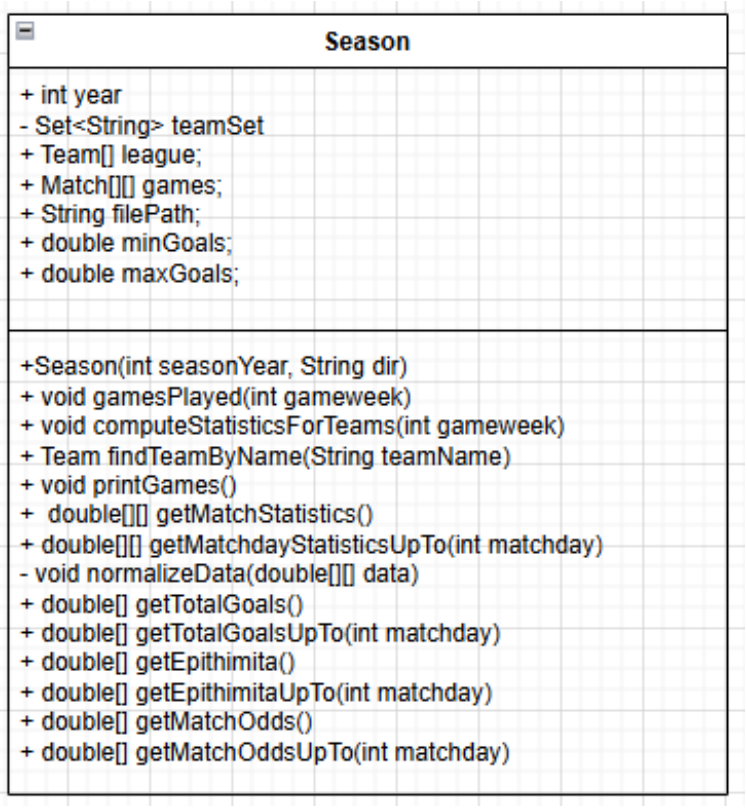
Η μέθοδος computeStatisticsForTeams() καλεί την αντίστοιχη μέθοδο για κάθε ομάδα και υπολογίζει ένα πίνακα στατιστικών statistics[matchday][29], ο οποίος περιλαμβάνει 29 χαρακτηριστικά που περιγράφουν επιθετική και αμυντική συμπεριφορά εντός και εκτός έδρας, πρόσφατη φόρμα και σειρές αγώνων. Οι πίνακες statistics αποθηκεύονται σε κάθε ομάδα και ανανεώνονται εβδομαδιαία.

Οι μέθοδοι getMatchStatistic , getMatchOdds, getTotalGoals, getEpithimita και παραλλαγές τους επιτρέπουν την εξαγωγή δεδομένων εισόδου και εξόδου για την εκπαίδευση και αξιολόγηση του νευρωνικού δικτύου. Η μέθοδος getMatchStatistic επιστρέφει διανύσματα 34 διαστάσεων για κάθε ένα από τα στατιστικά που επηρεάζουν το αποτέλεσμα του αγώνα. Τα μισά από τα στατιστικά του διανύσματος εισόδου των δικτύων είναι για τα στατιστικά που αφορούν την γηπεδούχο ομάδα, τα άλλα μισά είναι

για τα στατιστικά που αφορούν την φιλοξενούμενη ομάδα και κάποια αφορούν στατιστικά των 2 αντίπαλων ομάδων ανεξαρτήτου έδρας. Οι μεθόδοι getTotalGoals, getEpithimita αφορούν διανύσματα επιθυμητών αποτελεσμάτων για όλα τα παιχνίδια με τα οποία θα εκπαιδευτούν τα νευρωνικά δίκτυα. Η μέθοδος getMatchOdds δίνεται ως είσοδο στην αξιολόγηση του δικτύου για να υπολογιστεί το συνολικό κέρδος από τις σωστές προβλέψεις. Περιλαμβάνουν κανονικοποίηση των δεδομένων με normalizeData όπως και δυναμική επιλογή των εβδομάδων από τις οποίες προκύπτουν τα δεδομένα.

Τέλος, οι μεταβλητές minGoals και maxGoals αποθηκεύουν τα άκρα των τερμάτων για κάθε αγώνα, χρήσιμα για την κανονικοποίηση της εξόδου στα προβλήματα πρόβλεψης συνλικού αριθμού τερμάτων.

Συνοπτικά, η Season αποτελεί τον βασικό μηχανισμό συλλογής, οργάνωσης, επεξεργασίας και εξαγωγής των ποδοσφαιρικών δεδομένων, παρέχοντας όλα τα απαραίτητα εργαλεία για την προετοιμασία τους προς χρήση από μοντέλα μηχανικής μάθησης.



Εικόνα 3.10: UML που δίχνει τα πεδία και τις μεθόδους της Season

	0	1	2	3	....	11
0						
1						
2						
3						
...						
45						

Πίνακας **Match games[46][12]**. Κάθε γραμμή του πίνακα αναπαριστά μια αγωνιστική και κάθε στήλη αναπαριστά έναν αγώνα.

0	
1	
2	
3	
...	
23	

Πίνακας **Team league[24]**. Κάθε γραμμή του πίνακα αναπαριστά μια ομάδα που αγωνίζεται την τρέχουσα σεζόν στο πρωτάθλημα.

Κλάση MLP

Η κλάση MLP (παράρτημα Β) που περιλαμβάνεται στο αρχείο σου αποτελεί τον βασικό κορμό της υλοποίησης ενός τεχνητού νευρωνικού δικτύου με δυνατότητα χειρισμού προβλημάτων πρόβλεψης συνολικού αριθμολυ τερμάτων και δυαδικής ταξινόμησης. Χρησιμοποιείται για την πρόβλεψη στοιχηματικών αποτελεσμάτων (Over/Under 2.5 τέρματα) και υποστηρίζει αρχιτεκτονικές με ένα ή δύο κρυφά επίπεδα. Η όλη σχεδίαση ακολουθεί αντικειμενοστραφή προγραμματισμό και είναι παραμετροποιήσιμη σε όλα τα βασικά της σημεία.

Βασικά Attributes (πεδία):

- inputSize, hiddenSize1, hiddenSize2, outputSize: Καθορίζουν τις διαστάσεις των επιπέδων του δικτύου. Αν χρησιμοποιείται μόνο ένα κρυφό επίπεδο, τότε το hiddenSize2 αγνοείται.
- weights1, weights2, weightsOutput: Πίνακες με τα βάρη μεταξύ των επιπέδων.
- bias1, bias2, biasOutput: Μετατοπίσεις (biases) που προστίθενται στις εξόδους κάθε επιπέδου.
- useTwoHiddenLayers: Boolean flag που δηλώνει αν θα χρησιμοποιηθεί δεύτερο κρυφό επίπεδο.
- random: Αντικείμενο για τυχαία αρχικοποίηση των βαρών.

Προώθηση (Forward Pass)

- public double forward(double[] input): Εκτελεί ένα forward pass για πρόβλεψη συνεχών τιμών (πρόβλεψη συνολικού αριθμού τερμάτων). Χρησιμοποιεί ReLU στα κρυφά επίπεδα και καμία ενεργοποίηση στην έξοδο.
- public double forwardSigm(double[] input): Εκτελεί forward pass για προβλήματα δυαδικής ταξινόμησης. Η έξοδος περνάει από συνάρτηση Sigmoid για να δοθεί πιθανότητα.

Εκπαίδευση (Training)

- public void train(...): Εκπαιδεύει το MLP για πρόβλεψη συνολικού αριθμού τερμάτων. Περιλαμβάνει forward pass, υπολογισμό σφάλματος, υπολογισμό παραγώγων, και ενημέρωση βαρών μέσω backpropagation.
- public void trainInt(...): Αντίστοιχη μέθοδος για δυαδική ταξινόμηση. Χρησιμοποιεί Sigmoid στην έξοδο και διαχειρίζεται ετικέτες 0/1.

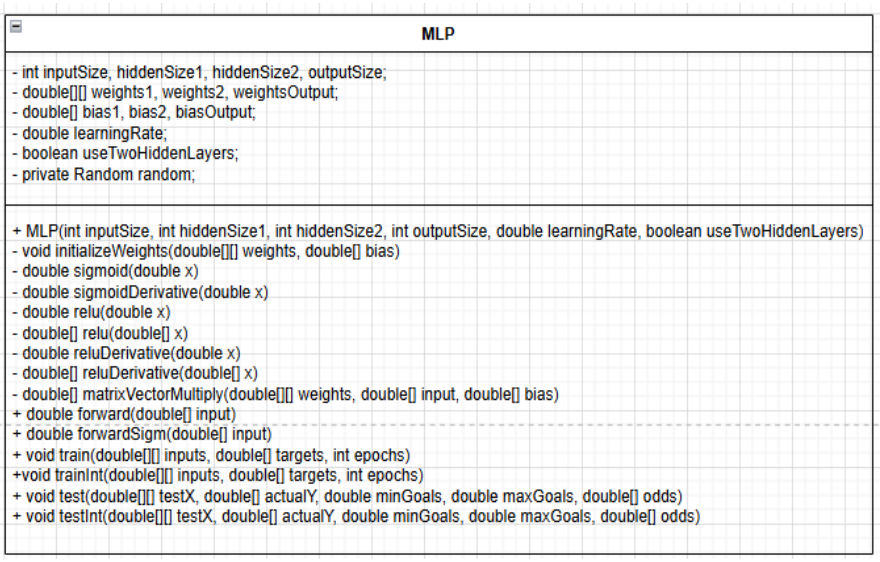
Αξιολόγηση (Testing)

- public void test(...): Αξιολογεί το δίκτυο σε σετ δεδομένων δοκιμής για πρόβλεψη συνολικού αριθμού τερμάτων. Υπολογίζει:
    - MSE (μέσο τετραγωνικό σφάλμα)
    - Απόδοση πρόβλεψης
    - ROI (Return on Investment)
    - Καταγράφει τα αποτελέσματα σε αρχείο .txt.
- public void testInt(...): Αντίστοιχα. Επιστρέφει ακρίβεια (accuracy) και ROI.

Συναρτήσεις ενεργοποίησης:

- sigmoid(x) και sigmoidDerivative(x): Χρησιμοποιούνται στην έξοδο για δυαδική ταξινόμηση.
- relu(x) και reluDerivative(x): Χρησιμοποιούνται στα κρυφά επίπεδα για αποδοτικότερη εκπαίδευση.
- matrixVectorMultiply(double[][] matrix, double[] vector): Χρησιμοποιείται εκτενώς στο forward και backward pass.
- Gradient descent update με learning rate: Τα βάρη και τα biases ενημερώνονται μέσω της εξίσωσης

Η κλάση MLP είναι μια πλήρης, εύχρηστη και παραμετροποιήσιμη υλοποίηση ενός νευρωνικού δικτύου για regression και classification. Η διαχείριση των βαρών, η ρύθμιση των επιπέδων και οι διαδικασίες εκπαίδευσης/δοκιμής επιτρέπουν την αξιοποίηση της σε εφαρμογές πρόβλεψης ποδοσφαιρικών αποτελεσμάτων.



Εικόνα 3.11: UML που δίχνει τα πεδία και τις μεθόδους της MLP

Κλάση RBFNetwork

Η κλάση RBFNetwork(Παράρτημα Γ) αποτελεί την κεντρική μονάδα υλοποίησης ενός τεχνητού νευρωνικού δικτύου τύπου Radial Basis Function (RBF), το οποίο χρησιμοποιείται στην παρούσα εργασία για την πρόβλεψη του αριθμού των τερμάτων σε ποδοσφαιρικούς αγώνες και την κατηγοριοποίηση των αποτελεσμάτων ως Over ή Under 2.5. Το δίκτυο αυτό ανήκει στην κατηγορία των εποπτευόμενων μοντέλων μάθησης, γεγονός που σημαίνει πως κατά τη φάση της εκπαίδευσης του δικτύου, χρησιμοποιούνται γνωστές είσοδοι και οι αντίστοιχες επιθυμητές έξοδοι. Ο στόχος του είναι να μάθει να "γενικεύει" ώστε να δίνει σωστά αποτελέσματα και σε εισόδους που δεν έχει "δει" στο παρελθόν.

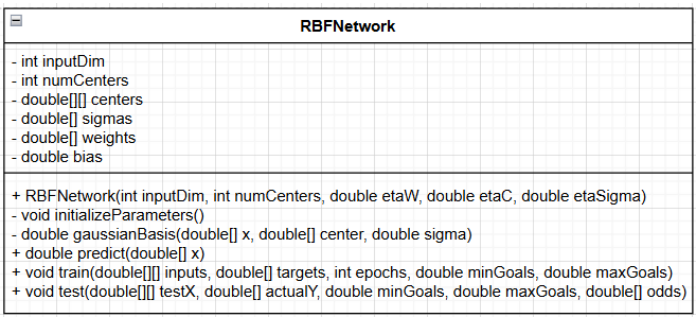
Περιγραφή πεδίων (fields):

- inputDim: Ορίζει τη διάσταση του διανύσματος εισόδου. Στην πράξη, αντιστοιχεί στον αριθμό των στατιστικών χαρακτηριστικών που περιγράφουν κάθε ομάδα πριν τον αγώνα, όπως μέσος όρος τερμάτων, ποσοστά Over/Under κλπ.
- numCenters: Αντιπροσωπεύει τον αριθμό των RBF κόμβων (ή αλλιώς "κέντρα"), δηλαδή τις μονάδες του κρυφού στρώματος. Κάθε τέτοιος κόμβος ενεργοποιείται με βάση την απόσταση μεταξύ της εισόδου και του αντίστοιχου κέντρου.
- centers: Είναι ένας πίνακας δύο διαστάσεων που περιέχει τις συντεταγμένες των κέντρων στον πολυδιάστατο χώρο των εισόδων. Οι τιμές αυτές μπορούν είτε να οριστούν χειροκίνητα είτε να μάθονται δυναμικά κατά την εκπαίδευση.
- sigmas: Περιέχει τις διασπορές (standard deviation) για κάθε κέντρο. Οι τιμές αυτές καθορίζουν το πόσο "ανοιχτή" είναι η Gaussian καμπύλη που αντιστοιχεί σε κάθε RBF κόμβο και επηρεάζουν την ευαισθησία κάθε κέντρου στις αποστάσεις από τα διανύσματα εισόδου.
- weights: Ο πίνακας των βαρών που συνδέει τους RBF κόμβους με τη μονάδα εξόδου. Η έξοδος του δικτύου είναι ένα γραμμικό άθροισμα των ενεργοποιήσεων των RBF κόμβων πολλαπλασιασμένων με τα αντίστοιχα βάρη.
- bias: Το bias (ή όρος μετατόπισης) προστίθεται στο τελικό άθροισμα ενεργοποιήσεων για να επιτρέψει στο δίκτυο να προσαρμόζεται καλύτερα σε δεδομένα που δεν έχουν συμμετρία γύρω από το μηδέν.
- etaW, etaC, etaSigma: Τρεις διαφορετικοί ρυθμοί μάθησης (learning rates) για τα βάρη, τα κέντρα και τις διασπορές αντίστοιχα. Ορίζουν πόσο μεγάλο θα είναι το κάθε βήμα ενημέρωσης των παραμέτρων κατά την εκπαίδευση.

Περιγραφή βασικών μεθόδων:

- initializeParameters(): Εκτελεί την αρχικοποίηση των παραμέτρων του δικτύου, δηλαδή των κέντρων, των βαρών και των διασπορών. Οι τιμές δίνονται τυχαία για τα κέντρα και τα βάρη (μικρές τιμές), ενώ οι διασπορές αρχικοποιούνται τυπικά με 1.
- gaussianBasis(double[] x, double[] c, double sigma): Υπολογίζει τη Gaussian συνάρτηση βάσης για ένα συγκεκριμένο RBF κόμβο. Η τιμή αυτή εξαρτάται από την ευκλείδεια απόσταση ανάμεσα στο διάνυσμα εισόδου και το κέντρο, καθώς και από τη διασπορά.
- predict(double[] input): Υπολογίζει την πρόβλεψη του δικτύου για μια συγκεκριμένη είσοδο. Εφαρμόζει Gaussian ενεργοποίηση σε κάθε κέντρο και υπολογίζει το τελικό άθροισμα ενεργοποιήσεων πολλαπλασιασμένων με τα αντίστοιχα βάρη.
- train(...): Η βασική μέθοδος εκπαίδευσης. Εκτελεί πολλές επαναλήψεις (epochs) και για κάθε δείγμα υπολογίζει την πρόβλεψη, το σφάλμα, και στη συνέχεια προχωρά στην ενημέρωση των βαρών, των κέντρων και των διασπορών χρησιμοποιώντας παραγώγους. Επιπλέον, καταγράφει τα αποτελέσματα σε αρχείο και υπολογίζει πόσο καλά προβλέπει η ταξινόμηση σε Over/Under.
- test(...): Εκτελεί δοκιμή του εκπαιδευμένου μοντέλου πάνω σε νέο σύνολο δεδομένων. Υπολογίζει metrics όπως το μέσο τετραγωνικό σφάλμα (MSE), ακρίβεια πρόβλεψης Over/Under, καθώς και οικονομικά στοιχεία όπως το ROI και το καθαρό κέρδος.

Η κλάση RBFNetwork συνδυάζει όλα τα απαραίτητα δομικά και λειτουργικά στοιχεία για την υλοποίηση ενός ικανού μοντέλου πρόβλεψης/ταξινόμησης βασισμένου σε Gaussian βάσεις. Είναι παραμετροποιήσιμη, επεκτάσιμη και εφαρμόζει απλές αλλά αποτελεσματικές μεθόδους εκπαίδευσης. Η ακρίβεια του δικτύου εξαρτάται άμεσα από την επιλογή των παραμέτρων (ρυθμοί μάθησης, αριθμός κέντρων, αρχικοποιήσεις), αλλά και από τη σωστή επιλογή στατιστικών χαρακτηριστικών στα δεδομένα εισόδου. Με σωστή ρύθμιση, μπορεί να λειτουργήσει αποτελεσματικά σε προβλήματα με δύσκολη καμπυλότητα και μη γραμμική φύση, όπως είναι η πρόβλεψη αθροισμάτων γκολ σε ποδοσφαιρικούς αγώνες.



Εικόνα 3.12: UML που δίχνει τα πεδία και τις μεθόδους της RBFNetwork

Κλάση HFONetwork

Η κλάση HFONetwork αποτελεί το κεντρικό στοιχείο της υλοποίησης ενός deep learning συστήματος βασισμένου στη μέθοδο Hessian-Free Optimization (HFO). Είναι σχεδιασμένη με γνώμονα την αντικειμενοστραφή προσέγγιση και υλοποιεί πλήρως ένα fully connected feedforward νευρωνικό δίκτυο με δύο κρυφά επίπεδα και ένα επίπεδο εξόδου. Η κλάση υποστηρίζει προβλήματα πρόβλεψης συνεχής τιμής ενώ η **κλάση HFONetworkSigm** περιέχει τις ίδιες λειτουργίες αλλά υποστηρίζει προβλήματα δυαδικής κατηγοριοποίησης (Binary Classification), γεγονός που την καθιστά ιδανική για προβλέψεις όπως αυτή του αριθμού των τερμάτων ή του αν θα ξεπεραστούν τα 2.5 γκολ σε έναν ποδοσφαιρικό αγώνα.

Σκοπός της Κλάσης

Ο βασικός ρόλος της HFONetwork είναι να παρέχει όλα τα απαραίτητα εργαλεία για την εκπαίδευση και αξιολόγηση ενός νευρωνικού δικτύου μέσω second-order μεθόδων, αποφεύγοντας τους περιορισμούς των κλασικών first-order τεχνικών όπως το Gradient Descent. Η μέθοδος HFO παρέχει πιο ακριβή και γρήγορη σύγκλιση αξιοποιώντας τοπικές παραβολικές προσεγγίσεις της συνάρτησης σφάλματος.

Περιγραφή πεδίων (Fields)

- inputSize, hiddenSize1, hiddenSize2, outputSize: Ορίζουν το μέγεθος κάθε στρώματος στο δίκτυο. Είναι ακέραιες μεταβλητές που καθορίζονται κατά την αρχικοποίηση του δικτύου.
- W1, W2, W3: Μήτρες βαρών που συνδέουν τα επίπεδα μεταξύ τους (Input → Hidden1, Hidden1 → Hidden2, Hidden2 → Output).
- b1, b2, b3: Διανύσματα bias για κάθε επίπεδο αντίστοιχα.
- rand: Αντικείμενο τύπου Random που χρησιμοποιείται για την τυχαία αρχικοποίηση των βαρών.

Λειτουργικότητα και Μέθοδοι

1. Προώθηση δεδομένων (Forward Propagation)

- forward(double[] input): Πραγματοποιεί πλήρη προώθηση του διανύσματος εισόδου μέσω των επιπέδων και επιστρέφει την έξοδο του δικτύου.

Χρησιμοποιείται για prediction. Οι ενεργοποιήσεις γίνονται με ReLU για τα κρυφά επίπεδα και linear για το output.

2. Εκπαίδευση και Βελτιστοποίηση (Training & Optimization)

Υπολογισμός Σφάλματος

- computeObjective(...): Υπολογίζει την τιμή της συνάρτησης κόστους (συνήθως MSE), βασισμένη στις εξόδους του δικτύου και τα πραγματικά δεδομένα.

Υπολογισμός Παραγώγων

- computeGradient(...): Υλοποιεί backpropagation για την εξαγωγή των παραγώγων των βαρών σε σχέση με το σφάλμα.

Hessian-Vector Product (Hessian-Free Μέθοδοι)

- hessianVectorProduct(...): Υπολογίζει το γινόμενο H(w)v χρησιμοποιώντας τον R-operator (Pearlmutter). Είναι ο πιο αξιόπιστος τρόπος για νευρωνικά δίκτυα με πολλές παραμέτρους.
- hessianVectorProductF(...): Εναλλακτική μέθοδος βασισμένη σε finite differences, με μικρότερο κόστος αλλά μειωμένη ακρίβεια.
- hessianVectorProductR(...): Βασική υλοποίηση του forward-over-backward pass που υποστηρίζει τις second-order παραγώγους.

R-Operator (Directional Derivatives)

- Το δίκτυο υποστηρίζει τον υπολογισμό της κατεύθυνσης βελτιστοποίησης χωρίς να απαιτείται ο πλήρης πίνακας Hessian, καθιστώντας την εκπαίδευση γρήγορη και αποτελεσματική ακόμη και για δίκτυα με πολλές παραμέτρους.

3. Χειρισμός Βαρών

- getWeights(), setWeights(...): Παρέχουν τρόπο για ανάκτηση και ορισμό των βαρών του δικτύου ως μονοδιάστατο διάνυσμα.
- updateWeights(...): Προσθέτει κάποια μεταβολή στα υπάρχοντα βάρη (συνήθως από κάποιον βελτιστοποιητή).
- flattenParams(...): Επίπεδοποιεί τις μήτρες βαρών σε ένα ενιαίο διάνυσμα.
- unflattenPerturbation(...): Μετατρέπει ένα επίπεδο διάνυσμα μεταβολής σε πίνακες κατάλληλους για ενημέρωση των W1, W2, W3 κ.λπ.

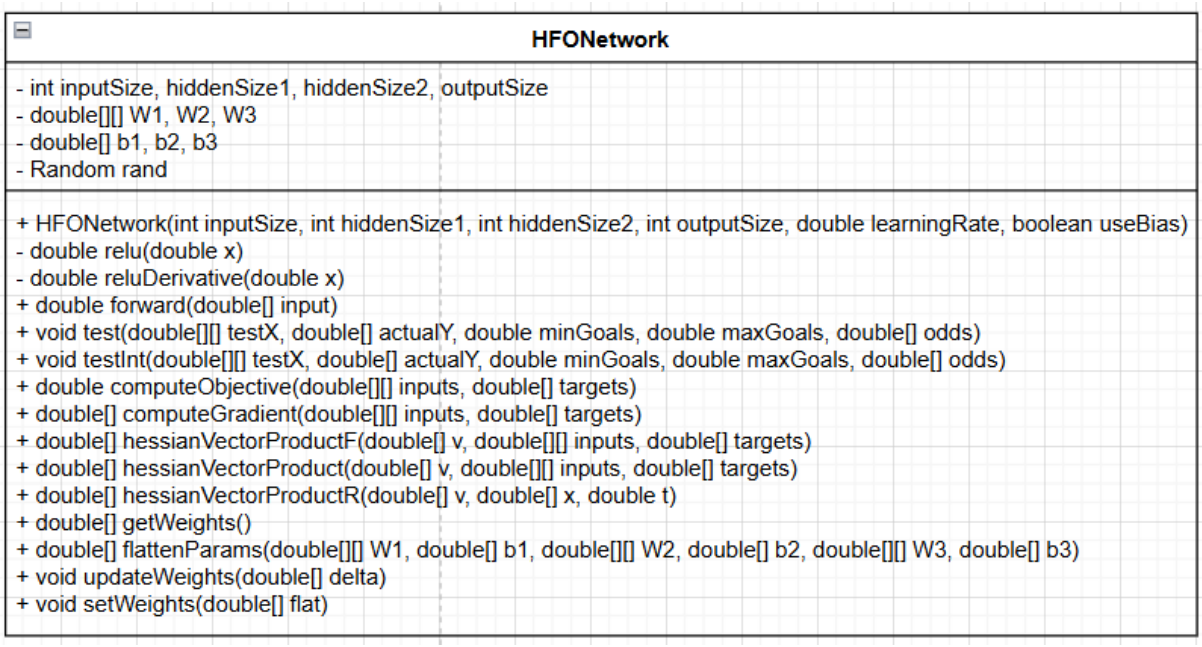
4. Αξιολόγηση & Εξαγωγή Απόδοσης

- test(...): Χρησιμοποιείται για προβλέψεις σε regression προβλήματα. Αξιολογεί το μοντέλο μετρώντας MSE, ROI, accuracy και αποθηκεύει τα αποτελέσματα σε αρχείο.

