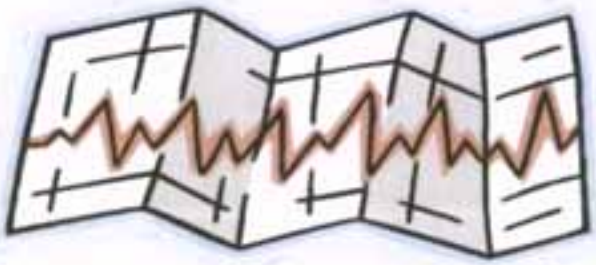
ΤΕΧΝΗΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΚΑΙ ΠΡΟΒΛΕΨΗ ΧΡΟΝΟΣΕΙΡΩΝ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΤΩΝ:

Ατσαλάκη Ζωή

Κουτζαγιώτη Αγγέλα

ΕΙΣΗΓΗΤΗΣ: Παπαϊωάννου Βάϊος

Τμήμα Επιχειρηματικού Σχεδιασμού και Πληροφοριακών Συστημάτων

Σχολή ΣΔΟ

Α.Τ.Ε.Ι. Πατρών

Μάρτιος 2005

ΤΕΧΝΗΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΓΙΑ ΠΡΟΒΛΕΨΗ ΧΡΟΝΟΣΕΙΡΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΤΟΥ:

Δ΄ΤΣΑΔΑΚΗ ΖΩΗ

Κουτσογιώτα Λογγάδα

ΕΞΕΤΑΣΤΗΣ: Παπαϊωάννου Βάσος

Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων

Σχολή ΔΙΟΔ

Α.Τ.Ε.Ι. Πάτρας

Μάρτιος 2005

Η απουσία της γνώσης των εναλλακτικών λύσεων ωθεί στην απόλυτη αποδοχή των καθιερωμένων θεωρητικών δογμάτων, και αφαιρεί κάθε δυνατότητα αμφισβήτησής τους.

Horton R., 1972

Ο σχεδιασμός είναι μια από τις πλέον περίπλοκες και δύσκολες πνευματικές δραστηριότητες με τις οποίες είναι δυνατόν να εμπλακεί ο άνθρωπος. Αν δεν γίνει καλά δεν είναι αμαρτία, αν αποδειχθεί, όμως, ότι στοχεύει χαμηλότερα από αυτό το «καλά», είναι.

Ansoff H., 1968

Ευχαριστούμε θερμά τους, Παπαϊωάννου Βάϊο, Ψαράκη Εμμανουήλ και Μπεληγιάννη Γρηγόριο για την πολύτιμη συμβολή τους στην προετοιμασία και ολοκλήρωση της πτυχιακής μας εργασίας, καθώς επίσης και όλους τους καθηγητές του τμήματος για την στήριξη που μας πρόσφεραν τα τέσσερα χρόνια των σπουδών μας, αλλά και για τις γνώσεις που μας μετέδωσαν, τις ιδέες, τις έννοιες και τα συναισθήματα που μοιράστηκαν μαζί μας ανοίγοντας μας νέους δρόμους.

Ο Νίκος Καζαντζάκης υποστηρίζει πως ιδανικός δάσκαλος είναι εκείνος που γίνεται γέφυρα για να περάσει αντίπερα ο μαθητής του και όταν πια του έχει διευκολύνει το πέρασμα, αφήνεται χαρούμενα να γκρεμιστεί, ενθαρρύνοντας το μαθητή του να φτιάξει δικές του γέφυρες. Τέτοιες γέφυρες χτίσαμε τα φοιτητικά μας χρόνια χάρης του καθηγητές μας που μας δίδαξαν με ενθουσιασμό και ειλικρίνεια και τις ίδιες γέφυρες είμαστε αποφασισμένες να συνεχίζουμε να στήνουμε.

Τέλος θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας που υπήρξαν για μας ένας ολόκληρος θησαυρός και μια ατέλειωτη πηγή έμπνευσης, θάρρους, ψυχολογικής στήριξης και αποφασιστικότητας βοηθώντας μας να αναζητήσουμε μια καινούργια διάσταση στην προσωπικότητα μας και στη ζωή μας μέσα από τους δρόμους της κατανόησης και της αγάπης.

Περίληψη

Το θέμα που διαπραγματεύεται αυτή η εργασία είναι η πρόβλεψη χρονοσειρών σε τεχνητά νευρωνικά δίκτυα. Έτσι αρχικά γίνεται μια αναφορά στην ιστορική πορεία των νευρωνικών δικτύων, στην αρχιτεκτονική τους, τα δομικά στοιχεία τους, την διαδικασία που ακολουθούν κατά την φάση της εκμάθησης τους καθώς και τους αλγόριθμους που χρησιμοποιούν όπως ο αλγόριθμος οπισθοδρόμησης (Back- propagation) και ο αλγόριθμος LMS (Least Mean Square). Στη συνέχεια παρατίθενται επιστημονικά πεδία στα οποία ευρέως χρησιμοποιούνται νευρωνικά δίκτυα, όπως στην ιατρική, στις επιχειρήσεις και μερικά άλλα που αναφέρονται στη σελίδα 9. Στα τρία τελευταία κεφάλαια της εργασίας γίνεται αναφορά στο μαθηματικό περιβάλλον «Matlab» όπου μια από τις χρήσεις του είναι η εκπαίδευση συστημάτων νευρωνικών δικτύων. Παρατίθενται κάποιες εφαρμογές των νευρωνικών δικτύων που τρέξαμε χρησιμοποιώντας τεχνητά δεδομένα καθώς και κάποιες εφαρμογές που τρέξαμε χρησιμοποιώντας κάποια πραγματικά δεδομένα.

Περιεχόμενα

1. Χρήση και εφαρμογές των νευρωνικών δικτύων

1.1 Τι είναι ένα νευρωνικό δίκτυο;

Ένα τεχνητό νευρωνικό δίκτυο (ΤΝΔ) είναι ένα σύστημα του οποίου η δομή είναι εμπνευσμένη από τον τρόπο λειτουργίας του νευρικού συστήματος και του εγκεφάλου. Το βασικό στοιχείο αυτού του συστήματος είναι η νέα δομή επεξεργασίας πληροφοριών που διαθέτει. Ένα νευρωνικό δίκτυο αποτελείται από έναν μεγάλο αριθμό διασυνδεόμενων μεταξύ τους νευρώνων που λειτουργούν σύμφωνα για να λύσουν τα συγκεκριμένα προβλήματα. Ένα ΤΝΔ εκπαιδεύεται μέσω μιας διαδικασίας, γνωστή σαν διαδικασία μάθησης, για μια συγκεκριμένη εφαρμογή, όπως για παράδειγμα για την αναγνώριση προτύπων ή την ταξινόμηση στοιχείων. Η μάθηση στα βιολογικά συστήματα περιλαμβάνει τις ρυθμίσεις στις διασυνδέσεις που υπάρχουν μεταξύ των νευρώνων. Αυτό ισχύει και στα ΤΝΔ .

1.2 Ιστορική αναδρομή

Τα νευρωνικά δίκτυα έχουν αναπτυχθεί πρόσφατα. Εντούτοις, αυτός ο τομέας καθιερώθηκε πριν από την εμφάνιση των υπολογιστών.

Η ιστορία των νευρωνικών δικτύων μπορεί να διαιρεθεί στις παρακάτω περιόδους :

1. **Πρώτες προσπάθειες:** Το 1943 οι McCulloch και Pitts ανέπτυξαν τα πρότυπα των νευρωνικών δικτύων, βασισμένοι στην επιστήμης της νευρολογίας. Αυτά τα πρότυπα έκαναν διάφορες υποθέσεις για το πώς λειτουργούν οι νευρώνες. Τα δίκτυά τους βασίστηκαν σε απλούς νεύρωνες που θεωρήθηκαν δυαδικές συσκευές με σταθερά κατώτατα όρια. Τα πρότυπα αυτά χρησιμοποιήθηκαν με επιτυχία σε απλές λογικές λειτουργίες όπως οι υλοποιήσεις των λογικών πράξεων. Μια άλλη προσπάθεια ήταν με τη χρησιμοποίηση των υπολογιστών των προσομοιώσεων. Υπήρχαν δύο ομάδες ερευνητών (Farley και Clark, 1954 Rochester, Ολλανδία, Haibit και Duda, 1956), εκ των οποίων η πρώτη ομάδα (ερευνητές της IBM) διατήρησε στενή επαφή με τους επιστήμονες της νευρολογίας στο πανεπιστήμιο McGill. Αυτή η αλληλεπίδραση καθιέρωσε μια πολυεπιστημονική τάση που συνεχίζει να υπάρχει μέχρι και σήμερα.

2. **Ελπιδοφόρος & νέα τεχνολογία:** Όχι μόνο ήταν η επίδραση της νευροεπιστήμης στην ανάπτυξη των νευρικών δικτύων, αλλά και οι ψυχολόγοι και οι μηχανικοί που

συνέβαλαν επίσης στην πρόοδο των προσομοιώσεων των νευρωνικών δικτύων. Ο Rosenblatt (1958) συνδύασε ενδιαφέροντα και δραστηριότητες στον τομέα των νευρωνικών δικτύων όταν σχεδίασε και ανέπτυξε τα perceptron. Τα Perceptron έχουν τρία επίπεδα με το μέσο επίπεδο γνωστό ως επίπεδο ένωσης. Αυτό το σύστημα μπορούσε να μαθαίνει να συνδέει ή να συσχετίζει μια δεδομένη είσοδο και μια τυχαία μονάδα εξόδου.

Ένα άλλο σύστημα ήταν το ADALINE (στοιχείο ADAptive LInear) που αναπτύχθηκε το 1960 από τον Widrow και τον Hoff (του Πανεπιστημίου του Στάνφορντ). Το ADALINE ήταν μια απλή αναλογική ηλεκτρονική συσκευή. Η μέθοδος που χρησιμοποιήθηκε για την εκπαίδευση της ήταν διαφορετική από αυτή του perceptron, η μέθοδος αυτή υιοθετούσε τον κανόνα μάθησης γνωστό ως ελάχιστον -μέσον-τετραγωνικού σφάλματος (LMS).