Εσωτερική Κλάση – ParamPerturbation

Η ParamPerturbation είναι βοηθητική εσωτερική κλάση που αποθηκεύει διαφοροποιήσεις στα βάρη για τις ανάγκες του HFO και της παραγωγής directional derivatives.

Η HFONetwork αποτελεί μια πλήρη υλοποίηση προηγμένου νευρωνικού μοντέλου που ενσωματώνει τεχνικές δεύτερης τάξης, κατάλληλες για προβλήματα με υψηλό αριθμό παραμέτρων και μεγάλο βάθος. Η υλοποίησή της είναι προσεκτικά σχεδιασμένη ώστε να είναι επεκτάσιμη, να συνεργάζεται με άλλες κλάσεις (όπως HFOTrainer, Team, Match) και να προσφέρει ακρίβεια και αποδοτικότητα σε πραγματικά δεδομένα.



Εικόνα 3.13: UML που δίχνει τα πεδία και τις μεθόδους της HFONetwork

Κλάση HFOTrainer

Η κλάση HFOTrainer αποτελεί το κεντρικό σημείο υλοποίησης του μηχανισμού εκπαίδευσης ενός νευρωνικού δικτύου μέσω της μεθόδου Hessian-Free Optimization (HFO) και υποστιρίζει εκπαίδευση για προβλήματα πρόβλεψης συνεχής τιμής ενώ η **κλάση HFOTrainerSigm** υποστιρίζει εκπαίδευση για προβλήματα δυαδικής κατηγοριοποίησης..

Περιγραφή πεδίων (Fields)

- HFONetwork network: Το υπό εκπαίδευση νευρωνικό δίκτυο, που υποστηρίζει HFO.
- double lambda: Η αρχική τιμή απόσβεσης (damping parameter), που χρησιμοποιείται στο πλαίσιο της τεχνικής Levenberg-Marquardt για να εξισορροπεί τη βελτιστοποίηση σε περιοχές χαμηλής αξιοπιστίας.
- int maxCGIterations: Ο μέγιστος αριθμός επαναλήψεων για την Conjugate Gradient (CG) μέθοδο, η οποία χρησιμοποιείται για την εύρεση της διεύθυνσης ενημέρωσης βαρών.
- double tolerance: Το όριο ανοχής για τη σύγκλιση της CG μεθόδου – μόλις η πρόοδος είναι μικρότερη από αυτήν την τιμή, η επανάληψη σταματά.

Κύρια Μέθοδος: train(double[][] inputs, double[][] targets, int epochs)

- Η μέθοδος αυτή εκτελεί την πλήρη διαδικασία εκπαίδευσης του νευρωνικού δικτύου για προκαθορισμένο αριθμό εποχών. Για κάθε εποχή:
- Υπολογίζεται το σφάλμα (objective function) του δικτύου για τα δεδομένα εισόδου.
- Υπολογίζεται το διάνυσμα παραγώγου (gradient) ως προς τα βάρη του δικτύου.
- Υπολογίζεται η καμπυλότητα του σφάλματος (χωρίς να δημιουργηθεί ολόκληρος ο πίνακας Hessian), με χρήση παραγώγων κατεύθυνσης.
- Εφαρμόζεται ο αλγόριθμος Συζυγών Κατευθύνσεων (CG) ή η παραλλαγή του με προϋπόθεση (preconditioned CG), για εύρεση της βέλτιστης κατεύθυνσης ενημέρωσης βαρών δ*.
- Γίνεται χρήση της γραμμικής αναζήτησης (line search) για εύρεση του κατάλληλου μεγέθους βήματος (step size).
- Ανανεώνονται τα βάρη του δικτύου.

- Προσαρμόζεται δυναμικά η τιμή της απόσβεσης λ μέσω της Levenberg-Marquardt heuristic, ώστε να διατηρηθεί σταθερό το "trust region" του παραβολικού μοντέλου.

Υποστηρικτικές Μέθοδοι:

- conjugateGradient(...): Κλασική μέθοδος CG χωρίς προϋπόθεση, για επίλυση συστημάτων Ax = b.
- conjugateGradientPrecond(...): Παραλλαγή της CG που χρησιμοποιεί preconditioner, βελτιώνοντας την ταχύτητα σύγκλισης.
- computePreconditioner(): Εκτιμά την προϋπόθεση του συστήματος (διαγώνιο πίνακα) με τη μέθοδο Hutchinson, για πιο σταθερό και αποδοτικό CG.
- lineSearch(...): Υλοποιεί γραμμική αναζήτηση με βάση το κριτήριο Armijo, για να βρεθεί ασφαλές και αποτελεσματικό μήκος βήματος.
- modelReduction(...): Εκτιμά τη θεωρητική μείωση του σφάλματος βάσει της παραβολικής προσέγγισης.
- adjustLambda(...): Τροποποιεί την τιμή του damping (λ) βάσει της αναλογίας μείωσης (reduction ratio), ώστε να υπάρχει ισορροπία μεταξύ μεγάλων και μικρών βημάτων.
- dot(...): Υπολογίζει το εσωτερικό γινόμενο δύο διανυσμάτων – χρησιμοποιείται σε CG και άλλες μεθόδους.

Η κλάση `HFOTrainer` λειτουργεί σε στενή συνεργασία με την κλάση `HFONetwork`, καθώς οι δύο αυτές κλάσεις αποτελούν τα δύο βασικά μέρη του συστήματος εκπαίδευσης μέσω της μεθόδου Hessian-Free Optimization (HFO). Ο ρόλος τους είναι διακριτός αλλά άρρηκτα συνδεδεμένος: η μία (HFONetwork) διαχειρίζεται τη δομή και τα δεδομένα του νευρωνικού δικτύου, ενώ η άλλη (HFOTrainer) υλοποιεί τη διαδικασία εκπαίδευσης με βάση τα χαρακτηριστικά του δικτύου.

Σχέση μεταξύ HFOTrainer και HFONetwork

Η κλάση HFONetwork είναι υπεύθυνη για τη δομή και τη λειτουργία του ίδιου του δικτύου:

- Καθορίζει τη διάταξη των στρωμάτων (layers), τους κόμβους (neurons), και τις συνδέσεις (weights).
- Περιλαμβάνει τις μεθόδους προώθησης της εισόδου (forward pass), υπολογισμού του σφάλματος, εξόδου του δικτύου και άλλων βοηθητικών μεθόδων.
- Διαθέτει τις παραμέτρους (βάρη) του δικτύου και τις μεθόδους για την ενημέρωσή τους.

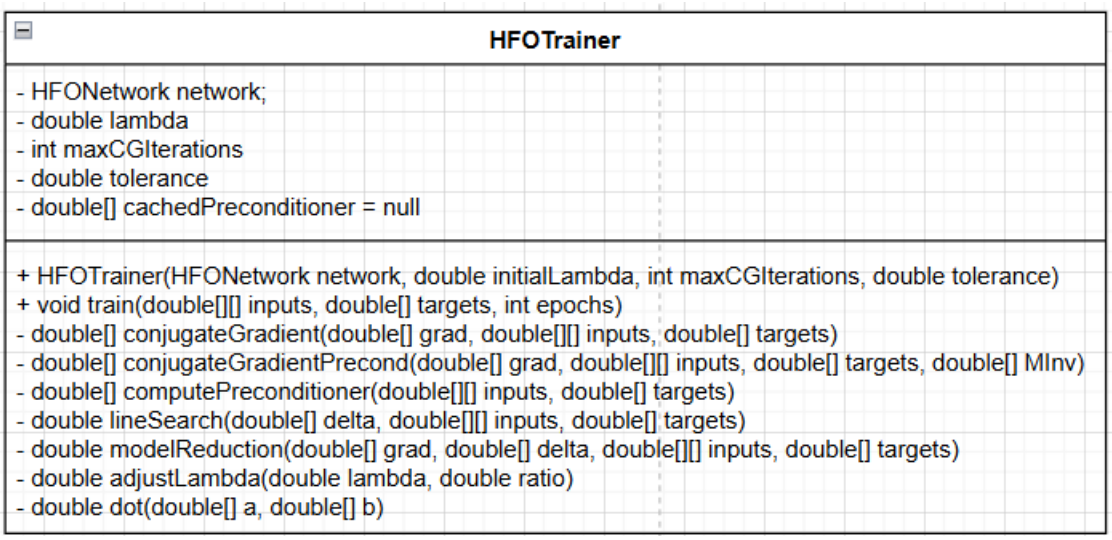
Η κλάση HFOTrainer είναι εκείνη που εκτελεί τον αλγόριθμο εκπαίδευσης, αξιοποιώντας τις μεθόδους του HFONetwork. Αναλαμβάνει:

- Να υπολογίσει το σφάλμα και τις παραγώγους (gradient) του δικτύου.
- Να εκτιμήσει την καμπυλότητα του σφάλματος μέσω των Hu ή Gu προϊόντων.
- Να εφαρμόσει την Conjugate Gradient (CG) μέθοδο για την εύρεση κατεύθυνσης ενημέρωσης.
- Να προσαρμόσει το μήκος βήματος μέσω γραμμικής αναζήτησης.
- Να τροποποιήσει την τιμή του damping factor (λ) σύμφωνα με τη Levenberg-Marquardt τεχνική.
- Και τέλος, να καλέσει την updateWeights() του HFONetwork, ενημερώνοντας τελικά τα βάρη.

Πρακτικά:

- Το HFONetwork είναι το νευρωνικό δίκτυο που «κρατάει» τις παραμέτρους.
- Το HFOTrainer είναι το εργαλείο που «εκπαιδεύει» αυτό το δίκτυο, τροποποιώντας τις παραμέτρους του μέσω μαθηματικών μεθόδων δεύτερης τάξης.
- Η δομή αυτή εξασφαλίζει σαφή διαχωρισμό ευθυνών: η μία κλάση ορίζει τι είναι το δίκτυο, ενώ η άλλη πώς εκπαιδεύεται. Έτσι, επιτυγχάνεται αρθρωτός και επεκτάσιμος σχεδιασμός, κατάλληλος για σύνθετες εφαρμογές μηχανικής μάθησης.

Η HFOTrainer είναι η βασική συνιστώσα για την εκπαίδευση του HFONetwork, παρέχοντας τη λογική εκπαίδευσης με Hessian-Free Optimization, χωρίς την ανάγκη για πλήρη υπολογισμό του Hessian Matrix. Συνδυάζει την ισχύ των μεθόδων δεύτερης τάξης με αποδοτικές αριθμητικές τεχνικές (CG, line search, preconditioning), καθιστώντας την ιδανική για σύνθετες εφαρμογές όπως η πρόβλεψη αποτελεσμάτων αγώνων, όπου απαιτείται υψηλή ακρίβεια και ταχύτητα εκπαίδευσης.



Εικόνα 3.14: UML που δίχνει τα πεδία και τις μεθόδους της HFOTrainer

Κλάση Main

Η κλάση Main αποτελεί το βασικό σημείο εκκίνησης (entry point) της εφαρμογής και είναι υπεύθυνη για την πλήρη διαχείριση της ροής της διαδικασίας: από την ανάγνωση των αρχείων δεδομένων, τη δημιουργία των στατιστικών κάθε ομάδας για κάθε αγωνιστική, έως την προετοιμασία των δεδομένων εκπαίδευσης και τη δοκιμή διαφορετικών τύπων τεχνητών νευρωνικών δικτύων.

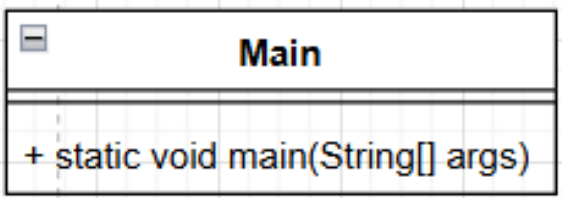
Ανάλυση λειτουργίας:

1. Προετοιμασία φακέλων και επεξεργασία CSV: Η κλάση ξεκινά καθορίζοντας τους φακέλους εισόδου και εξόδου (inputDir, outputDir). Δημιουργεί τον φάκελο εξόδου αν δεν υπάρχει, και έπειτα, μέσω της κλάσης FileEditor, γίνεται η επεξεργασία των αρχείων CSV για κάθε σεζόν, μετατρέποντάς τα σε μορφή κατάλληλη για ανάλυση, αποθηκευμένα ως .txt.
2. Δημιουργία αντικειμένων Season και υπολογισμός στατιστικών: Για κάθε σεζόν, δημιουργείται αντικείμενο τύπου Season, το οποίο αναπαριστά όλα τα παιχνίδια της χρονιάς και περιλαμβάνει τις ομάδες, τις αποδόσεις, τα αποτελέσματα και τα στατιστικά κάθε ομάδας ανά αγωνιστική. Η μέθοδος gamesPlayed(46) ενεργοποιεί την παραγωγή των στατιστικών για 46 αγωνιστικές (τυπικός αριθμός για Championship).
3. Δημιουργία εκπαιδευτικών και δοκιμαστικών δεδομένων: Οι σεζόν από το 2006 έως το 2023 χρησιμοποιούνται για εκπαίδευση, εξάγοντας πίνακες χαρακτηριστικών (34 διαστάσεις) και επιθυμητές εξόδους για τον αριθμό τερμάτων και την πιθανότητα Over. Τα δεδομένα της σεζόν 2024 χρησιμοποιούνται ως test set.
4. Εκπαίδευση και δοκιμή MLP δικτύων: Δημιουργούνται δύο MLP μοντέλα:
   - Το πρώτο εκπαιδεύεται για πρόβλεψη αριθμού τερμάτων (regression).
   - Το δεύτερο για κατηγοριοποίηση Over/Under (classification). Και τα δύο δίκτυα εκπαιδεύονται με χρήση μεθόδου backpropagation και εφαρμόζουν normalization στα δεδομένα εξόδου για σωστή σύγκριση.
5. Εκπαίδευση και δοκιμή RBF δικτύου: Δημιουργείται ένα Radial Basis Function (RBF) δίκτυο, με καθορισμένα learning rates για τα βάρη, τα κέντρα και τις αποκλίσεις των Gaussians. Εκπαιδεύεται και δοκιμάζεται όπως τα MLP.
6. Εκπαίδευση και δοκιμή HFO δικτύων: Δύο μοντέλα HFO (Hessian-Free Optimization) χρησιμοποιούνται:
   - Το πρώτο (HFONetwork) για αριθμό τερμάτων.
   - Το δεύτερο (HFONetworkSigm) για πιθανότητα Over. Τα HFO δίκτυα κατασκευάζονται με χρήση δεύτερης τάξης βελτιστοποίησης, αξιοποιώντας τις κλάσεις HFOTrainer και HFOTrainerSigm, που αναλαμβάνουν τον έλεγχο

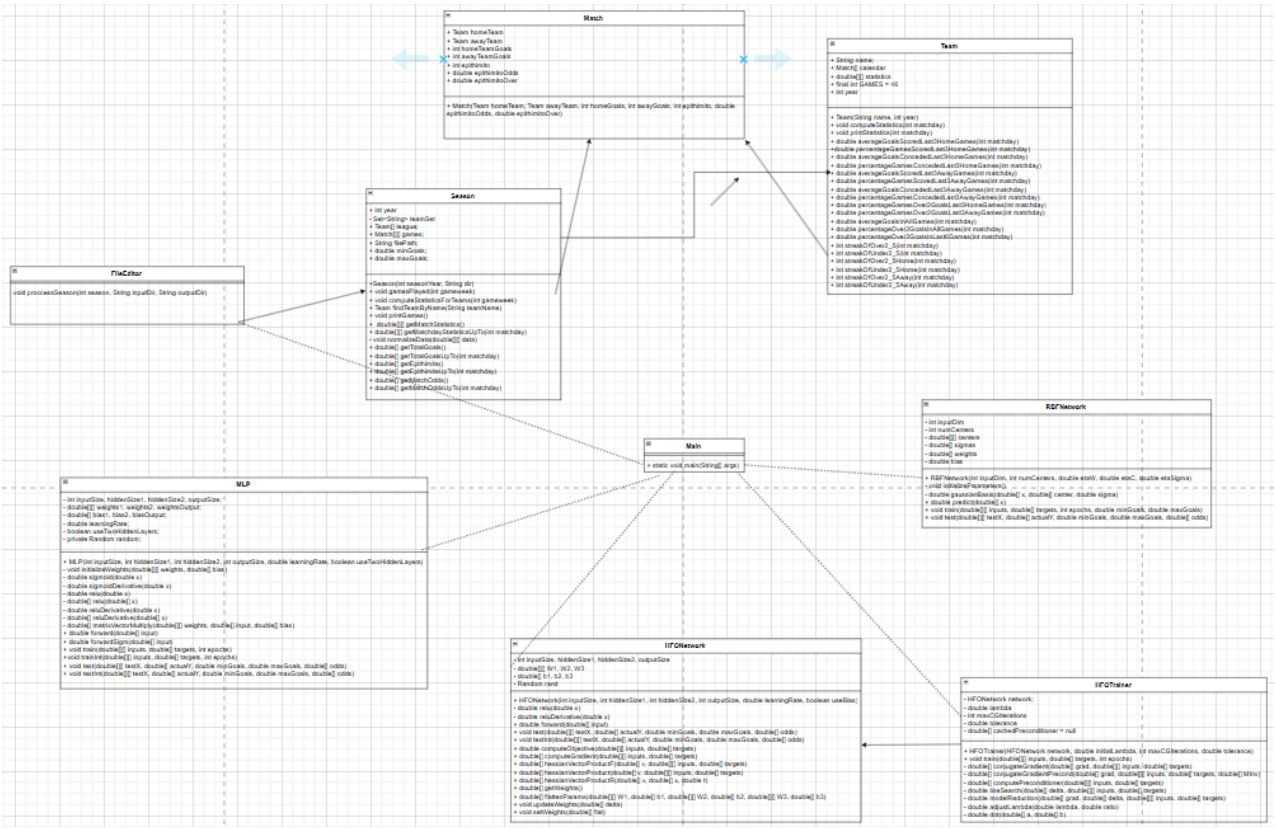
του τρόπου με τον οποίο ενημερώνονται τα βάρη μέσω εναλλακτικών αλγορίθμων όπως conjugate gradient και damping.

Συμπεράσματα:

Η Main λειτουργεί σαν ενορχηστρωτής που ενώνει όλα τα κομμάτια της εφαρμογής: διαχείριση δεδομένων, δημιουργία στατιστικών, προετοιμασία εκπαιδευτικών συνόλων και εκπαίδευση/δοκιμή διαφόρων αρχιτεκτονικών νευρωνικών δικτύων. Χρησιμοποιεί μια πολυμορφική προσέγγιση με multiple models (MLP, RBF, HFO), ενσωματώνοντας τόσο supervised regression όσο και classification προβλέψεις, παρέχοντας έτσι ολοκληρωμένα αποτελέσματα για το πρόβλημα της πρόβλεψης τερμάτων σε ποδοσφαιρικούς αγώνες.



Εικόνα 3.15: UML δίχνει τα πεδία και τις μεθόδους της Main



Εικόνα 3.16: UML Class diagram που παρουσιάζει όλες τις κλάσεις μαζί με τις σχέσεις τους

Κεφάλαιο 4

Αποτελέσματα και Συζήτηση

Σε αυτό το κεφάλαιο θα αναφέρω μερικά από τα πειράματα που έγιναν και μας δίνουν αποτελέσματα τα οποία επιδέχονται συζήτησης και μέσω αυτών μπορούν να εξαχθούν κάποια συμπεράσματα. Για τους αλγόριθμους MLP και HFO χρησιμοποίησα 2 προσεγγίσεις όπως ανάφερα και προηγούμενος, πρόβλεψη συνολικού αριθμού τερμάτων που θα επιτευχθούν (επιθυμητό αποτέλεσμα είναι ο αριθμός των τερμάτων που σημειώθηκαν σε έναν αγώνα) και κατηγοριοποίηση (επιθυμητό αποτέλεσμα είναι 1 αν στο αγώνα σημειωθήκαν περισσότερα από 2,5 τέρματα και 0 αν σημειώθηκαν λιγότερα από 2,5 τέρματα).

Για την αξιολόγηση των αποτελεσμάτων και την εξαγωγή συμπερασμάτων σχετικά με την αποδοτικότητα των μοντέλων, κρίθηκε απαραίτητο να υπολογιστούν τόσο η ακρίβεια πρόβλεψης όσο και η χρηματοοικονομική απόδοση της κάθε προσέγγισης. Για τον σκοπό αυτό, σε όλα τα πειράματα έγινε προσομοίωση πονταρίσματος σε στοιχηματικές αγορές βάσει των προβλέψεων των μοντέλων. Συγκεκριμένα, κάθε φορά που η πρόβλεψη ενός μοντέλου συμφωνούσε με το πραγματικό αποτέλεσμα (δηλαδή σωστή πρόβλεψη για over ή under 2.5 τέρματα), θεωρούνταν ότι επιτυγχανόταν ένα κέρδος ίσο με την απόδοση (odds) του στοιχήματος επί το ποντάρισμα (stake), το οποίο για λόγους απλοποίησης θεωρήθηκε σταθερό και ίσο με 1€. Αντίθετα, σε περίπτωση λανθασμένης πρόβλεψης, καταγραφόταν απώλεια ίση με το ποντάρισμα. Το συνολικό χρηματοοικονομικό κέρδος για κάθε πείραμα προέκυπτε από το άθροισμα των κερδών και των ζημιών σε όλη τη διάρκεια των αγώνων.

Παράλληλα, για την καλύτερη κατανόηση της απόδοσης των αλγορίθμων, υπολογίστηκε και το ποσοστό επιστροφής χρημάτων (ROI - Return on Investment). Ο δείκτης αυτός δείχνει πόσο αποδοτική ήταν συνολικά η στρατηγική πονταρίσματος με βάση τις προβλέψεις κάθε μοντέλου. Υπολογίστηκε ως εξής:

$$\text{ROI (\%)} = \left(\frac{\text{Συνολικό Κέρδος}}{\text{Συνολικό Ποντάρισμα}} \right) \times 100$$

Όπου το συνολικό ποντάρισμα ήταν το πλήθος των αγώνων επί το ποσό του κάθε στοιχήματος (δηλαδή το 1€ για κάθε αγώνα). Το ROI επιτρέπει την αξιολόγηση της πρακτικής αξίας των προβλέψεων κάθε μοντέλου, προσφέροντας μία ξεκάθαρη εικόνα για το κατά πόσο μια στρατηγική βασισμένη στις προβλέψεις του εκάστοτε αλγορίθμου θα μπορούσε να είναι επικερδής σε πραγματικές συνθήκες στοιχηματισμού.

4.1Αποτελέσματα MLP Backpropagation

Προσπάθεια 1

Στο πρώτο δίκτυο χρησιμοποίησα 26 νευρώνες εισόδου, 2 κρυφά επίπεδα όπου και τα δύο επίπεδα περιείχαν 10 κρυφούς νευρώνες και 1 νευρώνα εξόδου με 0,05 learning rate. Χρησιμοποίησα τα ίδια στατιστικά(26 σε αριθμό) για δεδομένα εισόδου που χρησιμοποιούσε και ο Φράγκος στην διπλωματική του εργασία. Δηλαδή τα ίδια στατιστικά που ανάφερα στο κεφάλαιο 3.1 εχτός αυτά που βρίσκονται στις θέσεις 23-28.

Νευρώνες εισόδους: 26

Κρυφά επίπεδα: 2

Νευρώνες 1ου κρυφού επιπέδου: 10

Νευρώνες 2ου κρυφού επιπέδου: 10

Νευρώνες εξόδου: 1

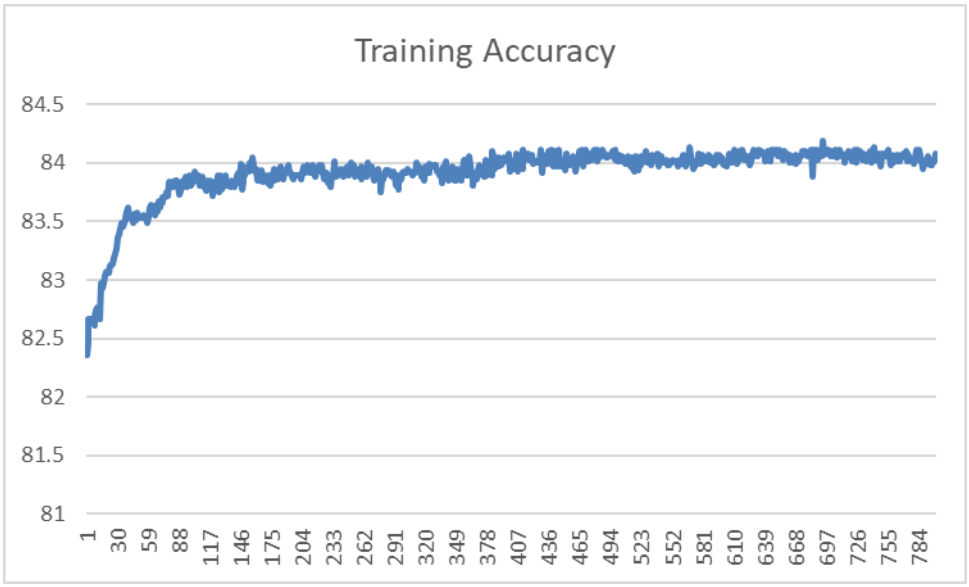
Ρυθμός μάθησης: 0.05

Χρονιές δεδομένων εκπαίδευση: 2009/10 - 2022/23

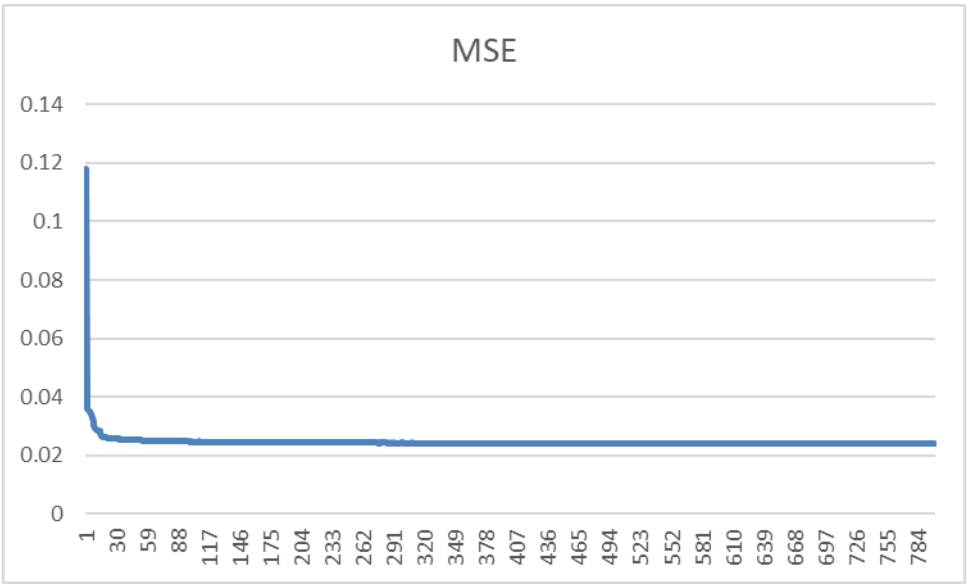
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 800

Αποτελέσματα Πρόβλεψης Τερμάτων (Επιθυμητή έξοδος: συνολικός αριθμός γκολ)



Εικόνα 4.1: Γραφική ακρίβειας Δεδομένων εκπαίδευσης MLP με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 1.

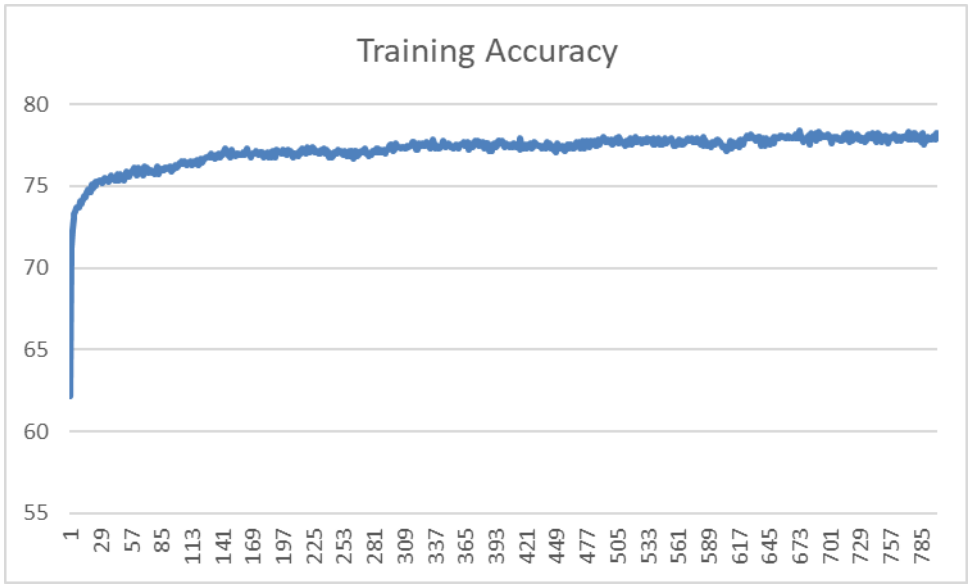


Εικόνα 4.2: Γραφική σφάλματος δεδομένων εκπαίδευσης MLP με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 1

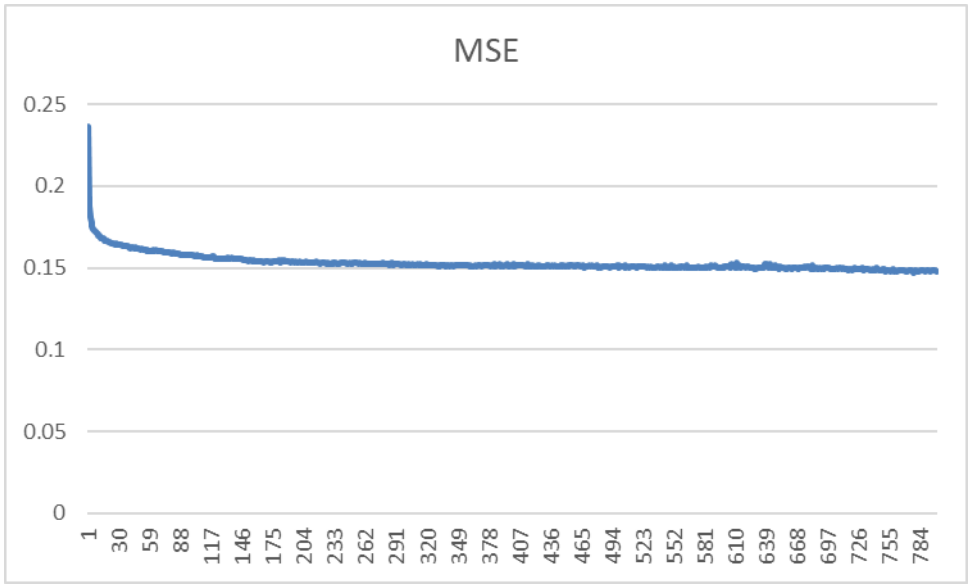
Ακρίβεια δεδομένων ελέγχου: 314/480 65.4%

Κέρδος: 413€ με επιστροφή χρημάτων στο 89%

Αποτελέσματα με Κατηγοριοποίηση(1 αν το επιθυμητό είναι over 2,5 και 0 under 2,5)



Εικόνα 4.3: Γραφική ακρίβειας Δεδομένων εκπαίδευσης MLP με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 1.



Εικόνα 4.4: Γραφική σφάλματος δεδομένων εκπαίδευσης MLP με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 1.

Ακρίβεια δεδομένων ελέγχου: 338/480 70,1%

Κέρδος: 522.8€ με επιστροφή χρημάτων στο 108.9%

Προσπάθεια 2

Σε αυτο το δίκτυο χρησιμοποίησα 34 νευρώνες εισόδου, 2 κρυφά επίπεδα όπου και τα δύο επίπεδα περιείχαν 10 κρυφούς νευρώνες και 1 νευρώνα εξόδου με 0,05 learning rate. Χρησιμοποίησα τα στατιστικά που ανάφερα στο κεφάλαιο 3.2. Στο κεφάλαιο 3.6 στο σημείο ανάλυσης της κλάσης season περιγράφεται εκτενώς τι είναι το διάνυσμα 34 θέσεων που δίνω σαν είσοδο στο δίκτυο.

Νευρώνες εισόδους: 34

Κρυφά επίπεδα: 2

Νευρώνες 1ου κρυφού επιπέδου: 10

Νευρώνες 2ου κρυφού επιπέδου: 10

Νευρώνες εξόδου: 1

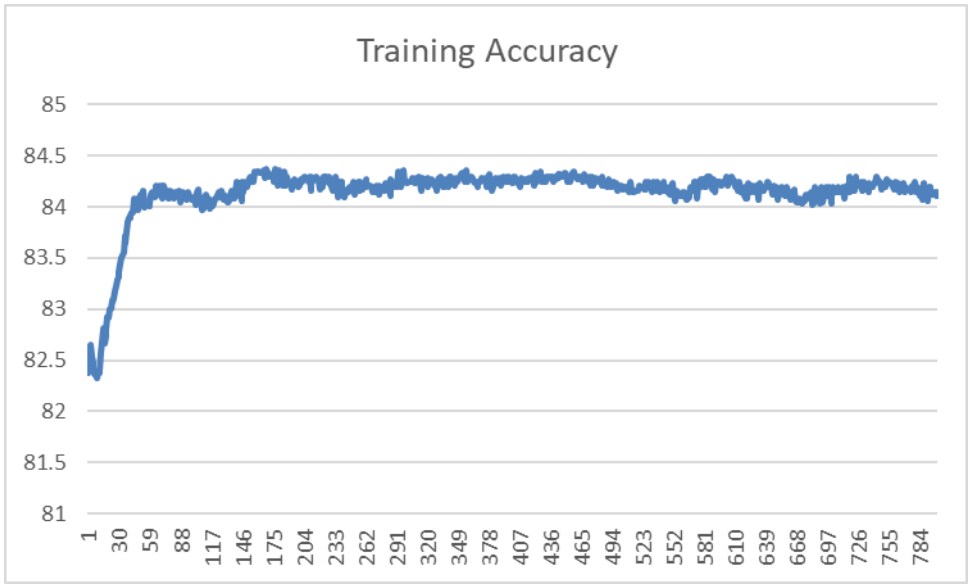
Ρυθμός μάθησης: 0.05

Χρονιές δεδομένων εκπαίδευση: 2009/10 - 2022/23

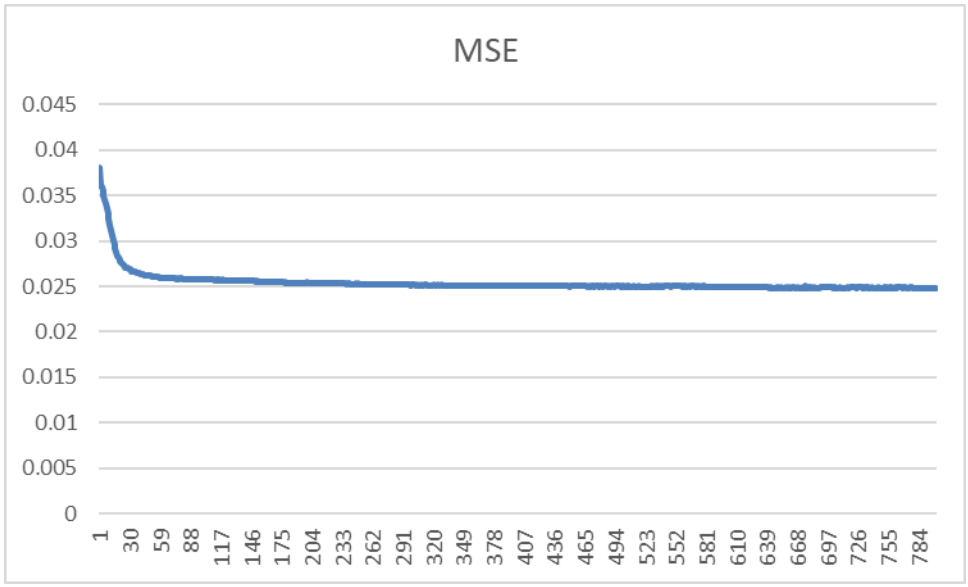
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 800

Αποτελέσματα Πρόβλεψης Τερμάτων (Επιθυμητή έξοδος: συνολικός αριθμός γκολ)



Εικόνα 4.5: Γραφική ακρίβειας Δεδομένων εκπαίδευσης MLP με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 2.

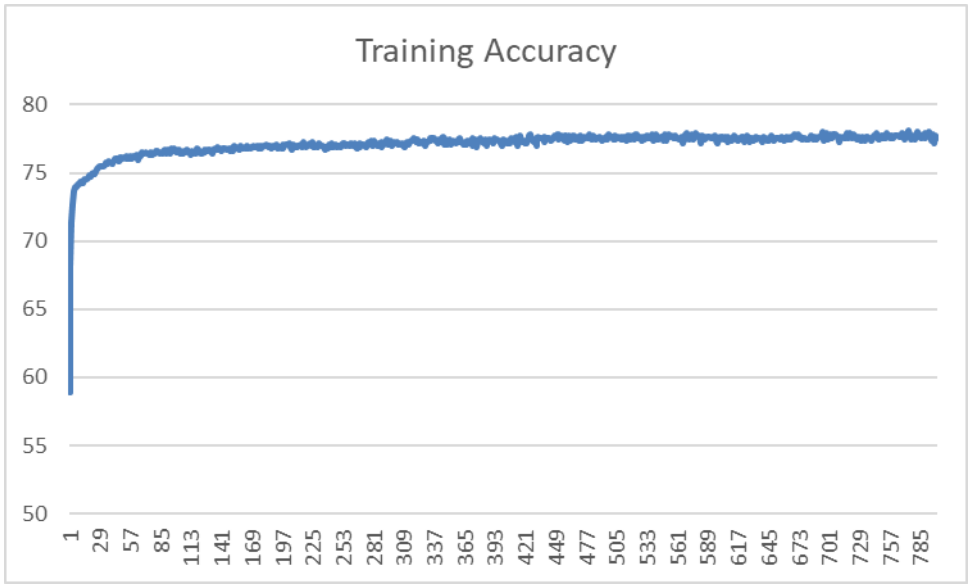


Εικόνα 4.6: Γραφική σφάλματος δεδομένων εκπαίδευσης MLP με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 2.

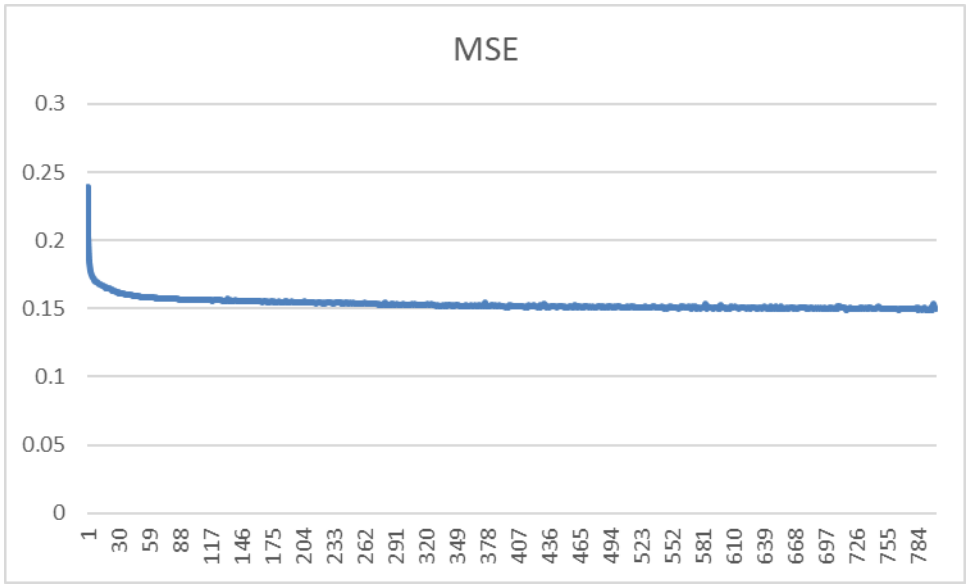
Ακρίβεια δεδομένων ελέγχου: 335/480 69.79%

Κέρδος: 487€ με επιστροφή χρημάτων στο 101.46%

Αποτελέσματα με Κατηγοριοποίηση(1 αν το επιθυμητό είναι over 2,5 και 0 under 2,5)



Εικόνα 4.7: Γραφική ακρίβειας Δεδομένων εκπαίδευσης MLP με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 2.



Εικόνα 4.8: Γραφική σφάλματος δεδομένων εκπαίδευσης MLP με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 2.

Ακρίβεια δεδομένων ελέγχου: 339/480 70.63%

Κέρδος: 502.8€ με επιστροφή χρημάτων στο 104.76%

Προσπάθεια 3

Αυτό το σύνολο δεδομένων μου πρόσφερε τα καλύτερα αποτελέσματα των MLP. Σε δίκτυο χρησιμοποίησα 34 νευρώνες εισόδου, 2 κρυφά επίπεδα όπου το ένα επίπεδο περιείχε 10 κρυφούς νευρώνες ενώ το δεύτερο 5 κρυφούς νευρώνες και 1 νευρώνα εξόδου με 0.04 learning rate. Χρησιμοποίησα τα στατιστικά που ανάφερα στο κεφάλαιο 3.2. Στο κεφάλαιο 3.6 στο σημείο ανάλυσης της κλάσης season περιγράφεται εκτενώς τι είναι το διάνυσμα 34 θέσεων που δίνω σαν είσοδο στο δίκτυο. Η διαφορά αυτού του συνόλου δεδομένου από το προηγούμενα είναι κυρίος ο αριθμός των νευρώνων στο 2ο κρυφό επίπεδο αλλά και το σύνολο των δεδομένων εκπαίδευσης.

Νευρώνες εισόδους: 34

Κρυφά επίπεδα: 2

Νευρώνες 1ου κρυφού επιπέδου: 10

Νευρώνες 2ου κρυφού επιπέδου: 5

Νευρώνες εξόδου: 1

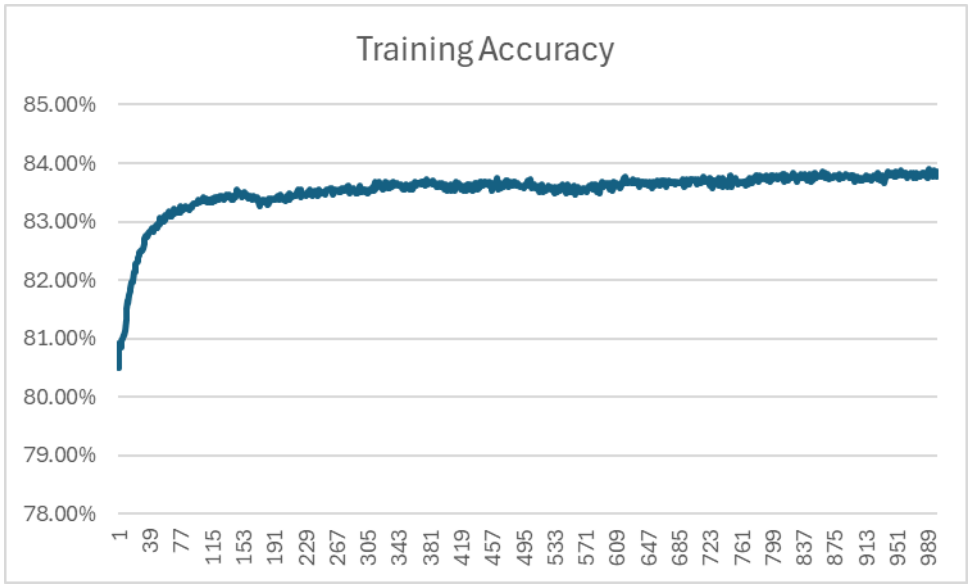
Ρυθμός μάθησης: 0.04

Χρονιές δεδομένων εκπαίδευση: 2005/06 - 2022/23

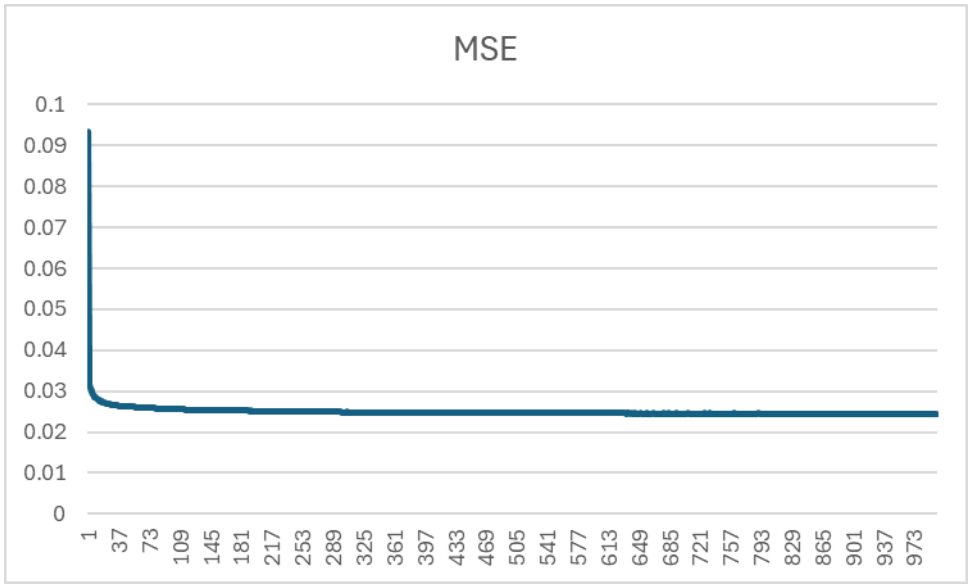
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 1000

Αποτελέσματα Πρόβλεψης Τερμάτων (Επιθυμητή έξοδος: συνολικός αριθμός γκολ)



Εικόνα 4.9: Γραφική ακρίβειας Δεδομένων εκπαίδευσης MLP με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 3.

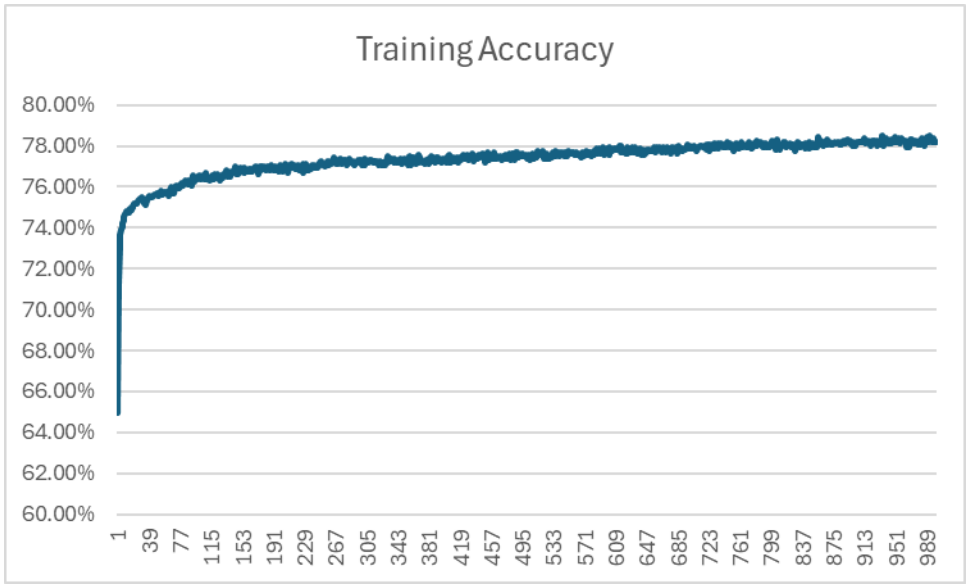


Εικόνα 4.10: Γραφική σφάλματος δεδομένων εκπαίδευσης MLP με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 3.

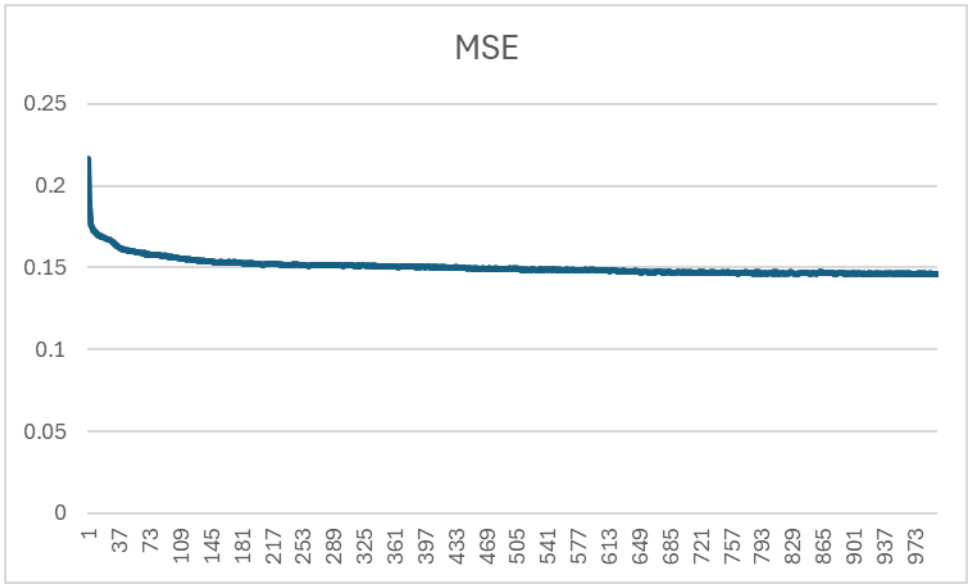
Ακρίβεια δεδομένων ελέγχου: 333/480 69.38%

Κέρδος: 482.1€ με επιστροφή χρημάτων στο 100.43%

Αποτελέσματα με Κατηγοριοποίηση(1 αν το επιθυμητό είναι over 2,5 και 0 under 2,5)



Εικόνα 4.11: Γραφική ακρίβειας Δεδομένων εκπαίδευσης MLP με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 3.