3. **Περίοδος απογοήτευσης & ανυποληψίας:** Το 1969 ο Minsky και ο Papert παρουσίασαν ένα βιβλίο που γενίκευε το Perceptron ενός επιπέδου σ' ένα πολυεπίπεδο σύστημα.

4. **Καινοτομία:** Αν και το δημόσιο ενδιαφέρον και η διαθέσιμη χρηματοδότηση ήταν ελάχιστα, διάφοροι ερευνητές συνέχιζαν την έρευνα με σκοπό να αναπτύξουν υπολογιστικές μεθόδους για την επίλυση προβλημάτων όπως αυτό της αναγνώριση πρότυπων. Κατά τη διάρκεια αυτής της περιόδου παρουσιάστηκαν διάφορες εφαρμογές στις οποίες τα ΤΝΔ μπορούσαν να χρησιμοποιηθούν με επιτυχία. Η επιρροή του Grossberg (Steve Grossberg και Gail Carpenter το 1988) ήταν σημαντική γιατί ίδρυσε ένα σχολείο διανόησης στο οποίο ερευνήθηκαν οι resonating αλγόριθμοι. Ανέπτυξαν τα δίκτυα ART (Adaptive Resonance Theory) βασισμένα σε βιολογικά πρότυπα. Οι Anderson και Kohonen ανέπτυξαν τη συνειρμική τεχνική, η οποία είναι ανεξάρτητη από τις άλλες τεχνικές. Ο Klopf το 1972, ανέπτυξε μια βάση για τα τεχνητούς νεύρωνες βασισμένη σε μια βιολογική αρχή για τη νευρωνική μάθηση αποκαλούμενη ως heterostasis. Ο Werbos (Paul Werbos 1974) ανέπτυξε και χρησιμοποίησε τη μέθοδο μάθησης του αλγόριθμου οπισθοδρόμησης (back-propagation). Τα δίκτυα που βασίζονται στον back propagation είναι πιθανώς σήμερα τα πιο γνωστά νευρωνικά δίκτυα. Στην ουσία ο αλγόριθμος οπισθοδρόμησης back-propagation είναι ένα perceptron με πολλαπλά

επίπεδα, επίσης μια διαφορετική λειτουργία στον τεχνητό νευρώνα είναι ο thershold που είναι ένας πιο ικανός κανόνας εκμάθησης. Ο Amari δημοσίευσε ένα έγγραφο που καθιέρωσε μια μαθηματική θεωρία για μια βάση εκμάθησης (μέθοδος λάθος-διορθώσεων) που εξετάζει την προσαρμοστική ταξινόμηση. Ενώ το Fukushima (F. Kunihiko) ανέπτυξε ένα σοφό εκπαιδευμένο πολυεπίπεδο νευρωνικό δίκτυο βημάτων για την ερμηνεία των χειρόγραφων χαρακτήρων. Το αρχικό δίκτυο δημοσιεύθηκε το 1975 και ονομάστηκε Cognitron.

5. **Επανεμφάνιση:** Η πρόοδος ανάμεσα στη δεκαετία του '70 και τη δεκαετία του '80 ήταν σημαντική και έτσι αναζωπυρώθηκε το ενδιαφέρον για τα νευρωνικά δίκτυα. Για παράδειγμα χάριν, τα περιεκτικά βιβλία και οι διασκέψεις παρείχαν ένα φόρουμ σε ανθρώπους που ασχολούνταν με διαφορετικούς τομείς και με εξειδικευμένες τεχνικές γλώσσες. Τα μέσα μαζικής ενημέρωσης με την αυξανόμενη δραστηριότητα και τα σεμινάρια ενισχύθηκαν για να διαδώσουν την τεχνολογία. Τα ακαδημαϊκά προγράμματα εμφανίστηκαν και σειρές μαθημάτων διδάχθηκαν στα περισσότερα σημαντικά πανεπιστήμια (των ΗΠΑ και της Ευρώπης). Η προσοχή τώρα στρέφεται στη χρηματοδότηση σε όλη την Ευρώπη, την Ιαπωνία και τις ΗΠΑ και αυτή η χρηματοδότηση διατίθεται, σε αρκετές νέες εμπορικές επιχειρήσεις με αιτήσεις από τη βιομηχανία και τα οικονομικά ιδρύματα.

6. **Σήμερα:** Σημαντική πρόοδος έχει σημειωθεί στον τομέα των νευρωνικών δικτύων. Η πρόοδος πέρα από τις τρέχουσες εμπορικές εφαρμογές εμφανίζεται να είναι ισχυρή και η έρευνα προωθεί τον τομέα των νευρωνικών δικτύων σε πολλά μέτωπα. Συνεπώς σήμερα είναι μια μεταβατική περίοδος για την τεχνολογία των νευρωνικών δικτύων.