Εικόνα 4.12: Γραφική σφάλματος δεδομένων εκπαίδευσης MLP με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 3.

Ακρίβεια δεδομένων ελέγχου: 349/480 71.3%

Κέρδος: 519.1€ με επιστροφή χρημάτων στο 110.17%

Αρχικά, στην Προσπάθεια 1, χρησιμοποιήθηκε ένα δίκτυο με 26 χαρακτηριστικά εισόδου και δύο κρυφά επίπεδα των 10 νευρώνων το καθένα, με ρυθμό μάθησης 0.05. Το δίκτυο εκπαιδεύτηκε για 800 εποχές με δεδομένα από τις περιόδους 2009/10 έως 2022/23. Όσον αφορά την πρόβλεψη συνολικού αριθμού τερμάτων, καταγράφηκε ακρίβεια 65.4% στα δεδομένα ελέγχου (314/480), με συνολικό κέρδος €413 και απόδοση επένδυσης στο 86%. Αντίστοιχα, στην κατηγοριοποίηση, η ακρίβεια αυξήθηκε σε 70.1% (338/480) και το κέρδος ανήλθε στα €522.8 με απόδοση 108.9%. Η διαφορά αυτή οφείλεται στο ότι η κατηγοριοποίηση φαίνεται να είναι πιο αποτελεσματική σε όρους επιλογών over/under 2.5, καθώς βασίζεται σε ένα πιο απλό σήμα εξόδου (0 ή 1).

Στην Προσπάθεια 2, το δίκτυο επεκτάθηκε σε 34 νευρώνες εισόδου, κρατώντας τον ίδιο αριθμό νευρώνων στα κρυφά επίπεδα και την ίδια τιμή learning rate. Η εκπαίδευση έγινε και πάλι για 800 εποχές, με τα ίδια χρονικά διαστήματα δεδομένων. Στην πρόβλεψη συνολικού αριθμού τερμάτων, η ακρίβεια βελτιώθηκε ελαφρώς σε 69.79% (335/480) με κέρδος €487 και απόδοση 101.46%, ενώ στην κατηγοριοποίηση παρατηρήθηκε περαιτέρω αύξηση της ακρίβειας σε 70.63% (339/480) με κέρδος €502.8 και επιστροφή 104.76%. Η μικρή αυτή βελτίωση επιβεβαιώνει ότι η αύξηση των χαρακτηριστικών εισόδου βελτιώνει την επίδοση, χωρίς όμως να οδηγεί σε θεαματικές αλλαγές, αν δεν συνοδευτεί και από άλλες προσαρμογές στο δίκτυο.

Η Προσπάθεια 3 ήταν αυτή που απέφερε τα καλύτερα αποτελέσματα στον αλγόριθμο MLP. Διατηρήθηκε ο αριθμός εισόδων στους 34, όμως διαφοροποιήθηκε η αρχιτεκτονική των κρυφών επιπέδων, με 10 νευρώνες στο πρώτο και 5 στο δεύτερο επίπεδο, και μειώθηκε ο ρυθμός μάθησης σε 0.04. Επιπλέον, διευρύνθηκε σημαντικά το σύνολο εκπαίδευσης (2005/06 έως 2022/23), ενώ αυξήθηκαν και οι εποχές εκπαίδευσης σε 1.000. Αυτές οι αλλαγές οδήγησαν σε καλύτερη συγκλίνουσα συμπεριφορά και ενίσχυση της ακρίβειας. Συγκεκριμένα, στην πρόβλεψη συνολικού αριθμού τερμάτων η ακρίβεια διαμορφώθηκε σε 69.38% (333/480) με κέρδος €482.1 και απόδοση 100.43%, ενώ στην κατηγοριοποίηση σημειώθηκε η καλύτερη επίδοση συνολικά: 71.3% ακρίβεια (349/480) και κέρδος €519.1 με απόδοση 110.17%.

Από την ανάλυση των αποτελεσμάτων που προέκυψαν από τις διαφορετικές υλοποιήσεις του πολυεπίπεδου νευρωνικού δικτύου (MLP – Multilayer Perceptron), διαπιστώθηκαν σημαντικά ευρήματα που αναδεικνύουν τόσο τη δυναμική των δικτύων

αυτών όσο και τις κρίσιμες παραμέτρους που επηρεάζουν την απόδοσή τους. Αξίζει να σημειωθεί ότι τα μοντέλα που εκπαιδεύτηκαν για πρόβλεψη συνολικού αριθμού τερμάτων παρουσίασαν ελαφρώς υψηλότερη ακρίβεια στα δεδομένα εκπαίδευσης σε σύγκριση με τα μοντέλα κατηγοριοποίησης. Αυτό υποδηλώνει ότι τα μοντέλα πρόβεψης συνολικού αριθμού τερμάτων είχαν μεγαλύτερη δυνατότητα «μάθησης» από τα δεδομένα, πιθανώς επειδή η συνεχής έξοδος τους επέτρεπε να προσαρμοστούν πιο λεπτομερώς στις τιμές-στόχους. Ωστόσο, κατά την αξιολόγηση με δεδομένα επαλήθευσης, η εικόνα αυτή αλλάζει. Τα μοντέλα που υλοποιήθηκαν με στόχο την κατηγοριοποίηση (δηλαδή πρόβλεψη αν το σκορ θα είναι over ή under 2.5 γκολ) σημείωσαν σταθερά υψηλότερη απόδοση. Η ακρίβεια στις προβλέψεις επαλήθευσης ήταν καλύτερη στις περισσότερες περιπτώσεις, με διαφορές που έφταναν μέχρι και τις 5 ποσοστιαίες μονάδες. Παράλληλα, τα μοντέλα αυτά παρουσίασαν σημαντικά μεγαλύτερο χρηματοοικονομικό όφελος, γεγονός που ενισχύει τη θέση ότι η κατηγοριοποίηση αποτελεί πιο κατάλληλη προσέγγιση για προβλεπτικά προβλήματα με στόχο την εφαρμογή σε στοιχηματικές αποφάσεις. Η αύξηση του πλήθους των χαρακτηριστικών εισόδου από 26 σε 34 βελτίωσε επίσης σημαντικά την ακρίβεια, γεγονός που αναδεικνύει τη σημασία της πολυδιάστατης πληροφορίας στην απόδοση του μοντέλου.

Επιπλέον, φάνηκε ότι η σωστή παραμετροποίηση –όπως η επιλογή αριθμού νευρώνων στα κρυφά επίπεδα και του ρυθμού μάθησης– επηρεάζει καθοριστικά την ταχύτητα σύγκλισης και τη σταθερότητα του μοντέλου. Συγκεκριμένα, παραλλαγές όπως η χρήση ενός δεύτερου κρυφού επιπέδου με λιγότερους νευρώνες, οδήγησαν σε πιο σταθερή εκπαίδευση και παρόμοια ή καλύτερη τελική απόδοση. Επιπρόσθετα, παρατηρήθηκε ότι η χρονική κάλυψη των δεδομένων εκπαίδευσης επηρέασε θετικά την ακρίβεια. Στο τρίτο πείραμα, η χρήση μεγαλύτερου συνόλου ιστορικών δεδομένων (από το 2005 και έπειτα) προσέφερε στο μοντέλο μια πιο πλούσια βάση για μάθηση και συνέβαλε στην παραγωγή πιο αξιόπιστων προβλέψεων.

Συνοψίζοντας, τα αποτελέσματα καταδεικνύουν ότι το MLP αποτελεί ένα ιδιαίτερα ισχυρό εργαλείο για προβλεπτικά προβλήματα στο χώρο του ποδοσφαίρου, όταν συνδυάζεται με καλά επιλεγμένα στατιστικά εισόδου και κατάλληλη παραμετροποίηση. Ειδικά στην περίπτωση της κατηγοριοποίησης, η εφαρμογή του δικτύου μπορεί να οδηγήσει σε σημαντικά πρακτικά οφέλη, τόσο σε επίπεδο ακρίβειας όσο και σε

χρηματοοικονομικές αποδόσεις, κάνοντάς το ιδιαίτερα χρήσιμο εργαλείο για προβλέψεις στοιχηματικού χαρακτήρα.

4.2 Αποτελέσματα Radial Basis Function (RBF)

Προσπάθεια 1

Στο πρώτο δίκτυο χρησιμοποίησα 26 νευρώνες εισόδου και 22 κέντρα. Χρησιμοποίησα τα ίδια στατιστικά(26 σε αριθμό) για δεδομένα εισόδου που χρησιμοποιούσε και ο Φράγκος στην διπλωματική του εργασία. Δηλαδή τα ίδια στατιστικά που ανάφερα στο κεφάλαιο 3.1 εχτός αυτά που βρίσκονται στις θέσεις 23-28.

Διαστάσεις εισόδου: 26

Αριθμός κέντρων: 22

Ρυθμός μάθησης για κέντρα: 0.03

Ρυθμός μάθησης για βάρη και bias: 0.03

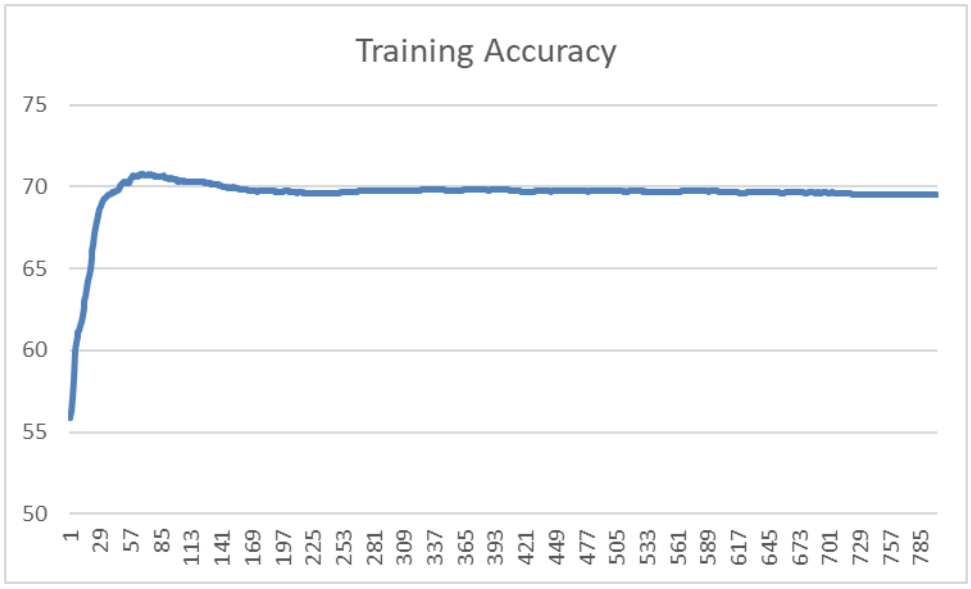
Ρυθμός μάθησης για sigma: 0.03

Χρονιές δεδομένων εκπαίδευση: 2009/10 - 2022/23

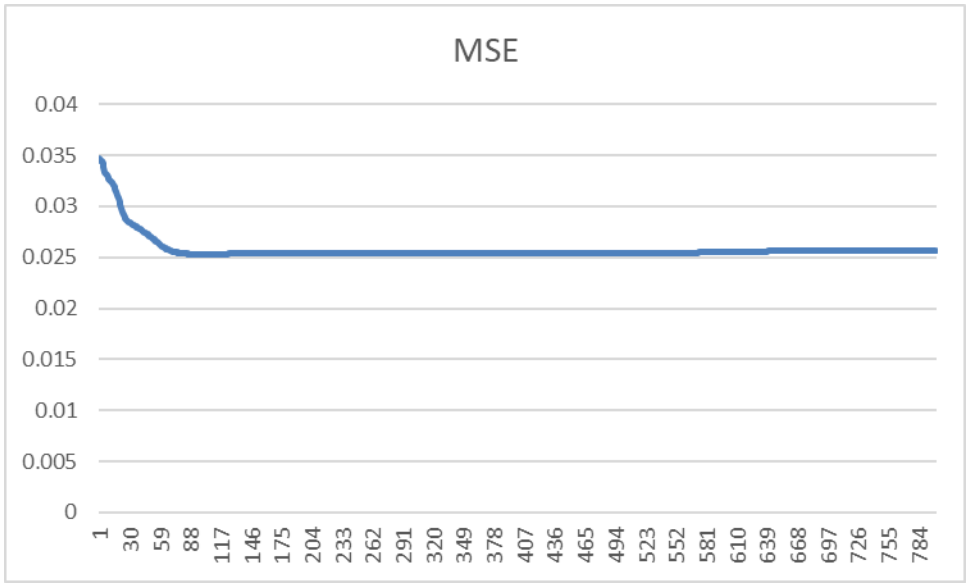
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 800

Αποτελέσματα



Εικόνα 4.13: Γραφική ακρίβειας Δεδομένων εκπαίδευσης RBF με σύνολο δεδομένων της προσπάθειας 1.



Εικόνα 4.14: Γραφική σφάλματος δεδομένων εκπαίδευσης RBF με σύνολο δεδομένων της προσπάθειας 1.

Ακρίβεια δεδομένων ελέγχου: 296/480 61.67%

Κέρδος: 377€ με επιστροφή χρημάτων στο 78.5%

Προσπάθεια 2

Σε αυτό το δίκτυο χρησιμοποίησα 34 νευρώνες εισόδου και 22 κέντρα Χρησιμοποίησα τα στατιστικά που ανάφερα στο κεφάλαιο 3.2. Στο κεφάλαιο 3.6 στο σημείο ανάλυσης της κλάσης season περιγράφεται εκτενώς τι είναι το διάνυσμα 34 θέσεων που δίνω σαν είσοδο στο δίκτυο.

Διαστάσεις εισόδου: 34

Αριθμός κέντρων: 22

Ρυθμός μάθησης για κέντρα: 0.03

Ρυθμός μάθησης για βάρη και bias: 0.03

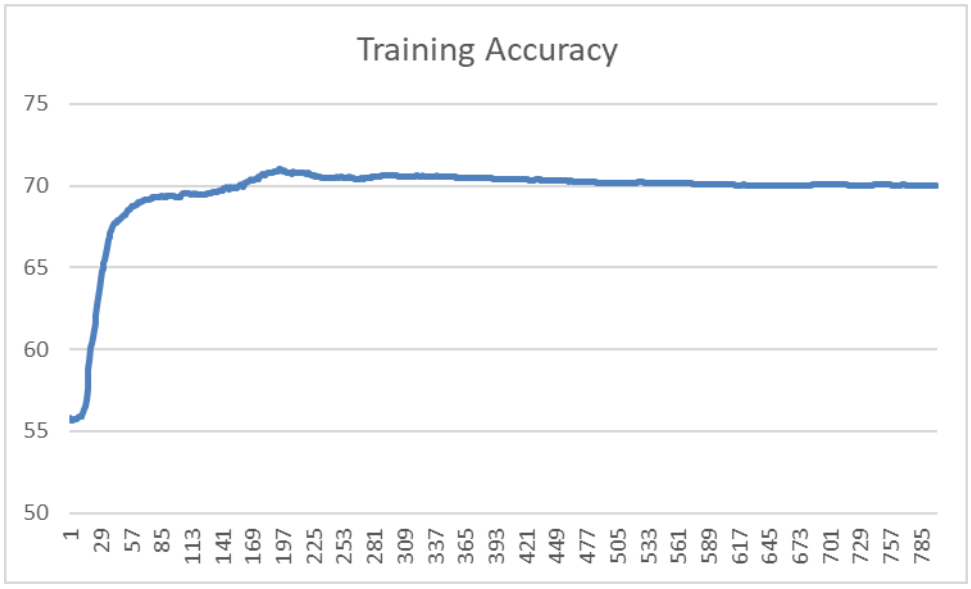
Ρυθμός μάθησης για sigma: 0.03

Χρονιές δεδομένων εκπαίδευση: 2009/10 - 2022/23

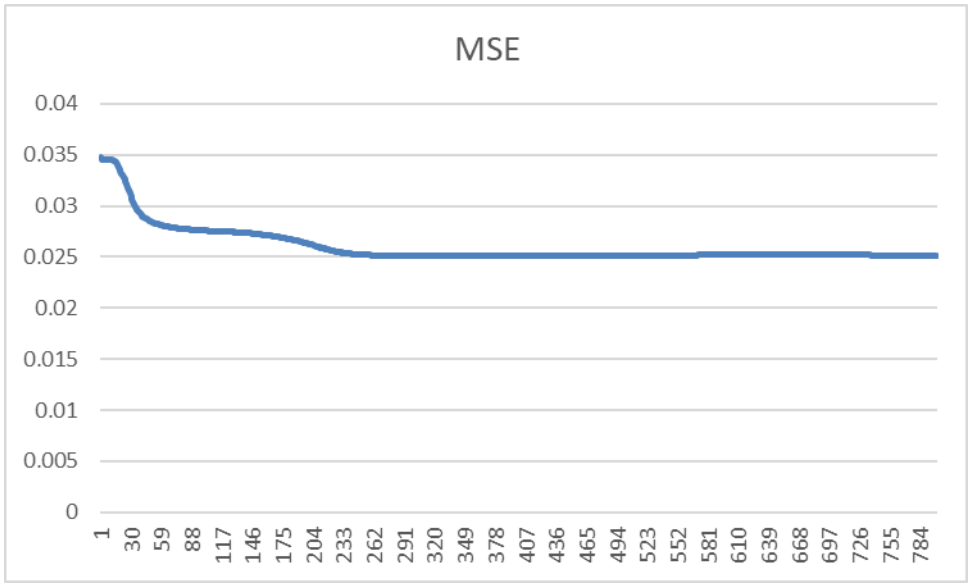
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 800

Αποτελέσματα



Εικόνα 4.15: Γραφική ακρίβειας Δεδομένων εκπαίδευσης RBF με σύνολο δεδομένων της προσπάθειας 2.



Εικόνα 4.16: Γραφική σφάλματος δεδομένων εκπαίδευσης RBF με σύνολο δεδομένων της προσπάθειας 2.

Ακρίβεια δεδομένων ελέγχου: 303/480 63.13%

Κέρδος: 377€ με επιστροφή χρημάτων στο 82.38%

Προσπάθεια 3

Σε αυτό το δίκτυο αποφάσισα να αυξήσω τον όγκο των δεδομένων εκπαίδευσης όπως και τις εποχές χαμηλώνοντας τον ρυθμό μάθησης. Χρησιμοποίησα 34 νευρώνες εισόδου και αύξησα το πλήθος των κέντρων στα 65. Τα στατιστικά που χρησιμοποίησα είναι τα ίδια με αυτά που της προσπάθειας 2.

Διαστάσεις εισόδου: 34

Αριθμός κέντρων: 65

Ρυθμός μάθησης για κέντρα: 0.005

Ρυθμός μάθησης για βάρη και bias: 0.005

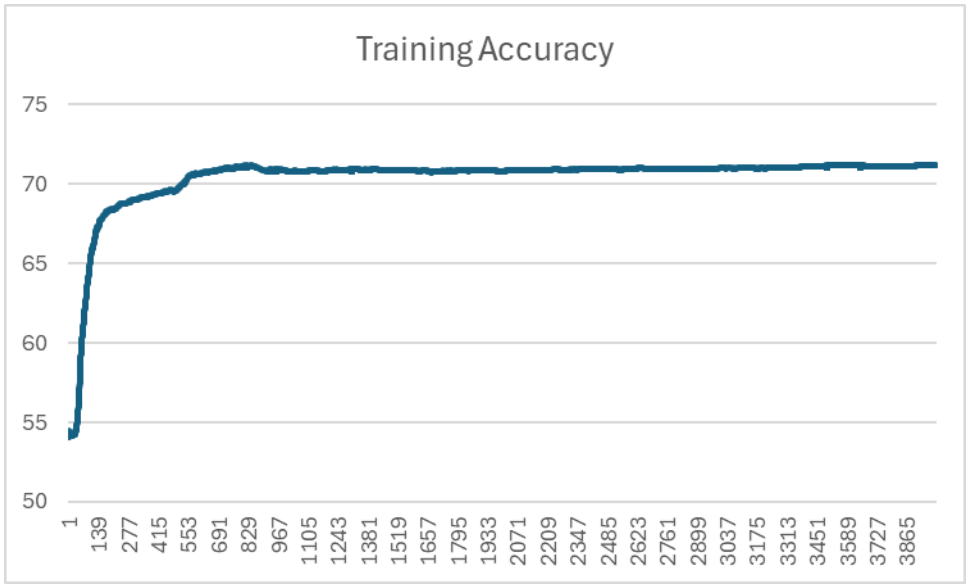
Ρυθμός μάθησης για sigma: 0.005

Χρονιές δεδομένων εκπαίδευση: 2005/06 - 2022/23

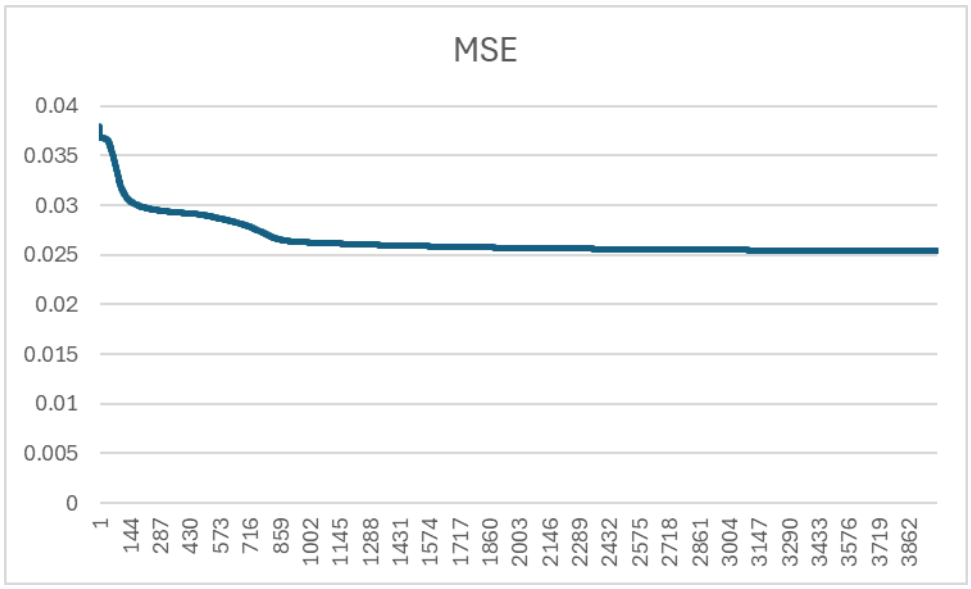
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 4000

Αποτελέσματα



Εικόνα 4.17: Γραφική ακρίβειας Δεδομένων εκπαίδευσης RBF με σύνολο δεδομένων της προσπάθειας 3.



Εικόνα 4.18: Γραφική σφάλματος δεδομένων εκπαίδευσης RBF με σύνολο δεδομένων της προσπάθειας 3.

Ακρίβεια δεδομένων ελέγχου: 317/480 66.04%

Κέρδος: 433.6€ με επιστροφή χρημάτων στο 90.3%

Προσπάθεια 4

Τα καλύτερα αποτελέσματα για τον αλγόριθμο RBF αντλήθηκαν με τα ακόλουθα δεδομένα. Αύξησα περισσότερο το πλήθος των κέντρων και εποχών μειώνοντας ταυτόχρονα τους ρυθμούς μάθησης. Διατήρησα το ίδιο σύνολο δεδομένων εκπαίδευσης και στατιστικών.

Διαστάσεις εισόδου: 34

Αριθμός κέντρων: 75

Ρυθμός μάθησης για κέντρα: 0.001

Ρυθμός μάθησης για βάρη και bias: 0.001

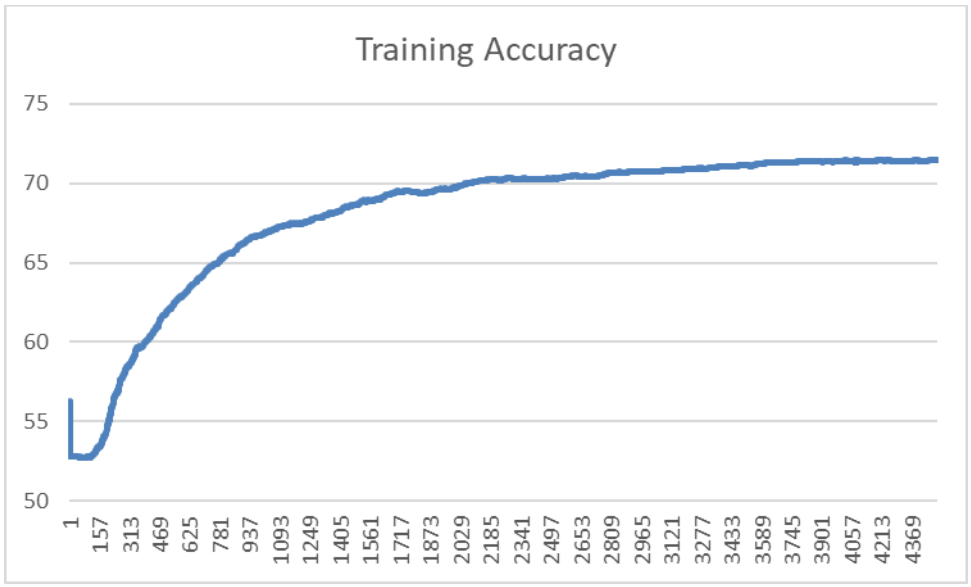
Ρυθμός μάθησης για sigma: 0.001

Χρονιές δεδομένων εκπαίδευση: 2005/06 - 2022/23

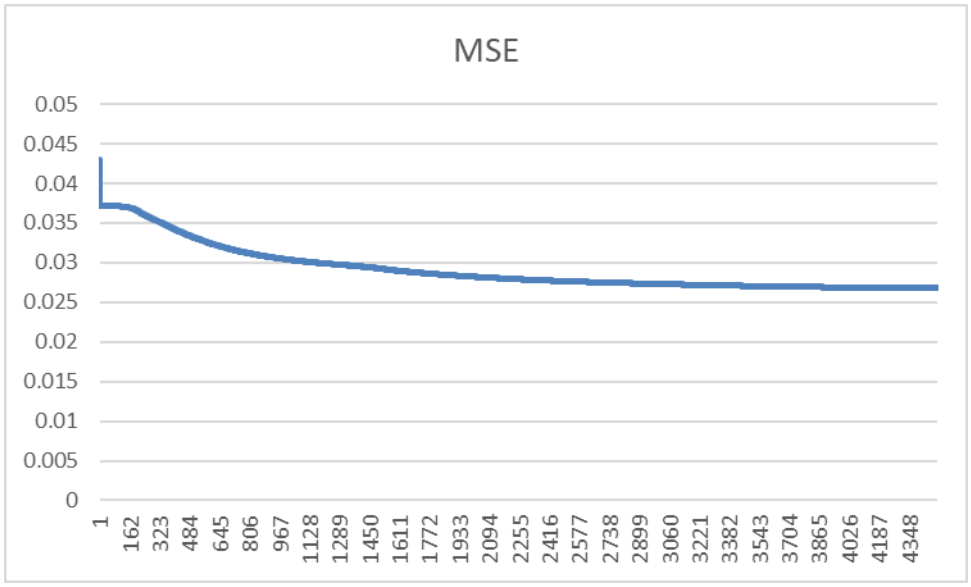
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 4500

Αποτελέσματα



Εικόνα 4.19: Γραφική ακρίβειας Δεδομένων εκπαίδευσης RBF με σύνολο δεδομένων της προσπάθειας 4.



Εικόνα 4.20: Γραφική σφάλματος δεδομένων εκπαίδευσης RBF με σύνολο δεδομένων της προσπάθειας 4.

Ακρίβεια δεδομένων ελέγχου: 350/480 72.6%

Κέρδος: 526.9€ με επιστροφή χρημάτων στο 109.77%

Τα αποτελέσματα που προέκυψαν από τις τέσσερις ποιο σημαντικές προσπάθειες που έγιναν με τον αλγόριθμο Radial Basis Function (RBF) αποκαλύπτουν ξεκάθαρες τάσεις ως προς την επίδοση του μοντέλου υπό διαφορετικές ρυθμίσεις και παραμέτρους. Σε όλες τις περιπτώσεις χρησιμοποιήθηκαν διαφορετικοί συνδυασμοί αριθμού νευρώνων εισόδου, πλήθους κέντρων και ρυθμών μάθησης, με στόχο τη βελτιστοποίηση της ακρίβειας και της οικονομικής απόδοσης του μοντέλου.

Αρχικά, στην Προσπάθεια 1, με 26 χαρακτηριστικά εισόδου και 22 κέντρα, καταγράφηκε η χαμηλότερη ακρίβεια στα δεδομένα ελέγχου (61.67% εικόνα 4.13) και το μικρότερο κέρδος (€377 με απόδοση 78.5%). Το γεγονός αυτό φανερώνει ότι το μικρότερο σε μέγεθος δίκτυο είχε περιορισμένες δυνατότητες γενίκευσης και μάθησης, παρά το ικανοποιητικό training accuracy που καταγράφηκε.

Στην Προσπάθεια 2, αυξήθηκε ο αριθμός των χαρακτηριστικών εισόδου σε 34, ενώ ο αριθμός των κέντρων διατηρήθηκε σταθερός. Αυτή η προσθήκη είχε θετική επίδραση στην ακρίβεια των δεδομένων ελέγχου (63.13% εικόνα 4.15) αλλά όχι στο κέρδος, το οποίο παρέμεινε στα ίδια επίπεδα με την πρώτη προσπάθεια. Το γεγονός αυτό υποδηλώνει ότι η αύξηση της πληροφορίας εισόδου δεν είναι από μόνη της αρκετή για ουσιαστική βελτίωση στην απόδοση του μοντέλου.

Η Προσπάθεια 3 παρουσίασε ακρίβεια δεδομένων εκπαίδευσης στο 72% (εικόνα 4.17) αλλά και μια σαφή βελτίωση, τόσο σε ακρίβεια (68.04%) όσο και σε οικονομικό όφελος (€433.6 με απόδοση 90.3%). Αυτό επιτεύχθηκε με την αύξηση των κέντρων στα 65 και τη σημαντική διεύρυνση του συνόλου δεδομένων εκπαίδευσης. Επιπλέον, η μείωση του ρυθμού μάθησης βοήθησε το δίκτυο να συγκλίνει πιο σταθερά, μειώνοντας τον κίνδυνο υπερπροσαρμογής. Εδώ φαίνεται η σημασία της σωστής εξισορρόπησης μεταξύ αριθμού κέντρων και ποιότητας δεδομένων.

Η Προσπάθεια 4 παρείχε τα καλύτερα συνολικά αποτελέσματα για το μοντέλο RBF. Με 75 κέντρα και πολύ χαμηλό ρυθμό μάθησης (0.001), το δίκτυο κατάφερε να πετύχει ακρίβεια 72.9% στα δεδομένα ελέγχου, 73.1% στα δεδομένα εκπαύδευσης (εικόνα 4.19) και κέρδος €528.9 με απόδοση στο 109.77%. Αυτό το σενάριο καταδεικνύει ότι το δίκτυο ωφελείται σημαντικά από την αυξημένη ικανότητα αναπαράστασης (μεγαλύτερο πλήθος κέντρων), όταν αυτή συνοδεύεται από επαρκή εκπαίδευση (μεγάλος αριθμός εποχών) και συντηρητικούς ρυθμούς μάθησης.

Αναλύοντας τα αποτελέσματα από τα τέσσερα διαφορετικά σενάρια που εφαρμόστηκαν με χρήση του αλγορίθμου Radial Basis Function (RBF), μπορούν να εξαχθούν ορισμένα ουσιαστικά γενικά συμπεράσματα ως προς τη συμπεριφορά και την αποτελεσματικότητα του μοντέλου σε σχέση με τις παραμέτρους εκπαίδευσης και τη δομή του δικτύου.

Πρώτον, διαπιστώνεται ότι η αύξηση της πληροφορίας εισόδου, δηλαδή ο αριθμός των χαρακτηριστικών (νευρώνων εισόδου), συμβάλλει σε μια γενική βελτίωση της απόδοσης, υπό την προϋπόθεση ότι συνοδεύεται από κατάλληλη διαχείριση των υπολοίπων παραμέτρων του μοντέλου. Η διαφορά στην ακρίβεια μεταξύ της πρώτης και της δεύτερης προσπάθειας, όπου η μοναδική μεταβολή ήταν η αύξηση των χαρακτηριστικών από 26 σε 34, ενισχύει αυτή την παρατήρηση. Ωστόσο, η μόνη αύξηση στα inputs δεν είναι από μόνη της αρκετή για ριζική βελτίωση της απόδοσης.

Δεύτερον, πολύ σημαντική αποδείχθηκε η επιλογή του αριθμού των κέντρων. Όσο αυξάνεται ο αριθμός των κέντρων, το δίκτυο αποκτά μεγαλύτερη ικανότητα προσαρμογής στις πολύπλοκες δομές των δεδομένων, κάτι που αντανακλάται άμεσα τόσο στην ακρίβεια των δεδομένων ελέγχου όσο και στο τελικό χρηματοοικονομικό κέρδος. Στις προσπάθειες 3 και 4, όπου τα κέντρα αυξήθηκαν από 65 σε 75, η απόδοση του μοντέλου ενισχύθηκε αισθητά, ιδιαίτερα στην τέταρτη προσπάθεια όπου καταγράφηκαν και τα υψηλότερα ποσοστά ακρίβειας και κερδοφορίας.

Τρίτον, καθοριστικός παράγοντας για την επιτυχία αποτέλεσε η σωστή επιλογή ρυθμών μάθησης. Μικρότεροι ρυθμοί μάθησης για τα κέντρα, τα βάρη και τα σφάλματα (sigma) οδήγησαν σε πιο σταθερή εκπαίδευση του μοντέλου και καλύτερη σύγκλιση, περιορίζοντας τον κίνδυνο υπερπροσαρμογής και βελτιώνοντας την ικανότητα γενίκευσης. Οι προσπάθειες 3 και 4 με ρυθμούς μάθησης από 0.005 και κάτω πέτυχαν σαφώς καλύτερη ακρίβεια σε σύγκριση με τις προηγούμενες.

Επιπλέον, η αύξηση των εποχών εκπαίδευσης συνεισέφερε στην ωρίμανση του μοντέλου. Η τέταρτη προσπάθεια, με 4.500 εποχές, ήταν αυτή με τη μεγαλύτερη ακρίβεια και το υψηλότερο κέρδος, γεγονός που δείχνει πως το RBF δίκτυο χρειάζεται επαρκές χρονικό διάστημα για να εκπαιδευτεί σωστά, ιδίως όταν αυξάνεται η πολυπλοκότητα του μοντέλου.

Συνολικά, τα πειραματικά αποτελέσματα δείχνουν ότι ο αλγόριθμος RBF μπορεί να αποδώσει αποτελεσματικά στο πρόβλημα πρόβλεψης ποδοσφαιρικών αγώνων, αρκεί να σχεδιαστεί προσεκτικά. Η ακρίβεια των προβλέψεων και το σχετικό οικονομικό όφελος βελτιώνονται όσο το δίκτυο ενισχύεται με περισσότερα κέντρα, μεγαλύτερο όγκο δεδομένων και συντηρητικούς ρυθμούς μάθησης. Ωστόσο, απαιτείται και σημαντική υπολογιστική προσπάθεια για την επίτευξη της μέγιστης απόδοσης, ιδιαίτερα σε πιο σύνθετες ρυθμίσεις.

4.3 Αποτελέσματα Hessian Free Optimization (HFO)

Προσπάθεια 1

Στο πρώτο δίκτυο χρησιμοποίησα 26 νευρώνες εισόδου, 2 κρυφά επίπεδα όπου το πρώτο περιείχε 10 κρυφούς νευρώνες και το δεύτερο 8 κρυφούς νευρώνες, 1 νευρώνα εξόδου με 0,05 learning rate και dumping factor. Χρησιμοποίησα τα ίδια στατιστικά(26 σε αριθμό) για δεδομένα εισόδου που χρησιμοποιούσε και ο Φράγκος στην διπλωματική του εργασία. Δηλαδή τα ίδια στατιστικά που ανάφερα στο κεφάλαιο 3.1 εχτός αυτά που βρίσκονται στις θέσεις 23-28.

Νευρώνες εισόδους: 26

Κρυφά επίπεδα: 2

Νευρώνες 1ου κρυφού επιπέδου: 10

Νευρώνες 2ου κρυφού επιπέδου: 8

Νευρώνες εξόδου: 1

Ρυθμός μάθησης: 0.05

Damping Factor: 0.05

Tolerance: 1e-3

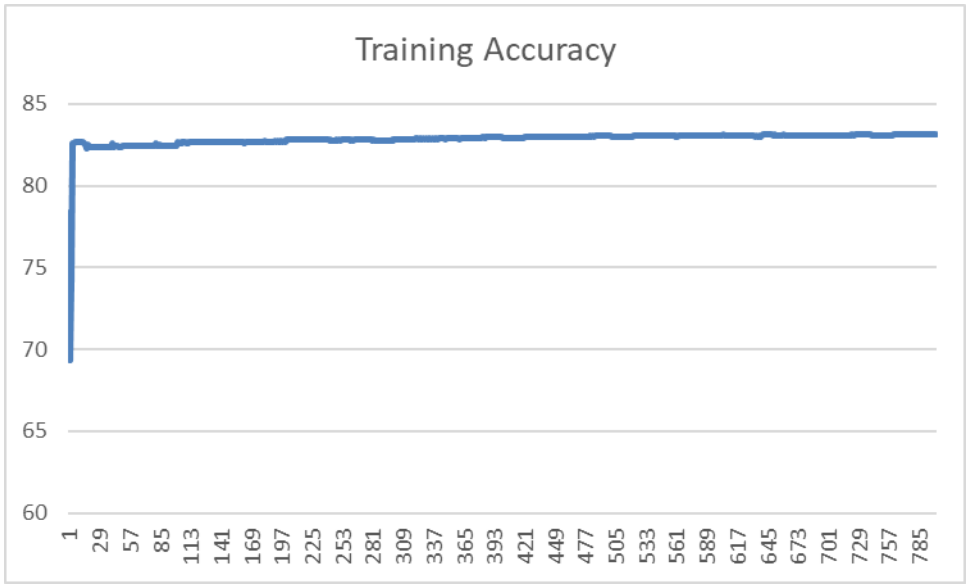
Μέγιστες επαναλήψεις βαθμίδας συζυγούς: 50

Χρονιές δεδομένων εκπαίδευση: 2009/10 - 2022/23

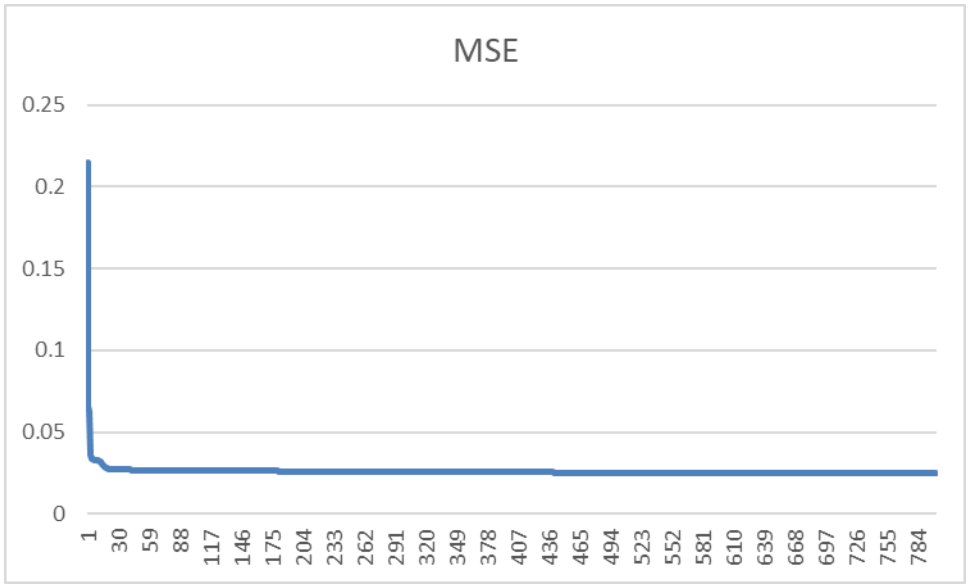
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 800

Αποτελέσματα Πρόβλεψης Τερμάτων (Επιθυμητή έξοδος: συνολικός αριθμός γκολ)



Εικόνα 4.21: Γραφική ακρίβειας Δεδομένων εκπαίδευσης HFO με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 1

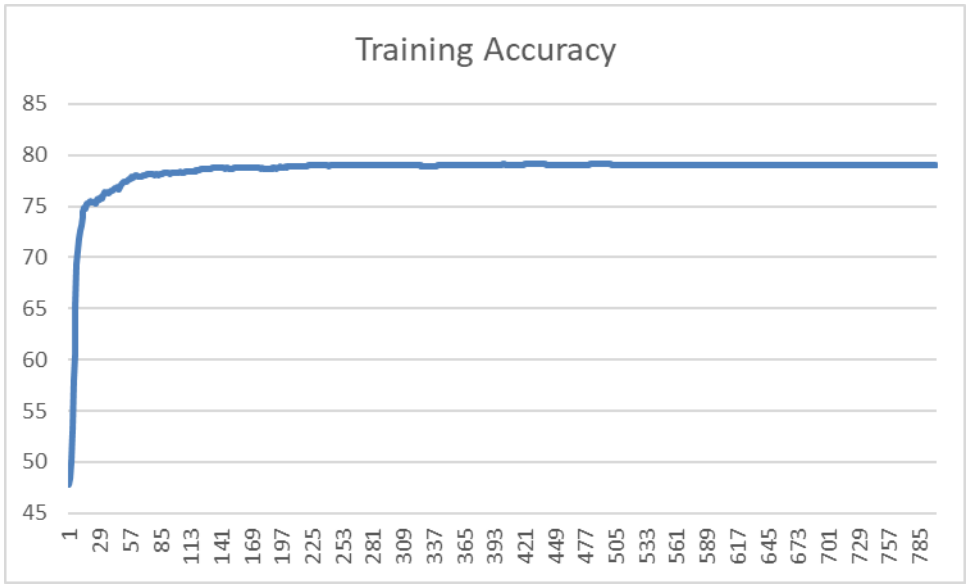


Εικόνα 4.22: Γραφική σφάλματος δεδομένων εκπαίδευσης HFO με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 1.

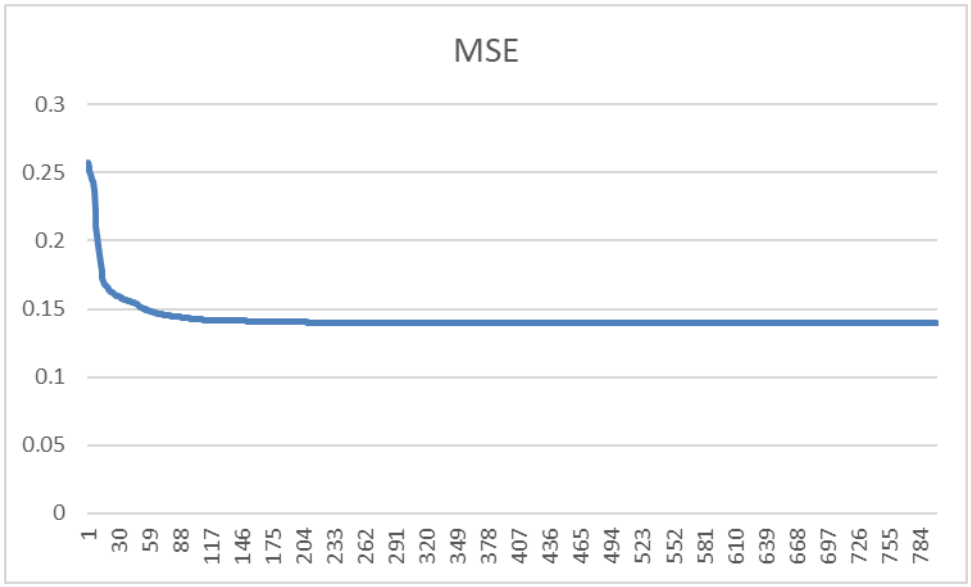
Ακρίβεια δεδομένων ελέγχου: 334/480 69.58%

Κέρδος: 483.5€ με επιστροφή χρημάτων στο 100.73%

Αποτελέσματα με Κατηγοριοποίηση(1 αν το επιθυμητό είναι over 2,5 και 0 under 2,5)



Εικόνα 4.23: Γραφική ακρίβειας Δεδομένων εκπαίδευσης HFO με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 1.



Εικόνα 4.24: Γραφική σφάλματος δεδομένων εκπαίδευσης HFO με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 1.

Ακρίβεια δεδομένων ελέγχου: 357/480 74.38%

Κέρδος: 549.7€ με επιστροφή χρημάτων στο 121.77%

Προσπάθεια 2

Σε αυτή την προσπάθεια αποφάσισα να αυξήσω τον όγκο των δεδομένων εκπαίδευσης. Επίσης προσπάθησα να μειώσω ακόμα περισσότερο το tolerance και το damping factor. Χρησιμοποίησα 34 νευρώνες εισόδου, 2 κρυφά επίπεδο όπου το πρώτο είχε 10 κρυφούς νευρώνες και το δεύτερο 5, 1 νευρώνα εξόδου. Χρησιμοποίησα τα στατιστικά που ανάφερα στο κεφάλαιο 3.2. Στο κεφάλαιο 3.6 στο σημείο ανάλυσης της κλάσης season περιγράφεται εκτενώς τι είναι το διάνυσμα 34 θέσεων που δίνω σαν είσοδο στο δίκτυο.

Νευρώνες εισόδους: 34

Κρυφά επίπεδα: 2

Νευρώνες 1ου κρυφού επιπέδου: 10

Νευρώνες 2ου κρυφού επιπέδου: 5

Νευρώνες εξόδου: 1

Ρυθμός μάθησης: 0.05

Damping Factor: 0.04

Tolerance: 1e-2

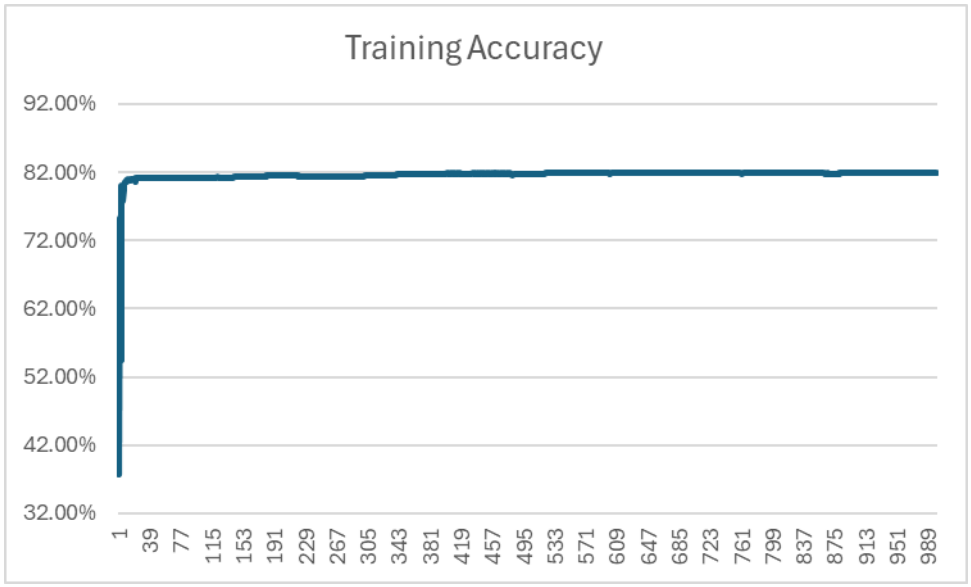
Μέγιστες επαναλήψεις βαθμίδας συζυγούς: 50

Χρονιές δεδομένων εκπαίδευση: 2005/06 - 2022/23

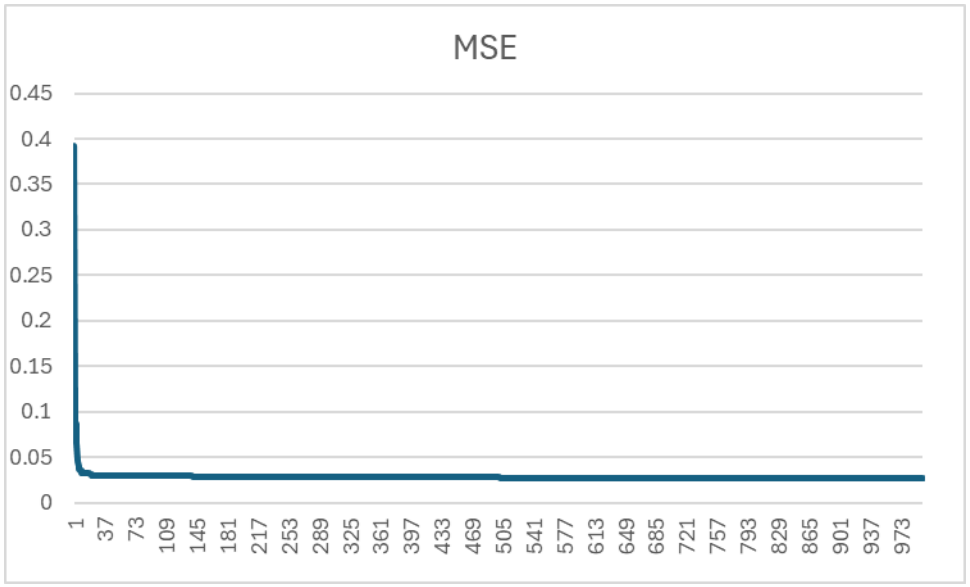
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 1000

Αποτελέσματα Πρόβλεψης Τερμάτων (Επιθυμητή έξοδος: συνολικός αριθμός γκολ)



Εικόνα 4.25: Γραφική ακρίβειας Δεδομένων εκπαίδευσης HFO με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 2.