1.3 Ποια η χρήση των νευρωνικών δικτύων

Τα νευρωνικά δίκτυα, με την δυνατότητά τους να αντλήσουν την έννοια από τα περίπλοκα ή ανακριβή στοιχεία, μπορούν να χρησιμοποιηθούν για να εξαγάγουν τα σχέδια και να ανιχνεύσουν τις τάσεις που είναι πάρα πολύ σύνθετες για να παρατηρηθούν είτε από υπολογιστές είτε από ανθρώπινες τεχνικές. Ένα εκπαιδευμένο νευρωνικό δίκτυο μπορεί να

θεωρηθεί ως "εμπειρογνώμονας" στην κατηγορία πληροφοριών που του έχουν δοθεί για να αναλύσει. Αυτός ο εμπειρογνώμονας μπορεί έπειτα να χρησιμοποιηθεί για να παρέχει τις προβολές δεδομένων των νέων καταστάσεων ενδιαφέροντος.

Μερικά από τα πλεονεκτήματα των νευρωνικών δικτύων είναι:

- Προσαρμοστική μάθηση (Adaptive learning): Είναι μια δυνατότητα που βοηθάει να υλοποιεί ένα νευρωνικό δίκτυο τους στόχους οι οποίοι βασίζονται στα στοιχεία που δίνονται για την κατάρτιση ή την αρχική εμπειρία.
- Αυτο-οργάνωση (Self- Organization): Ένα ΤΝΔ μπορεί να δημιουργήσει την οργάνωση ή την αντιπροσώπευσή των πληροφοριών που λαμβάνει κατά τη διάρκεια του χρόνου εκμάθησης.
- Λειτουργία πραγματικού χρόνου(Real Time Operation): Οι υπολογισμοί ΤΝΔ μπορούν να πραγματοποιηθούν παράλληλα, με τις ειδικές συσκευές υλικού που σχεδιάζονται και κατασκευάζονται για να εκμεταλλεύονται αυτήν την ικανότητα.
- Ανοχή ελαττωμάτων με την βοήθεια τεχνικών που βασίζονται στην κωδικοποίηση πλεονάζουσας πληροφορίας (Fault Tolerance via Redundant Information Coding): Η μερική καταστροφή ενός δικτύου οδηγεί στην υποβάθμιση της απόδοσης του. Ωστόσο, μερικές από τις ικανότητες του δικτύου μπορούν να διατηρηθούν ακόμη και μετά από μια σημαντική ζημία.

1.4 Νευρωνικά δίκτυα & συμβατικοί υπολογιστές

Τα νευρωνικά δίκτυα υιοθετούν μια διαφορετική μέθοδο επίλυσης των προβλημάτων από αυτή των συμβατικών υπολογιστών. Οι συμβατικοί υπολογιστές χρησιμοποιούν μια αλγοριθμική προσέγγιση δηλαδή, ακολουθούν ένα σύνολο οδηγιών προκειμένου να λύσουν ένα πρόβλημα. Όταν περιορίζεται η δυνατότητα επίλυσης προβλημάτων από τους συμβατικούς υπολογιστές , τα νευρωνικά δίκτυα είναι ικανά να δώσουν τη λύση. Αλλά οι υπολογιστές θα ήταν πολύ πιο χρήσιμοι εάν θα μπορούσαν να μας βοηθήσουν στην επίλυση προβλημάτων που δεν ξέρουμε πως ακριβώς να τα αντιμετωπίσουμε και ως εκ τούτου δεν μπορούμε να εκφράσουμε την λύση τους σε μορφή αλγορίθμου. Οι νεύρωνες των δικτύων λειτουργούν με παρόμοιο τρόπο όπως ο ανθρώπινος εγκέφαλος. Το δίκτυο αποτελείται από έναν μεγάλο αριθμό επεξεργαστικών στοιχείων που λειτουργούν παράλληλα για να λύσουν ένα συγκεκριμένο πρόβλημα. Τα νευρωνικά δίκτυα μαθαίνουν από παραδείγματα. Δεν μπορούν να προγραμματιστούν για να εκτελέσουν έναν συγκεκριμένο έργο. Τα παραδείγματα

πρέπει να έχουν επιλεχθεί προσεκτικά ειδάλλως χάνεται χρήσιμος χρόνος ή ακόμα χειρότερα το δίκτυο λειτουργεί λάθος. Το μειονέκτημα είναι ότι επειδή το δίκτυο βρίσκει πώς να λύσει το πρόβλημα μόνο του, η λειτουργία του μπορεί να είναι απρόβλεπτη.

Από την άλλη μεριά, οι συμβατικοί υπολογιστές χρησιμοποιούν μια γνωστική προσέγγιση στην επίλυση προβλήματος. Ο τρόπος με τον οποίο θα λυθεί το πρόβλημα πρέπει να είναι γνωστός και δηλωμένος σε μικρές σαφείς οδηγίες. Αυτές οι οδηγίες μετατρέπονται σε μια υψηλού επιπέδου γλώσσα προγραμματισμού και έπειτα σε κώδικα μηχανής που ο υπολογιστής μπορεί να καταλάβει. Αυτές οι μηχανές είναι εντελώς προβλέψιμες. Εάν κάτι πηγαίνει στραβά οφείλεται σε ελάττωμα είτε του λογισμικού είτε του υλικού.

Τα νευρωνικά δίκτυα και οι συμβατικοί αλγοριθμικοί υπολογιστές δεν είναι ανταγωνιστές αλλά συμπληρώνουν το ένα το άλλο. Υπάρχουν προβλήματα πιο κατάλληλα για αλγοριθμική προσέγγιση και προβλήματα που ταιριάζουν πιο πολύ στα νευρωνικά δίκτυα. Ακόμη, ένας μεγάλος αριθμός προβλημάτων, απαιτούν συστήματα που χρησιμοποιούν συνδυασμό των δύο προσεγγίσεων (συνήθως ένας συμβατικός υπολογιστής χρησιμοποιείται για να επιβλέπει το νευρωνικό δίκτυο) με σκοπό να λειτουργεί με μεγάλη ακρίβεια.

Τα νευρωνικά δίκτυα δεν κάνουν θαύματα. Αλλά εάν χρησιμοποιούνται λογικά μπορούν να παραγάγουν καταπληκτικά αποτελέσματα.

1.5 Νευρωνικά δίκτυα στην πράξη

Τα νευρωνικά δίκτυα έχουν ευρεία δυνατότητα εφαρμογής στα πραγματικά προβλήματα επιχειρήσεων παγκοσμίως. Ενώ, έχουν ήδη εφαρμοστεί επιτυχώς σε πολλές βιομηχανίες. Δεδομένου ότι τα νευρωνικά δίκτυα είναι τα καλύτερα για τον προσδιορισμό των τύπων ή των τάσεων στα δεδομένα, είναι προσαρμοσμένα στις ανάγκες πρόβλεψης συμπεριλαμβανομένων και των παρακάτω:

- ☐ πρόβλεψη πωλήσεων
- ☐ έλεγχος βιομηχανικής διαδικασίας
- ☐ έρευνα πελατών
- ☐ επικύρωση στοιχείων
- ☐ διαχείριση κινδύνου
- ☐ προώθηση μάρκετινγκ

Εκτός από τις επιχειρήσεις χρησιμοποιούνται με μεγάλη επιτυχία και σε άλλα πεδία της επιστήμης και της τεχνολογίας:

Αεροπορική Βιομηχανία - Με υψηλή απόδοση στους αυτόματους πιλότους αεροσκαφών, στην προσομοίωση διαδρομής πτήσης, στα συστήματα ελέγχου των αεροσκαφών.
Αυτοκινητοβιομηχανία - Χρησιμοποιούνται στα συστήματα αυτόματης καθοδήγησης στην αυτοκινητοβιομηχανία.
Τραπεζική λειτουργία – Έλεγχος και ανάγνωση εγγράφων καθώς επίσης εκτίμηση των εφαρμογών πίστωσης.
Άμυνα – Ανίχνευση στόχου, διάκριση αντικειμένων, αναγνώριση προσώπου, νέα είδη αισθητήρων, σόναρ, διαδικασία σημάτων εικόνας και ραντάρ συμπεριλαμβανομένων και της κατανόησης των δεδομένων, εξαγωγή χαρακτηριστικών και καταστολή θορύβου, αναγνώριση σημάτων και εικόνας.
Ηλεκτρονικά – Πρόβλεψη της αλληλουχίας του κώδικα, σχέδιο ολοκλήρωσης της περιφέρειας του τσίπ, έλεγχος διαδικασίας, αποτυχία ανάλυσης του τσίπ, όραμα μηχανής, σύνθεση φωνής, μη γραμμική μοντελοποίηση.
Διασκέδαση – Κινούμενα σχέδια, ειδικά εφέ, πρόβλεψη αγοράς.
Οικονομικά – Αποτίμηση πραγματικής ιδιοκτησίας, σύμβουλος δανείων, διαχωρισμός υποθηκών, εκτίμηση των δεσμών συνεργασίας, χρησιμοποίηση της ανάλυσης του ορίου πίστωσης, πρόγραμμα συναλλαγής πορτοφολιού, πρόβλεψη της τρέχουσας τιμής.
Ασφάλιση – Πολιτική εφαρμογής της αποτίμησης, βελτιστοποίηση προϊόντος.
Κατασκευή – Έλεγχος της διαδικασίας της κατασκευής, ανάλυση και σχεδιασμός του προϊόντος, διάγνωση της διαδικασίας και της μηχανής, καθορισμός μορίων πραγματικού χρόνου, συστήματα επιθεώρησης οπτικής ποιότητας, έλεγχος μπύρας, πρόβλεψη ποιότητας χαρτιού, ανάλυση συντήρησης των μηχανών, σχεδιασμός και διοίκηση, δυναμική μοντελοποίηση και συστήματα χημικής διαδικασίας.
Λάδι και Αέριο – Εξερεύνηση.
Ρομποτική – Έλεγχος τροχιάς, οπτικά συστήματα κ.α.
Ασφάλεια – Ανάλυση αγοράς, αυτόματη αποτίμηση ασφάλειας.
Τηλεπικοινωνίες – Κατανόηση δεδομένων και εικόνας, υπηρεσίες αυτοματοποιημένης πληροφορίας, πραγματικού χρόνου μεταφράσεις της καθομιλουμένης, συστήματα διαδικασίας πληρωμής πελατών.
Μεταφορικά μέσα – Συστήματα διάγνωσης των φρένων των φορτηγών, προγραμματισμός οχημάτων. (Hugan M.T.,Demuth H.B., 1999, σελ. 1642-1656), (Murray R., Sbarbaro D., 1992, σελ. 402-409)