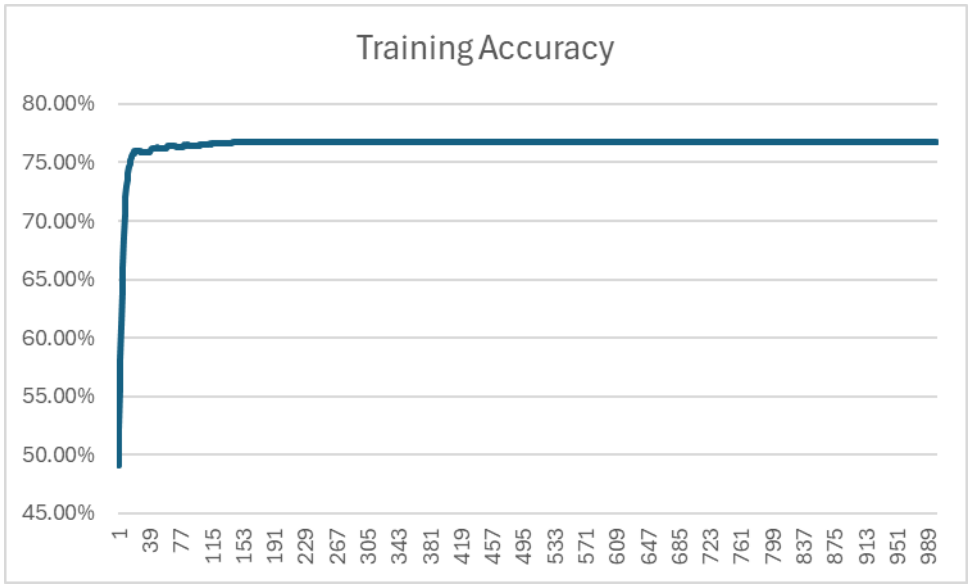


Εικόνα 4.26: Γραφική σφάλματος δεδομένων εκπαίδευσης HFO με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 2.

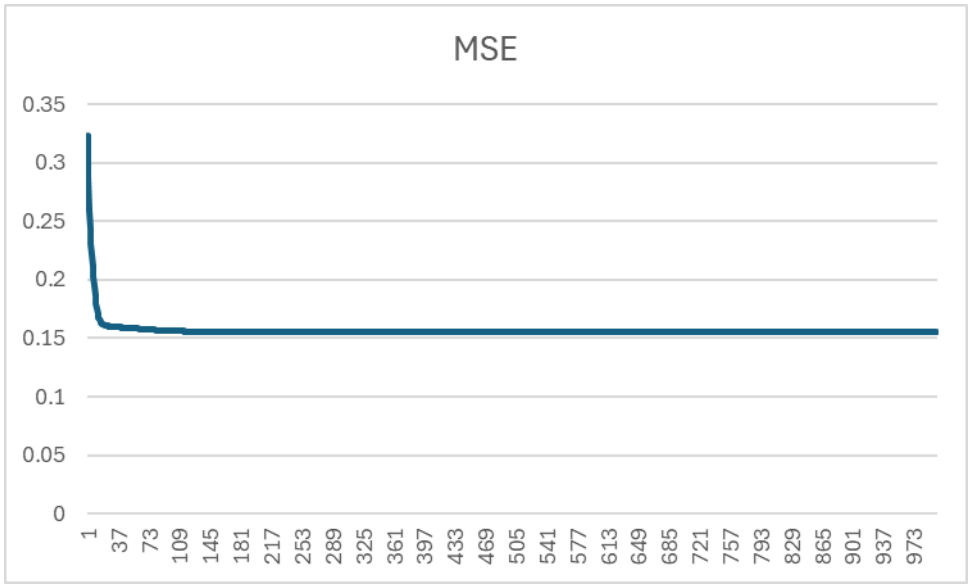
Ακρίβεια δεδομένων ελέγχου: 348/480 72.50%

Κέρδος: 522.6€ με επιστροφή χρημάτων στο 110.88%

Αποτελέσματα με Κατηγοριοποίηση(1 αν το επιθυμητό είναι over 2,5 και 0 under 2,5)



Εικόνα 4.27: Γραφική ακρίβειας Δεδομένων εκπαίδευσης HFO με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 2



Εικόνα 4.28: Γραφική σφάλματος δεδομένων εκπαίδευσης HFO με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 2.

Ακρίβεια δεδομένων ελέγχου: 362/480 75.42%

Κέρδος: 564.8€ με επιστροφή χρημάτων στο 125.67%

Προσπάθεια 3

Αφού είδα ότι τα αποτελέσματα βελτιώθηκαν με το να μειώσω το tolerance και το damping factor αποφάσισα να συνεχίσω να τα μειώνω ακόμα περισσότερο αλλά αυτή τη φορά να μειώσω και το learning rate. Χρησιμοποίηση την ίδια δομή νευρώνων όπως αυτή της προσπάθειας 2. Χρησιμοποίησα επίσης τα ίδια στατιστικά και δεδομένα. Η προσπάθεια 3 επέφερε τα καλύτερα αποτελέσματα του αλγόριθμου HFO.

Νευρώνες εισόδους: 34

Κρυφά επίπεδα: 2

Νευρώνες 1ου κρυφού επιπέδου: 10

Νευρώνες 2ου κρυφού επιπέδου: 5

Νευρώνες εξόδου: 1

Ρυθμός μάθησης: 0.02

Damping Factor: 0.03

Tolerance: 1e-1

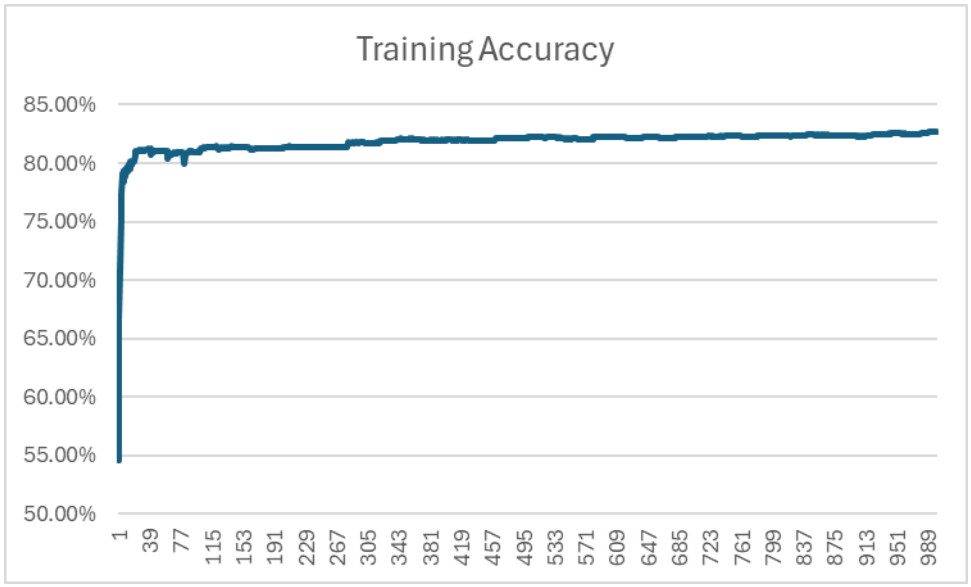
Μέγιστες επαναλήψεις βαθμίδας συζυγούς: 50

Χρονιές δεδομένων εκπαίδευση: 2005/06 - 2022/23

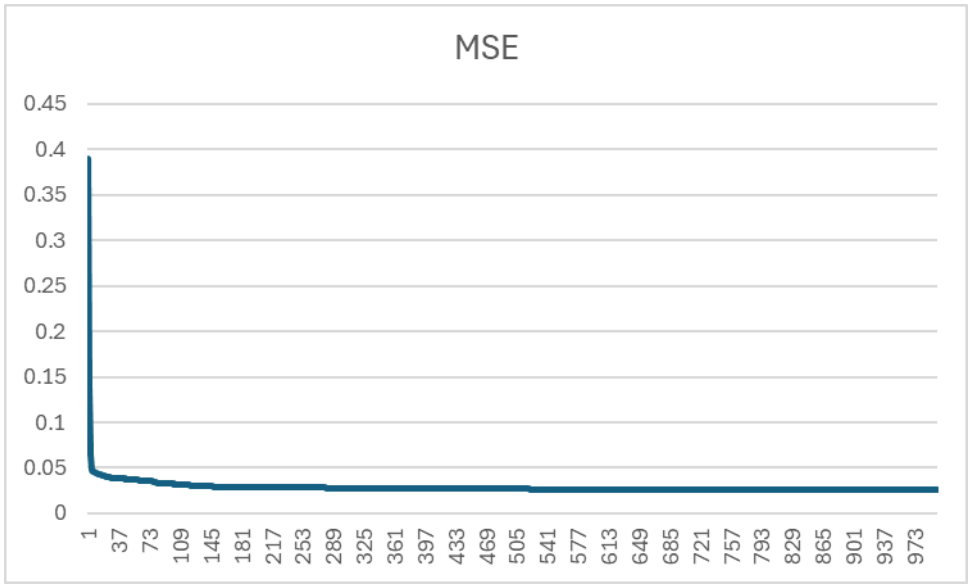
Χρονιές δεδομένων επαλήθευσης: 2023/24

Εποχές: 1000

Αποτελέσματα Πρόβλεψης Τερμάτων (Επιθυμητή έξοδος: συνολικός αριθμός γκολ)



Εικόνα 4.29: Γραφική ακρίβειας Δεδομένων εκπαίδευσης HFO με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 3.

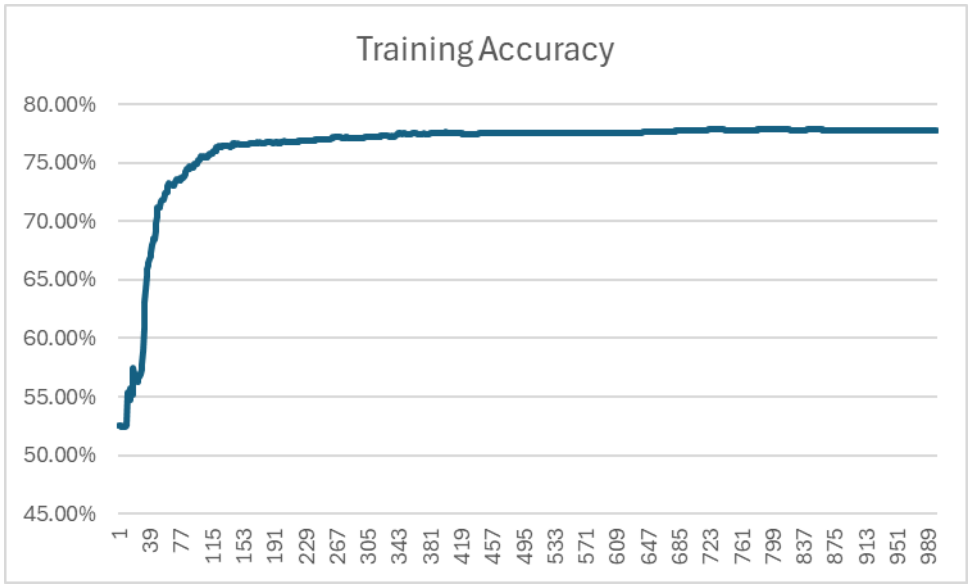


Εικόνα 4.30: Γραφική σφάλματος δεδομένων εκπαίδευσης HFO με πρόβλεψη τερμάτων και σύνολο δεδομένων της προσπάθειας 3

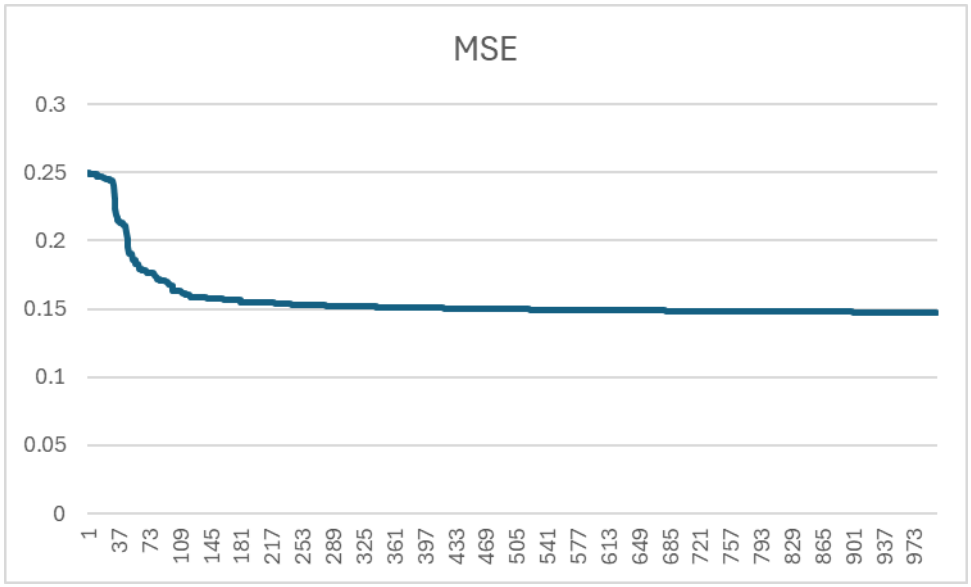
Ακρίβεια δεδομένων ελέγχου: 341/480 71.04%

Κέρδος: 503.8€ με επιστροφή χρημάτων στο 104.95%

Αποτελέσματα με Κατηγοριοποίηση(1 αν το επιθυμητό είναι over 2,5 και 0 under 2,5)



Εικόνα 4.31: Γραφική ακρίβειας Δεδομένων εκπαίδευσης HFO με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 3.



Εικόνα 4.32: Γραφική σφάλματος δεδομένων εκπαίδευσης HFO με κατηγοριοποίηση και σύνολο δεδομένων της προσπάθειας 3.

Ακρίβεια δεδομένων ελέγχου: 364/480 75.89%

Κέρδος: 571.3€ με επιστροφή χρημάτων στο 129.02%

Αξιολογήθηκαν παράμετροι όπως ο αριθμός νευρώνων εισόδου, τα επίπεδα του κρυφού στρώματος, ο ρυθμός μάθησης, ο παράγοντας απόσβεσης (damping factor) και το tolerance — η τελευταία αναπαριστά το κατώφλι ανοχής για τη σύγκλιση των επαναλήψεων εντός κάθε εποχής, με μικρότερες τιμές να συνεπάγονται πιο αυστηρά κριτήρια σύγκλισης και συνεπώς πιο ακριβή αλλά αργότερη εκπαίδευση.

Στην πρώτη προσπάθεια, χρησιμοποιήθηκαν 26 νευρώνες εισόδου και ένα πιο μικρότερο δίκτυο, το οποίο συνοδευόταν από ένα τυπικό learning rate (0.05), damping factor (0.05) και tolerance της τάξης του 1e-3. Παρότι το training accuracy της πρόβελψης συνολικού αριθμού τερμάτων ήταν αρκετά υψηλό(83.2% εικόνα 4.21), η ακρίβεια στα δεδομένα ελέγχου ήταν 69.58%, ενώ το οικονομικό αποτέλεσμα διαμορφώθηκε στα €483.5 (100.73% απόδοση). Η κατηγοριοποίηση απέδωσε ακρίβεια δεδομέων εκπαίδευσης 78.8% (εικόνα 4.23) αλλα είχε ακόμα καλύτερα αποτελέσματα δεδομένων ελέγχου (74.38% και €549.7 με 114.77% απόδοση), δείχνοντας τις δυνατότητες του μοντέλου σε binary προβλήματα.

Στη δεύτερη προσπάθεια αυξήθηκε ο αριθμός των χαρακτηριστικών εισόδου σε 34 και μειώθηκαν τόσο το tolerance (1e-2) όσο και το damping factor (0.04), διατηρώντας το learning rate σταθερό. Η ακρίβεια στα δεδομένα εκπαίδευσης παρέμεινε στα ίδια επίπεδα (εικόνες 4.25 και 4.27) ενώ τα αποτελέσματα τα αποτελέσματα στα δεδομένα ελέγχου ήταν βελτιωμένα, με την ακρίβεια να φτάνει το 72.5% για την πρόβλεψη συνολικού αριθμού τερμάτων και το 75.42% για την κατηγοριοποίηση. Αντίστοιχα, τα κέρδη ανήλθαν στα €522.6 και €564.8 αντίστοιχα. Εδώ επιβεβαιώθηκε ότι η χρήση περισσότερων χαρακτηριστικών και αυστηρότερη παραμετροποίηση έχει θετικό αντίκτυπο.

Η τρίτη προσπάθεια παρουσίασε τα καλύτερα συνολικά αποτελέσματα. Χρησιμοποιήθηκε το ίδιο πλήθος χαρακτηριστικών, αλλά το learning rate μειώθηκε στο 0.02, το damping factor στο 0.03 και το tolerance στο αυστηρό 1e-1.Τα ποσοστα ακρίβειας στα δεδομένα εκπαίδευσης (82.6% για πρόβλεψη συνολικου αριθμού τερμάτων από εικόνα 4.29 και 78.2% για κατηγοριοποίηση από εικόνα 4.30) δεν έδειξαν κάποια διαφορά σε σχέση με αυτά των 2 προηγούμενων προσπαθειών. Όμως, αυτή η πιο συντηρητική ρύθμιση οδήγησε στην καλύτερη ακρίβεια κατηγοριοποίησης δεδομένων ελέγχου (75.89%) και το μεγαλύτερο κέρδος (€571.3 με απόδοση 129.02%). Εδώ γίνεται φανερό ότι ο συνδυασμός πλούσιων δεδομένων,

αυστηρής σύγκλισης και χαμηλού ρυθμού μάθησης οδηγεί σε ένα πολύ πιο ισχυρό και γενικεύσιμο μοντέλο.

Η συνολική ανάλυση των αποτελεσμάτων που προέκυψαν από την εφαρμογή του αλγορίθμου Hessian Free Optimization (HFO) αποκαλύπτει ουσιαστικά συμπεράσματα για την επίδραση διαφορετικών παραμέτρων και επιλογών σχεδίασης στην απόδοση ενός νευρωνικού δικτύου. Ένα βασικό συμπέρασμα που προκύπτει είναι ότι ο HFO αποτελεί μια ιδιαίτερα αποδοτική μέθοδος εκπαίδευσης προσφέροντας υψηλό επίπεδο ακρίβειας και ιδιαίτερα ικανοποιητικά χρηματοοικονομικά αποτελέσματα.

Όπως φάνηκε στις επιμέρους προσπάθειες, η αύξηση του πλήθους των χαρακτηριστικών εισόδου, σε συνδυασμό με τον εμπλουτισμό του συνόλου εκπαίδευσης, βελτιώνει σημαντικά την ικανότητα γενίκευσης του δικτύου. Παράλληλα, η προσεκτική μείωση παραμέτρων όπως το damping factor, το tolerance και ο ρυθμός μάθησης συνέβαλαν καθοριστικά στην επίτευξη μεγαλύτερης σταθερότητας κατά τη διαδικασία εκπαίδευσης και αποφυγής φαινομένων υπερπροσαρμογής. Ειδικά η σταδιακή μείωση του tolerance αποδείχθηκε σημαντική, καθώς επέτρεψε πιο ακριβή επίλυση του γραμμικού συστήματος που περιλαμβάνει ο αλγόριθμος HFO, οδηγώντας σε καλύτερη σύγκλιση του μοντέλου.

Ένα ακόμη σημαντικό εύρημα είναι η υπεροχή της κατηγοριοποίησης έναντι της πρόβλεψης συνολικού αριθμού τερμάτων σε όλες τις δοκιμές, τόσο από πλευράς ακρίβειας όσο και από πλευράς τελικού οικονομικού κέρδους. Αυτό επιβεβαιώνει την καταλληλότητα της κατηγοριοποίησης για προβλήματα τύπου "over/under", όπως είναι ο στόχος στην παρούσα εργασία, δίνοντας ξεκάθαρες και πρακτικά αξιοποιήσιμες εξόδους.

Ένα επιπλέον αξιοσημείωτο συμπέρασμα που προκύπτει από την ανάλυση των αποτελεσμάτων του HFO είναι η παρατηρούμενη διαφορά στο training accuracy μεταξύ των δύο προσεγγίσεων – πρόβλεψη συνολικού αριθμού τερμάτων και κατηγοριοποίησης. Σε όλες τις προσπάθειες, το training accuracy για την πρόβλεψη συνολικού αριθμού τερμάτων είναι σταθερά υψηλότερο από αυτό της κατηγοριοποίησης. Η διαφορά αυτή αποδίδεται στο γεγονός ότι η πρόβλεψη συνολικού αριθμού τερμάτων προσπαθεί να προσεγγίσει μια συνεχόμενη τιμή (το

πλήθος των γκολ), κάτι που της επιτρέπει μεγαλύτερη ευελιξία στη διαδικασία εκμάθησης, καθώς η απόσταση μεταξύ των προβλέψεων και των στόχων μετριέται με πιο "ανοχή".

Αντίθετα, η κατηγοριοποίηση απαιτεί αυστηρότερη διάκριση μεταξύ διακριτών τάξεων (over/under 2.5), κάτι που την καθιστά περισσότερο επιρρεπή σε δυσκολίες κατά την εκμάθηση, ιδιαίτερα όταν τα δεδομένα είναι οριακά ή μη ισορροπημένα. Παρόλα αυτά, όπως φάνηκε από την τελική ακρίβεια στα δεδομένα ελέγχου και από την οικονομική απόδοση, η κατηγοριοποίηση υπερέχει πρακτικά στις συνθήκες του προβλήματος, καθώς είναι περισσότερο ευθυγραμμισμένη με τον πραγματικό στόχο του συστήματος πρόβλεψης.

Συνεπώς, παρότι η πρόβλεψη συνολικού αριθμού τερμάτων καταγράφει υψηλότερο training accuracy, η κατηγοριοποίηση αποδεικνύεται πιο αποτελεσματική στην πράξη, καθώς συνδυάζει καλή γενίκευση με υψηλό χρηματοοικονομικό όφελος. Αυτή η παρατήρηση υπογραμμίζει τη σημασία της αξιολόγησης όχι μόνο με βάση τη συμπεριφορά στα δεδομένα εκπαίδευσης, αλλά κυρίως με βάση την ικανότητα γενίκευσης και την πρακτική απόδοση του μοντέλου.

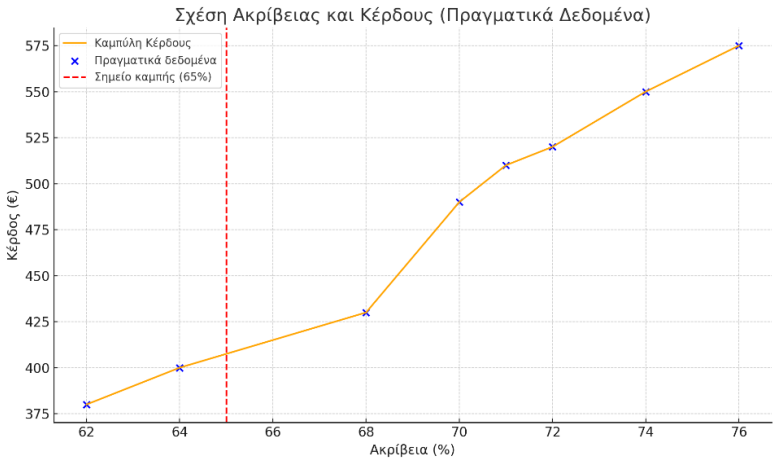
Από την ανάλυση των γραφημάτων εκπαίδευσης των τριών αλγορίθμων (MLP, RBF και HFO) είναι σαφές ότι ο αλγόριθμος HFO παρουσιάζει την ταχύτερη και πιο απότομη σύγκλιση προς υψηλά επίπεδα ακρίβειας, γεγονός που αναδεικνύει την αποδοτικότητά του. Συγκεκριμένα, στα γράφηματα του HFO παρατηρείται ότι ήδη από τις πρώτες 50 έως 100 εποχές η ακρίβεια εκτοξεύεται από χαμηλά επίπεδα σε άνω του 80% και έκτοτε σταθεροποιείται με ελάχιστες αυξομειώσεις. Το γεγονός αυτό υποδηλώνει ότι ο HFO είναι ικανός να προσαρμόζεται πολύ γρήγορα στα δεδομένα, εντοπίζοντας άμεσα τις βέλτιστες παραμέτρους, χωρίς να απαιτείται ιδιαίτερα μεγάλος αριθμός εποχών για να φτάσει σε ικανοποιητική απόδοση.