Τα ΤΝΔ χρησιμοποιούνται επίσης: στην αναγνώριση των ομιλητών στις επικοινωνίες, στη διάγνωση της ηπατίτιδας, στην αποκατάσταση των τηλεπικοινωνιών από ελαττωματικό λογισμικό, στην ερμηνεία των κινεζικών λέξεων, στην υποθαλάσσια ανίχνευση ορυχείων, στην ανάλυση σύστασης, στη τρισδιάστατη αναγνώριση αντικειμένου, στη χειρόγραφη

αναγνώριση λέξης και στην αναγνώριση προσώπου. (www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html)

1.6 Νευρωνικά Δίκτυα στην επιχείρηση

Η επιχείρηση είναι ένας τομέας με διάφορους γενικούς τομείς εξειδίκευσης όπως η λογιστική και η οικονομική ανάλυση. Σχεδόν οποιαδήποτε νευρωνική εφαρμογή δικτύων θα ταίριαζε σε ένα τομέα της επιχείρησης ή σε μια οικονομική ανάλυση. Υπάρχει δυνατότητα για τη χρησιμοποίηση των νευρωνικών δικτύων για επιχειρησιακούς λόγους, συμπεριλαμβανομένης της κατανομής των πόρων και του σχεδιασμού.

1.6.1 Μάρκετινγκ

Υπάρχει μια εφαρμογή μάρκετινγκ που έχει ενσωματωθεί με ένα νευρωνικό σύστημα δικτύων. Η Αερογραμμή Μάρκετινγκ Tactician (ένα εμπορικό σήμα που συμβολίζεται με τα αρχικά AMT) είναι ένα συγκρότημα ηλεκτρονικών υπολογιστών που αποτελείται από τις διάφορες ευφυείς τεχνολογίες συμπεριλαμβανομένων των έμπειρων συστημάτων. Ένα προς τα εμπρός (feedforward) νευρωνικό δίκτυο είναι ενσωματωμένο στο AMT και εκπαιδεύθηκε χρησιμοποιώντας τον αλγόριθμο back-propagation για να βοηθήσει τον έλεγχο μάρκετινγκ της κατανομής των καθισμάτων των αερογραμμών. Επιπροσθέτως, το περιβάλλον της εφαρμογής άλλαζε γρήγορα και συνεχώς και απαιτούσε μια συνεχώς προσαρμοστική λύση. Το σύστημα χρησιμοποιείται για να ελέγξει και να δώσει τις συμβουλές για κάθε αναχώρηση. Τέτοιες πληροφορίες ασκούν άμεση επίδραση στην αποδοτικότητα μιας αερογραμμής και μπορούν να παρέχουν ένα τεχνολογικό πλεονέκτημα για τους χρήστες του συστήματος (Hutchison & Stephens, 1987).

Ενώ είναι σημαντικό ότι τα νευρωνικά δίκτυα έχουν εφαρμοστεί σε αυτό το πρόβλημα, είναι επίσης σημαντικό να φανεί ότι αυτή η ευφυής τεχνολογία μπορεί να ενσωματωθεί με τα έμπειρα συστήματα και άλλες προσεγγίσεις για να κάνει ένα σύστημα λειτουργικό. Τα νευρωνικά δίκτυα χρησιμοποιήθηκαν για να ανακαλύψουν την επιρροή των απροσδιόριστων αλληλεπιδράσεων από τις διαφορετικές αλληλεπιδράσεις των διάφορων μεταβλητών. Ενώ αυτές οι αλληλεπιδράσεις δεν καθορίστηκαν, χρησιμοποιήθηκαν από το νευρωνικό σύστημα για να αναπτύξουν χρήσιμα συμπεράσματα.

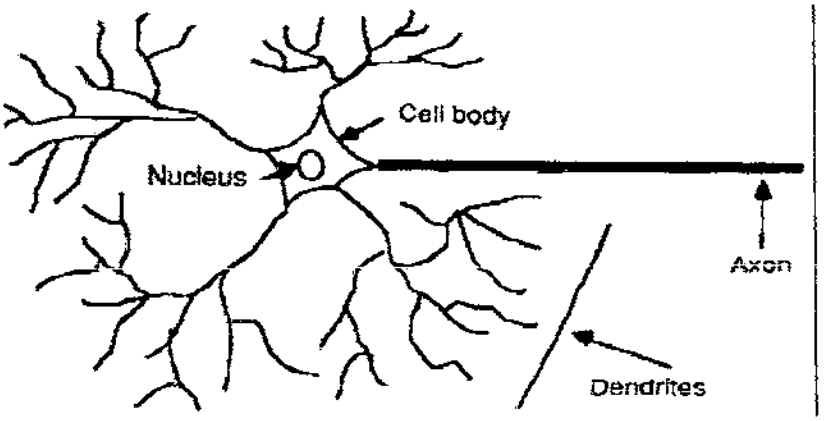
1.6.2 Πιστωτική αξιολόγηση

Η επιχείρηση HNC, που ιδρύθηκε από Robert Hecht-Nielsen, έχει αναπτύξει διάφορες εφαρμογές νευρωνικών δικτύων. Μια από αυτές είναι το πιστωτικό σύστημα που αυξάνει την αποδοτικότητα του υπάρχοντος προτύπου μέχρι 27%. Τα νευρωνικά συστήματα HNC εφαρμόστηκαν επίσης στη διαλογή υποθηκών. Ένα νευρωνικό αυτοματοποιημένο δίκτυο για την ασφάλεια των υποθηκών αναπτύχθηκε από την Εταιρία Nestor. Αυτό το σύστημα εκπαιδεύθηκε με 5048 εφαρμογές των οποίων οι 2597 πιστοποιήθηκαν. Τα στοιχεία αφορούσαν την ιδιοκτησία και τα προσόντα των οφειλετών. Με ένα συντηρητικό τρόπο το σύστημα συμφωνεί σχετικά με τους αντασφαλιστές σε ποσοστό 97% των περιπτώσεων. Στο φιλελεύθερο πρότυπο το σύστημα συμφώνησε με 84% των περιπτώσεων. Αυτό το σύστημα εφαρμόστηκε στο Apollo DN3000 και χρησιμοποίησε 250 Kbytes μνήμης καθώς αυτοματοποίησε τη διαδικασία μιας υπόθεσης φακέλου σε περίπου 1 λεπτό.

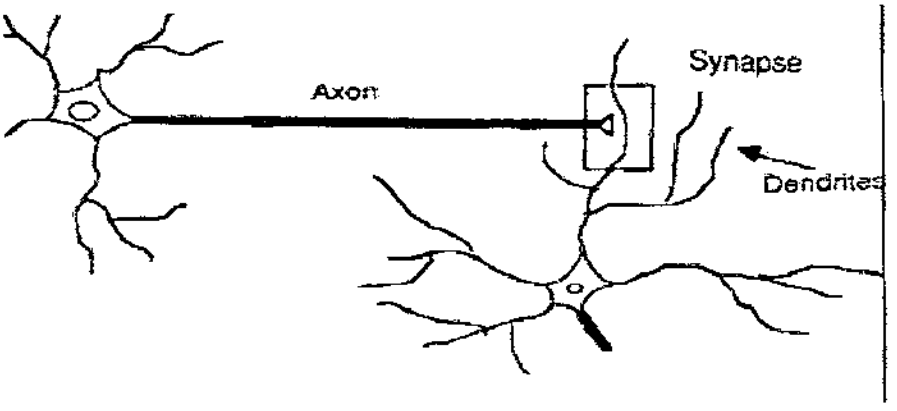
2. Ανθρώπινοι και τεχνητοί νευρώνες

2.1 Πώς μαθαίνει ο ανθρώπινος εγκέφαλος;

Πολλά είναι ακόμα άγνωστα για το πώς ο ανθρώπινος εγκέφαλος αυτοεκπαιδεύεται για να επεξεργαστεί πληροφορίες. Στον ανθρώπινο εγκέφαλο, ένας τυπικός νευρώνας συλλέγει τα σήματα από άλλους νευρώνες μέσω ενός πλήθους λεπτών δομών που αποκαλούνται *δενδρίτες*. Ο νευρώνας στέλνει τα σήματα της ηλεκτρικής δραστηριότητας μέσω ενός μακριού, λεπτού άξονα, ο οποίος χωρίζεται σε χιλιάδες κλάδους. Στο τέλος κάθε κλάδου, μια δομή αποκαλούμενη *σύναψη* μετατρέπει τη δραστηριότητα/ ενέργεια σε ηλεκτρική επίδραση που εμποδίζει ή διεγείρει την δραστηριότητα των συνδεδεμένων νευρώνων. Όταν ένας νευρώνας δέχεται μια είσοδο διέγερσης που είναι αρκετά μεγάλη σε σύγκριση με μια ανασταλτική είσοδο, στέλνει ένα σήμα ηλεκτρικής δραστηριότητας στον άξονα του. Η μάθηση επιτυγχάνεται με την αλλαγή της αποτελεσματικότητας των συνάψεων έτσι ώστε η επιρροή ενός νευρώνα σε άλλο να αλλάζει.



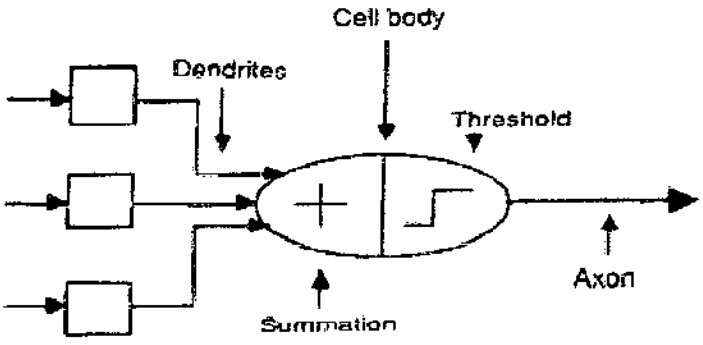
Σχήμα 1: Συστατικά μέρη ενός νευρώνα
Πηγή: www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html