Αντίθετα, το RBF εμφανίζει μία πιο αργή πορεία σύγκλισης. Παρατηρείται ότι η ακρίβεια αυξάνεται προοδευτικά, απαιτώντας πάνω από 2000 εποχές για να φτάσει σε σημείο όπου η αύξηση επιβραδύνεται και το μοντέλο σταθεροποιείται γύρω στο 70%-72%. Αυτή η συμπεριφορά είναι χαρακτηριστική δικτύων τα οποία δεν χρησιμοποιούν πληροφορία δεύτερης τάξης, όπως κάνει ο HFO, και βασίζονται σε πιο απλές τεχνικές εκπαίδευσης.

Το MLP, αν και συγκλίνει πιο γρήγορα από το RBF, παραμένει πίσω από τον HFO. Στην περίπτωση της πρόβλεψης συνολικού αριθμού τερμάτων η ακρίβεια αυξάνεται σχετικά απότομα στις πρώτες 50-100 εποχές, αλλά χρειάζεται περίπου 200-300 εποχές για να προσεγγίσει την τελική της τιμή και να σταθεροποιηθεί γύρω στο 84%. Το φαινόμενο αυτό δείχνει ότι, παρόλο που το MLP είναι αποδοτικό, δεν διαθέτει τους μηχανισμούς που θα του επέτρεπαν να κάνει τόσο στοχευμένες ενημερώσεις των παραμέτρων όσο ο HFO.

Μέσα από την οπτική ανάλυση των γραφημάτων γίνεται φανερό ότι ο HFO είναι ο πιο γρήγορος και αποδοτικός αλγόριθμος από τους τρεις. Ο λόγος για αυτήν την υπεροχή έγκειται στο γεγονός ότι, σε αντίθεση με τους υπόλοιπους που βασίζονται αποκλειστικά σε μεθόδους πρώτης τάξης (όπως η απλή gradient descent), ο HFO αξιοποιεί πληροφορία δεύτερης τάξης μέσω του πίνακα Hessian, επιτρέποντάς του να "πηγαίνει απευθείας" στα ακρότατα σημεία της επιφάνειας σφάλματος. Αυτό μεταφράζεται σε πιο στοχευμένες και ταχύτερες διορθώσεις των βαρών και συνεπώς σε πολύ ταχύτερη σύγκλιση.

Η ανάλυση των πειραμάτων αναδεικνύει μια σημαντική σχέση μεταξύ της ακρίβειας των προβλέψεων και του συνολικού κέρδους που αποκομίζεται. Η αύξηση της ακρίβειας δεν οδηγεί σε γραμμική αύξηση των κερδών αλλά σε εκθετική. Το διάγραμμα απεικονίζει ξεκάθαρα πως όσο προχωρούν οι προβλέψεις και βελτιώνεται η ακρίβεια (ιδιαίτερα μετά το 65-70%), το κέρδος εκτοξεύεται με πολύ ταχύτερους ρυθμούς. Αυτό το φαινόμενο δεν είναι τυχαίο και εξηγείται από τη φύση του στοιχήματος.



Εικόνα 4.33: Σχέση ακρίβειας και κέρδους που δείχνει την εκθετική αύξηση του κέρδους μέσω του αλγορίθμου HFO.

Συγκεκριμένα, όταν οι αποδόσεις είναι κοντά στο 2.00, όπως συνήθως συμβαίνει στις αγορές Over/Under 2.5, κάθε σωστή πρόβλεψη ουσιαστικά διπλασιάζει το ποντάρισμα. Αντιθέτως, κάθε λανθασμένη πρόβλεψη αφαιρεί απλώς το ποντάρισμα. Αυτό σημαίνει ότι το καθαρό κέρδος (Net Profit) ενισχύεται σημαντικά όταν η ακρίβεια ξεπεράσει το σημείο ισορροπίας (break-even), που συνήθως βρίσκεται περίπου στο 50%. Από εκεί και πέρα, κάθε επιπλέον ποσοστιαία μονάδα ακρίβειας προσφέρει δυσανάλογα μεγάλο οικονομικό όφελος. Το διάγραμμα, λοιπόν, δικαιολογεί αυτή τη συμπεριφορά, δείχνοντας ότι μετά από ένα σημείο, το συνολικό κέρδος δεν αυξάνεται απλά αλλά εκθετικά καθώς οι επιτυχημένες προβλέψεις συσσωρεύονται και οι αποδόσεις των επιτυχιών συσσωρεύονται αθροιστικά.

Επιπρόσθετα, αυτή η εκθετική αύξηση οφείλεται και σε έναν δεύτερο παράγοντα. Καθώς αυξάνεται το ποσοστό επιτυχημένων προβλέψεων, αυξάνεται και η διάρκεια των κερδοφόρων σερί, μειώνοντας τις απώλειες από τυχόν συνεχόμενες αποτυχημένες προβλέψεις. Το γεγονός αυτό σταθεροποιεί και ενισχύει περαιτέρω το συνολικό κέρδος.

Συνολικά, τα αποτελέσματα καταδεικνύουν πως για να επιτευχθεί η βέλτιστη απόδοση ενός μοντέλου HFO απαιτείται συντονισμένος σχεδιασμός που να περιλαμβάνει κατάλληλη επιλογή πλήθους χαρακτηριστικών, επαρκή αριθμό εποχών, ισχυρή εκπαιδευτική βάση και σωστή παραμετροποίηση. Η προσαρμογή αυτών των παραμέτρων ανάλογα με τις απαιτήσεις του προβλήματος μπορεί να οδηγήσει σε σημαντική αύξηση της ακρίβειας και της πρακτικής του χρησιμότητας, επιβεβαιώνοντας την αξία του HFO ως ένα ισχυρό εργαλείο στο πεδίο της πρόβλεψης αποτελεσμάτων σε αθλητικά γεγονότα.

Στοιχηματικό κέρδος με πραγματικές αποδόσεις

Μετά την ολοκλήρωση της εκπαίδευσης και της αξιολόγησης του δικτύου με τα σύνολα δεδομένων εκπαίδευσης και ελέγχου, επέκτεινα την ανάλυση στις αναμετρήσεις της 46ης αγωνιστικής. Κατά τη στιγμή των προβλέψεών μου για τα παιχνίδια της 46ης αγωνιστικής, αυτά δεν είχαν ακόμη διεξαχθεί, άρα οι αποδόσεις που χρησιμοποίησα ήταν πραγματικές. Με αυτόν τον τρόπο υπολόγισα το τυχόν κέρδος από τον στοιχηματισμό βάσει των προβλέψεων του νευρωνικού δικτύου μου. Συγκεκριμένα, για κάθε αγώνα πόνταρα ένα ευρώ: αν η πρόβλεψη ήταν σωστή, εισέπραττα την απόδοση που προσέφερε η bet365· αν ήταν λάθος, έχανα το ευρώ που είχα ποντάρει.

Αγώνας	**Under 2,5**	**Over 2,5**	**Πρόβλεψη**	**Επιστροφή**
Bristol City - Preston	1,73	2,10	Over 2,5	2,10
Burnley - Millwall	1,91	1,91	Over 2,5	1,91
Coventry - Middlesbrough	2,30	1,62	Under 2,5	0
Derby – Stoke City	1,50	2,63	Under 2,5	1,50
Norwich - Cardiff	2,30	1,62	Over 2,5	1,62
Plymouth - Leeds	2,75	1,44	Under 2,5	0
Portsmouth – Hull City	1,91	1,91	Under 2,5	1,91
Sheffield Utd - Blackburn	1,80	2,00	Under 2,5	1,80
Sunderland - QPR	2,00	1,80	Under 2,5	2,00

Swansea - Oxford Utd	1,91	1,91	Over 2,5	1,91
Watford – Sheffield Wed	2,30	1,62	Over 2,5	0
West Brom - Luton	1,80	2,00	Under 2,5	0

Συνολικό ποσό στοιχηματισμού: €12

Επιστροφές: €14,75 με 122.92% ποσοστιαία επιστροφή.

Κεφάλαιο 5

Συμπεράσματα και Μελλοντική Εργασία

5.1 Συμπεράσματα και μελλοντική εργασία

Η εργασία αυτή ολοκληρώνεται με ένα σύνολο κρίσιμων συμπερασμάτων που προέκυψαν από την εκτενή ανάλυση, υλοποίηση και αξιολόγηση τριών διαφορετικών αλγορίθμων τεχνητών νευρωνικών δικτύων: των MLP (Multilayer Perceptrons), RBF (Radial Basis Function Networks) και HFO (Hessian Free Optimization). Στόχος της μελέτης ήταν η πρόβλεψη του αν θα σημειωθούν πάνω ή κάτω από 2.5 γκολ (over/under) σε κάθε ποδοσφαιρικό αγώνα του αγγλικού πρωταθλήματος Championship, ένα πρόβλημα που χαρακτηρίζεται από υψηλή αβεβαιότητα, καθώς σπάνια υπάρχουν εμφανή φαβορί ή προφανή στατιστικά μοτίβα, όπως συμβαίνει σε άλλου τύπου προβλέψεις όπως η πρόβλεψη 1X2. Το γεγονός αυτό καθιστά την πρόβλεψη over/under ιδιαίτερα απαιτητική και κατάλληλη για τη δοκιμή της ικανότητας των νευρωνικών δικτύων να αναγνωρίσουν μη γραμμικά μοτίβα από πολύπλοκα σύνολα δεδομένων.

Ένα από τα σημαντικότερα ευρήματα της εργασίας είναι ότι ο αλγόριθμος HFO παρουσίασε την καλύτερη συνολική απόδοση, τόσο ως προς την ακρίβεια προβλέψεων στα δεδομένα ελέγχου, όσο και ως προς την οικονομική απόδοση (σεναριακή αξιολόγηση μέσω simulated betting strategy). Αυτό καθίσταται ακόμη πιο αξιοσημείωτο αν ληφθεί υπόψη ότι πρόκειται για την πρώτη φορά που ο HFO χρησιμοποιείται για την επίλυση ενός τέτοιου προβλήματος πρόβλεψης στοιχηματικών αποτελεσμάτων. Αντίθετα, οι MLP και RBF έχουν δοκιμαστεί και σε παλαιότερες μελέτες ωστόσο η παρούσα εργασία κατάφερε να επιτύχει αισθητά καλύτερα αποτελέσματα συγκριτικά με προηγούμενες

μελέτες όχι λόγω πιο επιλεγμένων ή εμπλουτισμένων χαρακτηριστικών, αλλά κυρίως επειδή αξιοποιήθηκε σημαντικά μεγαλύτερος όγκος δεδομένων εκπαίδευσης. Συγκεκριμένα, χρησιμοποιήθηκαν πολλές αγωνιστικές περίοδοι (πολλαπλές σεζόν) από το αγγλικό πρωτάθλημα Championship, κάτι που επέτρεψε στα μοντέλα να εκπαιδευτούν σε μεγαλύτερη ποικιλία αγώνων και στατιστικών μοτίβων, βελτιώνοντας έτσι την ικανότητα γενίκευσης και προβλεπτικής ακρίβειας. Αυτή η αύξηση του πλήθους των δεδομένων αποτέλεσε καθοριστικό παράγοντα στη συνολική απόδοση, ανεξαρτήτως της αρχιτεκτονικής του εκάστοτε μοντέλου.

Σε ό,τι αφορά τη συμπεριφορά τον MLP, παρατηρήθηκε ότι η καλή επίδοση στα training δεδομένα δεν μεταφράστηκε πάντα σε επιτυχία στα test data. Το φαινόμενο αυτό υποδηλώνει την ύπαρξη πιθανής υπερπροσαρμογής σε κάποιες περιπτώσεις, κυρίως όταν το δίκτυο γινόταν πολύπλοκο χωρίς παράλληλη αύξηση της ποιότητας των χαρακτηριστικών ή της επαρκούς κανονικοποίησης. Τα RBF παρουσίασαν σταδιακή βελτίωση όσο αυξανόταν ο αριθμός των κέντρων και επιλέγονταν κατάλληλες τιμές για τις υπερπαραμέτρους, ωστόσο η σταθερότητά τους ήταν χαμηλότερη σε σχέση με τα υπόλοιπα μοντέλα.

Από την άλλη πλευρά, ο HFO κατάφερε να επιτύχει αξιοσημείωτα αποτελέσματα από τα πρώτα κιόλας πειράματα, κυρίως λόγω της φύσης του ως μεθόδου δεύτερης τάξης, η οποία λαμβάνει υπόψη την καμπυλότητα της επιφάνειας σφάλματος. Ο αλγόριθμος αξιοποίησε δυναμικά το curvature information χωρίς την ανάγκη υπολογισμού ή αποθήκευσης ολόκληρου του πίνακα Hessian, παρέχοντας μεγαλύτερη ακρίβεια, πιο σταθερή σύγκλιση και αυξημένο financial gain. Παρόλο που η υλοποίηση του HFO είναι πιο σύνθετη και απαιτητική υπολογιστικά, τα αποτελέσματά του τον καθιστούν την πιο υποσχόμενη μέθοδο για περαιτέρω μελέτη και εφαρμογή σε παρόμοια προβλήματα.

Αξιοσημείωτο είναι επίσης το γεγονός ότι σε όλα τα μοντέλα –ιδιαίτερα στα RBF και MLP– παρατηρήθηκε ότι η ακρίβεια της πρόβλεψης συνολικού αριθμού τερμάτων στα δεδομένα εκπαίδευσης ήταν υψηλότερη από αυτή της κατηγοριοποίησης (over/under). Ωστόσο, όταν αξιολογήθηκαν στα δεδομένα ελέγχου, η κατηγοριοποίηση αποδείχθηκε πιο αξιόπιστη και σταθερή, γεγονός που υποδηλώνει ότι η πρόβλεψη συνολικού αριθμού τερμάτων ήταν πιο ευάλωτη στο overfitting. Αυτό ενισχύει την άποψη ότι, όταν ο τελικός

στόχος είναι η βελτίωση της πρακτικής χρηστικότητας των προβλέψεων (π.χ. κερδοφορία), η κατηγοριοποίηση αποτελεί μια πιο στιβαρή προσέγγιση.

Από τη συγκριτική αξιολόγηση των τριών αλγορίθμων – MLP, RBF και HFO – προκύπτουν σαφή συμπεράσματα σχετικά με την απόδοσή τους ως προς το πρόβλημα πρόβλεψης τερμάτων και over/under. Ο MLP, ως κλασικός αλγόριθμος εμπρόσθιας τροφοδότησης με backpropagation, προσέφερε σταθερά και προβλέψιμα αποτελέσματα, αλλά με περιορισμένη δυνατότητα εκμάθησης πιο πολύπλοκων συσχετίσεων, ιδιαίτερα στο πρόβλημα κατηγοριοποίησης. Ο RBF εμφάνισε μεγαλύτερη ευαισθησία στις παραμέτρους, ειδικά στον αριθμό των κέντρων, αλλά όταν διαμορφώθηκε σωστά, παρουσίασε ανταγωνιστικά αποτελέσματα. Ωστόσο, ο πιο αποδοτικός αλγόριθμος, τόσο ως προς την ακρίβεια όσο και ως προς την οικονομική απόδοση, ήταν ξεκάθαρα ο HFO (Hessian Free Optimization). Η ικανότητά του να αξιοποιεί τη γεωμετρία της επιφάνειας σφάλματος μέσω του προσεγγιστικού υπολογισμού της καμπυλότητας τού επέτρεψε να συγκλίνει πιο σταθερά και να επιτύχει ανώτερες προβλέψεις. Τα πειράματα κατέδειξαν ότι ο HFO υπερτερεί σημαντικά των υπολοίπων, τόσο στο σκέλος της πρόβλεψης συνολικού αριθμού τερμάτων όσο και της κατηγοριοποίησης, αποδεικνύοντας πως αποτελεί ιδανική επιλογή για προβλήματα με μεγάλη πολυπλοκότητα, όπως η πρόβλεψη αποτελεσμάτων σε αθλητικά γεγονότα.

Συνοψίζοντας, η εργασία αυτή απέδειξε ότι το πρόβλημα του στοιχηματισμού στο ποδόσφαιρο μπορεί να αντιμετωπιστεί με αποτελεσματικότητα με τη βοήθεια τεχνητών νευρωνικών δικτύων, ειδικά όταν αυτά συνοδεύονται από κατάλληλα επιλεγμένα και επεξεργασμένα στατιστικά δεδομένα. Ο HFO φάνηκε να αποτελεί την πιο αποδοτική και ελπιδοφόρα προσέγγιση, αλλά και οι MLP και RBF απέδωσαν ιδιαίτερα ικανοποιητικά όταν ρυθμίστηκαν σωστά. Η πολυπλοκότητα του προβλήματος απαιτεί έξυπνη χρήση της πληροφορίας, κατάλληλη αρχιτεκτονική και συνεχείς πειραματισμούς, ώστε να επιτευχθούν προβλέψεις με πρακτική αξία και αξιοπιστία.

Μια πιθανή κατεύθυνση για μελλοντική εργασία θα μπορούσε να αφορά τη διεύρυνση και ενίσχυση της παρούσας προσέγγισης τόσο σε επίπεδο δεδομένων όσο και σε επίπεδο αλγοριθμικών μοντέλων. Αρχικά, η ενσωμάτωση δεδομένων από περισσότερα πρωταθλήματα, διαφορετικές κατηγορίες ή και διεθνείς διοργανώσεις θα μπορούσε να

οδηγήσει σε πιο γενικεύσιμα μοντέλα πρόβλεψης. Παράλληλα, η χρήση real-time στατιστικών δεδομένων ή δυναμικών χαρακτηριστικών, όπως τρέχουσα φόρμα, τραυματισμοί, πρόσφατες μεταγραφές ή ακόμα και ψυχολογικοί παράγοντες (π.χ. σημασία αγώνα, πίεση αποτελέσματος), μπορεί να προσδώσει μεγαλύτερη ακρίβεια στο σύστημα.

Μια ακόμα σημαντική επέκταση για μελλοντική εργασία αφορά την ενσωμάτωση της μετρικής Expected Goals (xG) [35] . Η συγκεκριμένη στατιστική προσφέρει έναν πολύ πιο αντικειμενικό και λεπτομερή τρόπο αξιολόγησης της ποιότητας των ευκαιριών σκοραρίσματος που δημιουργεί ή δέχεται μια ομάδα, ανεξαρτήτως του αν αυτές καταλήγουν τελικά σε γκολ. Συγκεκριμένα, το xG (αναμενόμενα γκολ) εκτιμά την πιθανότητα που έχει κάθε τελική προσπάθεια να καταλήξει σε γκολ, λαμβάνοντας υπόψη πληθώρα παραγόντων όπως η απόσταση από την εστία, η γωνία εκτέλεσης, ο τύπος της τελικής προσπάθειας (σουτ, κεφαλιά κτλ), η κατάσταση του παίκτη (αν πιέζεται ή όχι), ακόμα και το αν η φάση προήλθε από αντεπίθεση ή στημένη μπάλα. Για παράδειγμα, ένα σουτ από το ύψος του πέναλτι χωρίς πίεση έχει υψηλό xG, ενώ ένα μακρινό σουτ από πλάγια θέση έχει πολύ χαμηλό xG.

Στο πλαίσιο της μελλοντικής εργασίας, θα μπορούσε να αξιοποιηθεί ένα σύνολο δεδομένων που να περιλαμβάνει xG υπέρ και κατά ανά αγώνα, καθώς και τρέχοντες μέσους όρους xG, xG διαφοράς, ή ακόμα και xG per shot. Αυτές οι πληροφορίες μπορούν να χρησιμοποιηθούν είτε ως νέα χαρακτηριστικά εισόδου (features) για τα νευρωνικά δίκτυα, είτε ως βάσεις για δημιουργία πιο εξειδικευμένων δεικτών αξιολόγησης απόδοσης.

Η ενσωμάτωση του xG είναι επίσης χρήσιμη για την πρόβλεψη αγώνων τύπου over/under, καθώς συχνά τα αποτελέσματα επηρεάζονται από τη σύμπτωση ή την αποτελεσματικότητα και όχι από τη σταθερή επιθετική παραγωγή. Το xG βοηθά στον εντοπισμό "τυχερών" αποτελεσμάτων και συμβάλλει στη δημιουργία μοντέλων που βασίζονται σε πιο ρεαλιστική αποτύπωση της αγωνιστικής εικόνας. Συνεπώς, η αξιοποίησή του θα μπορούσε να αποτελέσει έναν ουσιαστικό παράγοντα αναβάθμισης της ακρίβειας και της αξιοπιστίας των μοντέλων που στοχεύουν στην πρόβλεψη αγώνων ποδοσφαίρου.

πιπλέον, η μελλοντική εργασία θα μπορούσε να εξετάσει τη χρήση υβριδικών ή ensemble μοντέλων, τα οποία θα συνδυάζουν διαφορετικούς τύπους νευρωνικών δικτύων όπως χρήση Συνελικτικών Νευρωνικών Δικτύων (Convolutional Neural Networks - CNNs). Η δομή ενός CNN επιτρέπει στο μοντέλο να εντοπίσει συγκεκριμένα μοτίβα ή συνδυασμούς χαρακτηριστικών που εμφανίζονται σε κοντινές αγωνιστικές – κάτι που μπορεί να είναι εξαιρετικά χρήσιμο, καθώς ομάδες συχνά παρουσιάζουν διαδοχικές τάσεις (φόρμα, αγωνιστικά μοτίβα, σκαμπανεβάσματα) μέσα στη σεζόν. Επιπλέον, μέσω της ικανότητας του CNN να εντοπίζει χωροχρονικά μοτίβα μέσα στα δεδομένα, θα μπορούσαν να μοντελοποιηθούν πιο σύνθετες σχέσεις ανάμεσα στα στατιστικά των ομάδων και τα τελικά αποτελέσματα.

Η υλοποίηση ενός CNN για τέτοιου τύπου πρόβλεψη θα απαιτούσε κατάλληλη αναδιαμόρφωση των δεδομένων εισόδου σε μορφή πίνακα ή «εικόνας», π.χ. με τις αγωνιστικές να αποτελούν γραμμές και τα χαρακτηριστικά στήλες. Σε τέτοιο σενάριο, ένα CNN θα μπορούσε να μάθει πώς τα στατιστικά εξελίσσονται με την πάροδο του χρόνου και να εντοπίσει κρίσιμες χρονικές περιόδους που προμηνύουν υψηλά σκορ.

Συνεπώς, η εφαρμογή CNNs σε συνδυασμό με εμπλουτισμένα χαρακτηριστικά, όπως τα Expected Goals (xG), θα μπορούσε να αποτελέσει ένα καινοτόμο και ελπιδοφόρο ερευνητικό βήμα με στόχο την περαιτέρω αύξηση της ακρίβειας και αξιοπιστίας στις προβλέψεις ποδοσφαιρικών αγώνων.

Επιπλέον, μια ιδιαίτερα ενδιαφέρουσα κατεύθυνση για μελλοντική μελέτη είναι η ενσωμάτωση τεχνικών επεξηγήσιμης τεχνητής νοημοσύνης (Explainable AI - XAI). Οι τεχνικές αυτές επιτρέπουν την καλύτερη κατανόηση του τρόπου με τον οποίο τα νευρωνικά δίκτυα λαμβάνουν αποφάσεις, αναδεικνύοντας ποια χαρακτηριστικά (input features) επηρεάζουν περισσότερο την πρόβλεψη. Μέσω εργαλείων όπως οι SHAP values ή οι LIME αναλύσεις, θα μπορούσαν να εντοπιστούν κρίσιμα στατιστικά που οδηγούν σε over ή under αποτελέσματα, ενισχύοντας τόσο την εμπιστοσύνη του χρήστη στο μοντέλο όσο και τη δυνατότητα βελτίωσης των εισόδων. Τέτοια προσέγγιση θα είχε ιδιαίτερη αξία σε εφαρμογές στοιχηματισμού, καθώς οι αποφάσεις με οικονομικό ρίσκο απαιτούν διαφάνεια και αιτιολόγηση.

Μια ενδιαφέρουσα προέκταση για μελλοντική εργασία θα μπορούσε να είναι η αξιοποίηση των Transformers, ενός ισχυρού αρχιτεκτονικού μοντέλου που έχει φέρει σημαντικές επιδόσεις στον χειρισμό ακολουθιακών και χρονικών δεδομένων. Η εφαρμογή τους στην πρόβλεψη ποδοσφαιρικών αποτελεσμάτων μπορεί να επιτρέψει την πιο αποτελεσματική κατανόηση της χρονολογικής εξέλιξης στατιστικών (π.χ. πρόσφατη φόρμα ομάδων), καθώς μέσω του μηχανισμού προσοχής (attention) μπορούν να εστιάσουν σε κρίσιμες αγωνιστικές ή μεταβλητές που επηρεάζουν έντονα το τελικό αποτέλεσμα. Η επεξεργασία δεδομένων πολλών προηγούμενων αγώνων ως χρονική ακολουθία μπορεί να αποκαλύψει πρότυπα απόδοσης που δύσκολα ανιχνεύονται με συμβατικά δίκτυα. Επιπλέον, λόγω της δυνατότητάς τους να συλλαμβάνουν μακροχρόνιες εξαρτήσεις, τα Transformers είναι κατάλληλοι υποψήφιοι για την ανάπτυξη πιο ακριβών και γενικεύσιμων μοντέλων, ειδικά σε προβλήματα υψηλής πολυπλοκότητας όπως το στοίχημα over/under.

Βιβλιογραφία

[1] Matt Jackson (2001) “Football Result Predictor”, MSc project, School of Computer Science and Information Systems, Birkbeck College, University of London.

[2] Kevin Davies (2004) “A Mug’s Game? Predicting the Result of Football Matches Using Neural Networks”, MSc project, School of Computer Science and Information Systems, Birkbeck College, University of London.

[3] Sian Turner (2005) “Neural Networks and football: is there money to be made?”, MSc project, School of Computer Science and Information Systems, Birkbeck College, University of London.

[4] Θεοφάνης Νεοκλέους (2007) “Πρόβλεψη Ποδοσφαιρικών Αποτελεσμάτων με Τεχνητά Νευρωνικά Δίκτυα : μπορεί να είναι κερδοφόρα;”, Ατομική Διπλωματική Εργασία, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου

[5] Ιακώβου Α. (2010) “Πρόβλεψη ποδοσφαιρικών αποτελεσμάτων με τη χρήση τεχνητών νευρωνικών δικτύων”, Διπλωματική Εργασία, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου

[6] Φράγκου Α. (2016) “Πρόγνωση συνολικού αριθμού επίτευξης τερμάτων ποδοσφαιρικών αγώνων μέσω deep neuronal networks”, Τμήμα πληροφορικής, Πανεπιστήμιο Κύπρου

[7] Constantinou, A. C., & Fenton, N. E. (2018). Predicting football match outcomes: A comparative review of statistical and machine learning models. International Journal of Forecasting, 34(2), 227–236. https://doi.org/10.1016/j.ijforecast.2018.01.003

[8] Groll, A., Ley, C., Schauberger, G., & Van Eetvelde, H. (2022). Prediction of football match outcomes: A comparison of statistical learning methods. International Journal of Forecasting, 38(2), 733–747. https://doi.org/10.1016/j.ijforecast.2021.06.008

[9] Rischard, M., Ke, G., & Wang, Y. (2024). Enhancing sports analytics through meta-learning: A case study on football prediction. Machine Learning. https://doi.org/10.1007/s10994-024-06608-w

[10] kelifos.physics.auth.gr/COURSES/neural/K2.pdf

[11] W. S. McCulloch and W. Pitts, A logical calculus of ideas immanent in nervous activity, Bullettin of Mathematical Biophysics, 5, 115(1943).

[12] W. Pitts and W. S. McCulloch, Bullettin of Mathematical Biophysics, 9, 127(1947).

[13] J. von Neumann, The computer and the brain, Yale University Press, New Haven (1958).

[14] D. Hebb, The organisation of behavior, (1949).

[15] F. Rosenblatt, Principles of Neuodynamics, Spartan (N.Y), 1962.

[16] M. Minsky and S. Papert, Perceptrons, MIT Press (1969); expanded edition, 1988.

[17] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79,2554(1982).

[18] J. L. McClelland and D. E. Rumelhart, Parallel Distributed Processing, Volumes 1 and 2, MIT Press, Cambridge, Mass. (1986)

[19] psi-gr.tripod.com/choc_20_app_neuro.html

[20] karantzis.blogspot.com.cy/2012/10/blog-post_24.html

[21] el.wikipedia.org/wiki/Νευρωνικό_δίκτυο

[22]http://nemertes.lis.upatras.gr/jspui/bitstream/10889/4419/1/ΔΙΠΛΩΜΑΤΙΚΗ%20ΣΤΑΘΟΠΟΥΛΟΥ%20ΔΗΜΗΤΡΑΣ.pdf

[23] https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[24] https://en.wikipedia.org/wiki/Backpropagation

[25] RBF_2022_ppt Σημειώσεις μαθήματος machine learning Πανεπιστήμιο Κύπρου

[26] Martens, J. (2010) Deep learning via Hessian-free optimization. In Proceedings of the 27th International Conference on Machine Learning (ICML'10), Bottou, L., and Littman, M., (eds.) , pp. 732-742.

[27] Nash, S. G. (1984). Newton-type minimization via the Lanczos method. SIAM Journal on Numerical Analysis, 21(4), 770-788.

[28] Nash, S.G. (2000). A survey of truncated-newton methods. Journal of Computational and Applied Mathematics, 124(1), pp 45–59.

[29] Martens, J., Sutskever, I. (2012). Training Deep and Recurrent Networks with Hessian-Free Optimization. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg

[30] Hush, D. R. and Salas, J. M., (1988). Improving the learning rate of back-propagation with the gradient reuse algorithm, Proceedings of the IEEE 1988 International Conference on Neural Networks, San Diego, CA, USA, vol.1, pp. 441-447

[31] Bishop, C.M. (1995). Neural networks for pattern recognition. Oxford University Press, Oxford, UK

[32] Nocedal, J., & Write, S. SJ (1999). Numerical optimization. Springer-Verlag, New York

[33] Schraudolph, N. (2002). Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent. Neural Computation, 14(7), pp.1723-1738.

[34] Rasmussen, D. (2015), Hessian-free optimization for deep networks, GitHub repository, https://github.com/drasmuss/hessianfree, May, 2018.

[35] https://statsbomb.com/soccer-metrics/expected-goals-xg-explained/

Παράρτημα Α

Κώδικας υλοποίησης FileEditor σε java.

```
import com.opencsv.CSVReader;
import com.opencsv.exceptions.CsvException;

import java.io.*;
import java.nio.file.*;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class FileEditor {
  public static void processSeason(int season, String inputDir, String outputDir) {
    String inputFilePath = inputDir + File.separator + "E1_" + season + ".csv";
    String outputFilePath = outputDir + File.separator + "championship_" + season +
".txt";

    try (
      Reader reader = Files.newBufferedReader(Paths.get(inputFilePath));
      CSVReader csvReader = new CSVReader(reader);
      BufferedWriter writer = new BufferedWriter(new FileWriter(outputFilePath))
    ) {
      // Skip the header row
        // Extract the header row
        List<String[]> rows = csvReader.readAll();
      String[] header = rows.get(0);
      Map<String, Integer> columnIndexMap = new HashMap<>();
      for (int i = 0; i < header.length; i++) {
        columnIndexMap.put(header[i], i);
      }

      //Find index of over and under odds because in each excel file over and under
columns are not in the same place
      int overIndex=0;
      int underIndex=0;
      int HomeTeamIndex=0;
      int AwayTeamIndex=0;
```

```
        int HomeTeamGoalsIndex=0;
        int AwayTeamGoalsIndex=0;
        for (int i = 0; i < header.length; i++) {
          if (header[i].contains(">2.5")) {
            overIndex = i;
          } else if (header[i].contains("<2.5")) {
            underIndex = i;
          }

          if(header[i].equals("HomeTeam")) {
                HomeTeamIndex = i;
          }
          if(header[i].equals("AwayTeam")) {
                AwayTeamIndex = i;
          }
          if(header[i].equals("FTHG")) {
                HomeTeamGoalsIndex = i;
          }
          if(header[i].equals("FTAG")) {
                AwayTeamGoalsIndex = i;
          }
        }

        rows.remove(0); // Assuming first row is header

        // Process each match row
        for (String[] row : rows) {
          String homeTeam = row[HomeTeamIndex];  // HomeTeam column
          String awayTeam = row[AwayTeamIndex];  // AwayTeam column
          String homeGoals = row[HomeTeamGoalsIndex]; // FTHG (Full Time Home
Goals)
          String awayGoals = row[AwayTeamGoalsIndex]; // FTAG (Full Time Away Goals)
          String overOdds = row[overIndex];
          String underOdds = row[underIndex];

          // Format the result: Team A - Team B:GoalsA:GoalsB
          if(!homeTeam.isEmpty() && !awayTeam.isEmpty() && !homeGoals.isEmpty() &&
!awayGoals.isEmpty() && !overOdds.isEmpty() && !underOdds.isEmpty()) {
              String result = homeTeam + "-" + awayTeam + "-" + homeGoals + ":" +
awayGoals + " " + overOdds + "/" + underOdds;
```

```
                // Write the result to the file
                writer.write(result);
                writer.newLine();
            }
        }

        System.out.println("Processed season " + season + " successfully!");

    } catch (FileNotFoundException e) {
        System.out.println("File not found for season " + season + ". Skipping.");
    } catch (IOException | CsvException e) {
        e.printStackTrace();
    }
  }
}
```

Παράρτημα Β

Κώδικας υλοποίησης MLP σε java.

```
import java.util.Random;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class MLP {
   private int inputSize, hiddenSize1, hiddenSize2, outputSize;
   private double[][] weights1, weights2, weightsOutput;
   private double[] bias1, bias2, biasOutput;
   private double learningRate;
   private boolean useTwoHiddenLayers;
   private Random random;

   //Constructor: Initializes network architecture & random weights
   public MLP(int inputSize, int hiddenSize1, int hiddenSize2, int outputSize, double
learningRate, boolean useTwoHiddenLayers) {
      this.inputSize = inputSize;
      this.hiddenSize1 = hiddenSize1;
      this.hiddenSize2 = hiddenSize2;
      this.outputSize = outputSize;
      this.learningRate = learningRate;
      this.useTwoHiddenLayers = useTwoHiddenLayers;
      this.random = new Random();

      // Initialize weights and biases randomly in range (-1,1)
      weights1 = new double[inputSize][hiddenSize1];
      bias1 = new double[hiddenSize1];
      initializeWeights(weights1, bias1);

      if (useTwoHiddenLayers) {
         weights2 = new double[inputSize][hiddenSize2];
         bias2 = new double[hiddenSize2];
         initializeWeights(weights2, bias2);
      }

      weightsOutput = useTwoHiddenLayers ? new double[hiddenSize2][outputSize] :
```

```
new double[hiddenSize1][outputSize];
    biasOutput = new double[outputSize];
    initializeWeights(weightsOutput, biasOutput);
  }

  //Helper method to initialize weights & biases randomly (-1 to 1)
  private void initializeWeights(double[][] weights, double[] bias) {
    for (int i = 0; i < weights.length; i++) {
      for (int j = 0; j < weights[i].length; j++) {
        weights[i][j] = random.nextDouble() * 2 - 1;
      }
    }
    for (int i = 0; i < bias.length; i++) {
      bias[i] = random.nextDouble() * 2 - 1;
    }
  }

  //ReLU activation function for a single value
  private double relu(double x) {
    return Math.max(0, x);
  }

  //ReLU activation function for an array (element-wise)
  private double[] relu(double[] x) {
    double[] result = new double[x.length];
    for (int i = 0; i < x.length; i++) {
      result[i] = Math.max(0, x[i]);
    }
    return result;
  }

  //ReLU derivative for a single value
  private double reluDerivative(double x) {
    return (x > 0) ? 1 : 0;
  }

  //ReLU derivative for an array (element-wise)
  private double[] reluDerivative(double[] x) {
    double[] result = new double[x.length];
    for (int i = 0; i < x.length; i++) {
      result[i] = (x[i] > 0) ? 1 : 0;
```

```
    }
    return result;
  }


  //Matrix-vector multiplication with bias
  private double[] matrixVectorMultiply(double[][] weights, double[] input, double[]
bias) {
    double[] output = new double[weights[0].length];

    for (int j = 0; j < weights[0].length; j++) {
      output[j] = bias[j]; // Add bias
      for (int i = 0; i < weights.length; i++) {
        output[j] += weights[i][j] * input[i];
      }
    }

    return output;
  }
```

... (ΣΥΝΕΧΕΙΑ ΘΑ ΠΕΡΑΣΕΙ ΜΕ ΤΟΝ ΙΔΙΟ ΤΡΟΠΟ - λόγω περιορισμού μηνύματος κόπηκε εδώ για την ώρα)

```
// -------------------- TRAIN METHOD --------------------

public void train(double[][] inputs, double[] targets, int epochs) {
  String logPath = "C:\\Users\\micha\\EPL442\\training_MLP_goals.txt";
  try (BufferedWriter writer = new BufferedWriter(new FileWriter(logPath))) {
    for (int epoch = 0; epoch < epochs; epoch++) {
      double mse = 0;
      int correctPredictions = 0;
      for (int i = 0; i < inputs.length; i++) {
        double[] hidden1 = relu(matrixVectorMultiply(weights1, inputs[i], bias1));
        double[] hidden2 = null;
        if (useTwoHiddenLayers) hidden2 = relu(matrixVectorMultiply(weights2,
hidden1, bias2));
        double output = (useTwoHiddenLayers)
            ? relu(matrixVectorMultiply(weightsOutput, hidden2, biasOutput))[0]
            : relu(matrixVectorMultiply(weightsOutput, hidden1, biasOutput))[0];
```

```
            double error = targets[i] - output;
            mse += error * error;
            boolean predictedOver = output >= 0.5;
            boolean actualOver = targets[i] >= 0.5;
            if (predictedOver == actualOver) correctPredictions++;
            double deltaOutput = error * reluDerivative(output);
            double[] deltaHidden2 = null;
            if (useTwoHiddenLayers) {
              deltaHidden2 = new double[hiddenSize2];
              for (int j = 0; j < hiddenSize2; j++) deltaHidden2[j] = deltaOutput *
weightsOutput[j][0] * reluDerivative(hidden2[j]);
            }
            double[] deltaHidden1 = new double[hiddenSize1];
            if (useTwoHiddenLayers) {
              for (int j = 0; j < hiddenSize1; j++) {
                for (int k = 0; k < hiddenSize2; k++) deltaHidden1[j] += deltaHidden2[k] *
weights2[j][k];
                deltaHidden1[j] *= reluDerivative(hidden1[j]);
              }
            } else {
              for (int j = 0; j < hiddenSize1; j++) deltaHidden1[j] = deltaOutput *
weightsOutput[j][0] * reluDerivative(hidden1[j]);
            }
            if (useTwoHiddenLayers) {
              for (int j = 0; j < weightsOutput.length; j++) weightsOutput[j][0] += learningRate
* deltaOutput * hidden2[j];
              biasOutput[0] += learningRate * deltaOutput;
            } else {
              for (int j = 0; j < weightsOutput.length; j++) weightsOutput[j][0] += learningRate
* deltaOutput * hidden1[j];
              biasOutput[0] += learningRate * deltaOutput;
            }
            if (useTwoHiddenLayers) {
              for (int j = 0; j < hiddenSize1; j++) for (int k = 0; k < hiddenSize2; k++)
weights2[j][k] += learningRate * deltaHidden2[k] * hidden1[j];
            }
            for (int j = 0; j < inputSize; j++) for (int k = 0; k < hiddenSize1; k++) weights1[j][k]
+= learningRate * deltaHidden1[k] * inputs[j][i];
          }
          mse /= inputs.length;
          double accuracy = (correctPredictions / (double) inputs.length) * 100.0;
```

```
        String logLine = "Epoch " + epoch + ", MSE: " + mse + ", Accuracy: " +
String.format("%.2f", accuracy) + "%";
        writer.write(logLine); writer.newLine();
      }
   } catch (IOException e) { e.printStackTrace(); }
}

// -------------------- TEST METHOD --------------------

public void test(double[][] testX, double[] actualY, double minGoals, double maxGoals,
double[] odds) {
   double mse = 0; int correctPredictions = 0; int totalMatches = testX.length; int
gameweek = 7; int matches = 0;
   double totalProfit = 0.0, gameweekProfit = 0.0, matchProfit = 0.0;

   String logPath = "C:\\Users\\micha\\EPL442\\test_MLP_goals.txt";
   try (BufferedWriter writer = new BufferedWriter(new FileWriter(logPath))) {
      for (int i = 72; i < totalMatches; i++) {
         matchProfit = 0.0;
         double predicted = forward(testX[i]);
         matches++;
         double denormalizedPredicted = predicted * (maxGoals - minGoals) + minGoals;
         double denormalizedActual = actualY[i] * (maxGoals - minGoals) + minGoals;
         boolean predictedOver = denormalizedPredicted >= 2.5;
         boolean actualOver = denormalizedActual >= 2.5;
         if (predictedOver == actualOver) {
            correctPredictions++;
            matchProfit = odds[i] * 1.0;
            totalProfit += matchProfit;
            gameweekProfit += matchProfit;
         } else {
            totalProfit -= 1.0;
            gameweekProfit -= 1.0;
         }
         String matchLog = "Match " + (i + 1) + " - Predicted: " + denormalizedPredicted +
               ", Actual: " + denormalizedActual + " - " +
               (predictedOver == actualOver ? "Correct" : "Incorrect") +
               " Profit: " + matchProfit + "€";
         writer.write(matchLog); writer.newLine();
         if ((i + 1) % 12 == 0) {
            writer.write("Gameweek " + gameweek + " profit: " + gameweekProfit + "€");
```

```
writer.newLine();
            gameweek++; gameweekProfit = 0.0;
          }
          mse += Math.pow(denormalizedPredicted - denormalizedActual, 2);
      }
      mse /= totalMatches;
      double accuracy = (correctPredictions / (double) matches) * 100;
      double percentReturn = (totalProfit / matches) * 100.0;

      writer.write("Final MSE on test set: " + mse); writer.newLine();
      writer.write("Correct Predictions: " + correctPredictions + " / " + matches + " Total
Profit: " + totalProfit + "€"); writer.newLine();
      writer.write("Prediction Accuracy: " + accuracy + "%"); writer.newLine();
      writer.write("ROI: " + percentReturn + "%"); writer.newLine();
    } catch (IOException e) { e.printStackTrace(); }
}
```

Παράρτημα Γ

Κώδικας υλοποίησης RBFNetwork σε java.

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

public class RBFNetwork {

   private int inputDim;
   private int numCenters;
   private double[][] centers;
   private double[] sigmas;
   private double[] weights;
   private double bias;

   private double etaW;
   private double etaC;
   private double etaSigma;

   public RBFNetwork(int inputDim, int numCenters, double etaW, double etaC, double
etaSigma) {
      this.inputDim   = inputDim;
      this.numCenters = numCenters;
      this.etaW      = etaW;
      this.etaC      = etaC;
      this.etaSigma   = etaSigma;

      centers = new double[numCenters][inputDim];
      sigmas  = new double[numCenters];
      weights = new double[numCenters];
      bias = 0.0;
      initializeParameters();
   }

   private void initializeParameters() {
      Random rand = new Random();
      for (int i = 0; i < numCenters; i++) {
```

```
        for (int j = 0; j < inputDim; j++) {
          centers[i][j] = 2.0 * rand.nextDouble() - 1.0;
        }
        sigmas[i] = 1.0;
      }
      for (int i = 0; i < numCenters; i++) {
        weights[i] = 0.02 * rand.nextDouble() - 0.01;
      }
      bias = 0.0;
    }

    private double gaussianBasis(double[] x, double[] center, double sigma) {
      double distSq = 0.0;
      for (int i = 0; i < inputDim; i++) {
        double diff = x[i] - center[i];
        distSq += diff * diff;
      }
      return Math.exp(-distSq / (2.0 * sigma * sigma));
    }

    public double predict(double[] x) {
      double output = bias;
      for (int i = 0; i < numCenters; i++) {
        double phi = gaussianBasis(x, centers[i], sigmas[i]);
        output += weights[i] * phi;
      }
      return output;
    }

    public void train(double[][] inputs, double[] targets, int epochs, double minGoals,
double maxGoals) {
      int N = inputs.length;
      String logFilePath = "C:\\Users\\micha\\EPL442\\train_RBF_goals.txt";

      try (BufferedWriter writer = new BufferedWriter(new FileWriter(logFilePath))) {
        writer.write("Epoch\tMSE\tTraining Accuracy (%)");
        writer.newLine();

        for (int epoch = 0; epoch < epochs; epoch++) {
          double mse = 0.0;
          int correctPredictions = 0;
```

```
        for (int p = 0; p < N; p++) {
          double[] x = inputs[p];
          double t = targets[p];

          double[] phi = new double[numCenters];
          double y_hat = bias;

          for (int i = 0; i < numCenters; i++) {
            phi[i] = gaussianBasis(x, centers[i], sigmas[i]);
            y_hat += weights[i] * phi[i];
          }

          double error = t - y_hat;
          mse += error * error;

          for (int i = 0; i < numCenters; i++) {
            weights[i] += etaW * error * phi[i];
          }
          bias += etaW * error;

          for (int i = 0; i < numCenters; i++) {
            double w_i = weights[i];
            double phi_i = phi[i];
            double[] diff = new double[inputDim];
            for (int d = 0; d < inputDim; d++) {
              diff[d] = x[d] - centers[i][d];
            }
            double centerFactor = etaC * error * w_i * phi_i / (sigmas[i] * sigmas[i]);
            for (int d = 0; d < inputDim; d++) {
              centers[i][d] += centerFactor * diff[d];
            }
            double distSq = 0.0;
            for (int d = 0; d < inputDim; d++) {
              distSq += diff[d] * diff[d];
            }
            double sigmaFactor = etaSigma * error * w_i * phi_i * distSq /
Math.pow(sigmas[i], 3);
            sigmas[i] += sigmaFactor;
            if (sigmas[i] < 1e-6) {
              sigmas[i] = 1e-6;
```

```
            }
          }

          double denormPredicted = y_hat * (maxGoals - minGoals) + minGoals;
          double denormTarget = t * (maxGoals - minGoals) + minGoals;
          boolean predictedOver = denormPredicted >= 2.5;
          boolean targetOver = denormTarget >= 2.5;
          if (predictedOver == targetOver) {
            correctPredictions++;
          }
        }

        mse /= N;
        double accuracy = ((double) correctPredictions / N) * 100.0;
        String logLine = "Epoch " + (epoch + 1) + "\t" + mse + "\t" + accuracy;
        writer.write(logLine);
        writer.newLine();
        System.out.println(logLine);
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  public void test(double[][] testX, double[] actualY, double minGoals, double
maxGoals, double[] odds) {
    double mse = 0.0;
    int correctPredictions = 0;
    int totalMatches = testX.length;
    int gameweek = 7;
    int matches = 0;
    double totalProfit = 0.0;
    double gameweekProfit = 0.0;
    double matchProfit;

    String logPath = "C:\\Users\\micha\\EPL442\\test_RBF_goals.txt";

    try (BufferedWriter writer = new BufferedWriter(new FileWriter(logPath))) {
      writer.write("\nTesting RBF Network on test set...");
      writer.newLine();
      System.out.println("\nTesting RBF Network on test set...");
```

```
        for (int i = 72; i < totalMatches; i++) {
            matchProfit = 0.0;
            double predicted = predict(testX[i]);
            matches++;

            double denormalizedPredicted = predicted * (maxGoals - minGoals) +
minGoals;
            double denormalizedActual = actualY[i] * (maxGoals - minGoals) + minGoals;

            boolean predictedOver = denormalizedPredicted >= 2.5;
            boolean actualOver = denormalizedActual >= 2.5;

            double stake = 1.0;
            if (predictedOver == actualOver) {
                correctPredictions++;
                matchProfit = odds[i] * stake;
                totalProfit += matchProfit;
                gameweekProfit += matchProfit;
            } else {
                totalProfit -= stake;
                gameweekProfit -= stake;
            }

            String matchLog = "Match " + (i + 1) + " - Predicted: " + denormalizedPredicted +
                    ", Actual: " + denormalizedActual + " - " +
                    (predictedOver == actualOver ? "Correct" : "Incorrect") +
                    " Profit: " + matchProfit + "€";
            System.out.println(matchLog);
            writer.write(matchLog);
            writer.newLine();

            if ((i + 1) % 12 == 0) {
                String gwLog = "Gameweek " + gameweek + " profit: " + gameweekProfit + "€";
                System.out.println(gwLog);
                writer.write(gwLog);
                writer.newLine();
                gameweekProfit = 0.0;
                gameweek++;
            }
```

```
            mse += Math.pow(denormalizedPredicted - denormalizedActual, 2);
        }

        mse /= totalMatches;
        double accuracy = (correctPredictions / (double) matches) * 100.0;
        double totalStake = matches * 1.0;
        double percentReturn = (totalProfit / totalStake) * 100.0;

        String finalMSE = "\nFinal MSE on test set: " + mse;
        String correctPred = "Correct Predictions: " + correctPredictions + " / " + matches
+
              " with total profit: " + String.format("%.1f", totalProfit) + "€";
        String acc = "Prediction Accuracy: " + String.format("%.2f", accuracy) + "%";
        String roi = "ROI (Return on Investment): " + String.format("%.2f", percentReturn) +
"%";

        System.out.println(finalMSE);
        System.out.println(correctPred);
        System.out.println(acc);
        System.out.println(roi);

        writer.write(finalMSE);
        writer.newLine();
        writer.write(correctPred);
        writer.newLine();
        writer.write(acc);
        writer.newLine();
        writer.write(roi);
        writer.newLine();

      } catch (IOException e) {
        e.printStackTrace();
      }
    }
}
```