Σχήμα 2: Η σύναψη
Πηγή: www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

2.2 Από τους ανθρώπινους νευρώνες στους τεχνητούς νευρώνες

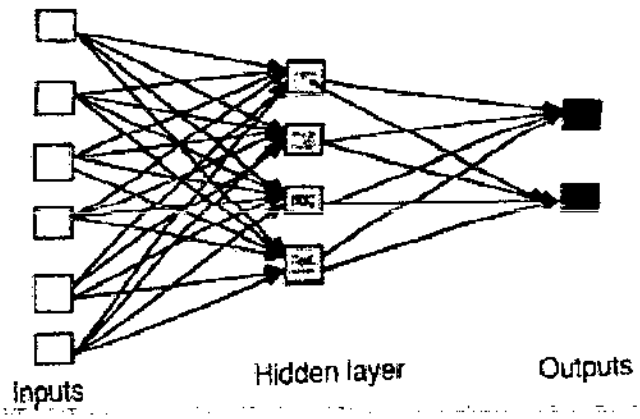
Οι άνθρωποι διευθύνουν αυτά τα νευρικά δίκτυα αφού πρώτα αποφασίσουν τα απαραίτητα χαρακτηριστικά γνωρίσματα που θα πρέπει να διαθέσουν οι νεύρωνες και οι διασυνδέσεις τους. Στη συνέχεια προγραμματίζουν ένα υπολογιστή για να μιμηθεί αυτά τα χαρακτηριστικά γνωρίσματα. Ωστόσο επειδή η γνώση γύρω από την ακριβή λειτουργία των βιολογικών νευρώνων είναι ελλιπής και η υπολογιστική ισχύ που διαθέτουμε είναι ακόμα περιορισμένη, τα χρησιμοποιούμενα πρότυπά είναι απαραίτητα, πιο εξιδανικευμένα από αυτά των αληθινών νευρωνικών δικτύων. Παρακάτω βλέπουμε ένα νευρώνα με τρεις εισόδους των οποίων οι διακλαδισμένες αποφύσεις αθροίζονται έπειτα οργανώνονται σε μια βασική μονάδα που ονομάζεται σώμα κατωφλίου και κατόπιν οδηγούνται μέσω ενός νευράξονα στην έξοδο.



Σχήμα 3: Το μοντέλο του νευρώνα
Πηγή: www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

3.1 Προς τα εμπρός δίκτυα (Feed Forward)

Τα Feed- forward ΤΝΔ (Σχήμα 4) επιτρέπουν στα σήματα να ταξιδεύουν μόνο προς μια κατεύθυνση, από την είσοδο προς την έξοδο. Δεν υπάρχει δηλαδή καμία ανατροφοδότηση. Η έξοδος, οποιουδήποτε επιπέδου, δεν επιδρά στο ίδιο ή σε προηγούμενο επίπεδο. Τα Feed-forward ΤΝΔ είναι απλά δίκτυα που συσχετίζουν τις εισόδους με τις εξόδους. Χρησιμοποιούνται υπερβολικά στην αναγνώριση προτύπων. Αυτός ο τύπος οργάνωσης αναφέρεται επίσης σαν από κάτω προς τα επάνω ή από επάνω προς τα κάτω. (www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html)



Σχήμα 4: Παράδειγμα ενός απλού προς τα εμπρός δικτύου
Πηγή: www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

3.2 Δίκτυα ανατροφοδότησης (Feedback)

Τα δίκτυα ανατροφοδότησης επιτρέπουν στα σήματα να "ταξιδεύουν" και προς τις δύο κατευθύνσεις εισάγοντας βρόχους στο δίκτυο. Τα δίκτυα ανατροφοδότησης είναι πολύ ισχυρά και μπορούν να γίνουν εξαιρετικά περίπλοκα. Είναι δυναμικά και βασίζονται στη συνεχόμενη αλλαγή έως ότου φθάσουν σε ένα σημείο ισορροπίας.

Παραμένουν στο σημείο ισορροπίας μέχρι που η είσοδος αλλάζει και πρέπει να βρεθεί μια νέα ισορροπία. Η δομή ανατροφοδότησης αναφέρεται επίσης ως αλληλεπίδραση ή επαναλαμβανόμενη, αν και ο τελευταίος όρος χρησιμοποιείται συχνά για να δείξει τις συνδέσεις ανατροφοδότησης σε οργάνωση ενός επιπέδου.

3.3 Στρώματα δικτύων

Ο πιο κοινός και ταυτόχρονα ο πιο απλός τύπος τεχνητού νευρωνικού δικτύου αποτελείται από τρία επίπεδα. Συγκεκριμένα: ένα επίπεδο των μονάδων "**εισόδου** " συνδέεται με ένα επίπεδο "**κρυμμένων** " μονάδων, το οποίο συνδέεται με τη σειρά του με ένα επίπεδο μονάδων "**εξόδου** " (Σχήμα 4)

- Η δραστηριότητα των μονάδων εισόδου περιορίζεται στο να δέχεται τις ακατέργαστες πληροφορίες που τροφοδοτούν το δίκτυο.

- Η δραστηριότητα κάθε κρυμμένης μονάδας καθορίζεται από τις μονάδες εισόδου και τα βάρη των συνδέσεων που υπάρχουν μεταξύ της εισόδου και των κρυμμένων μονάδων.

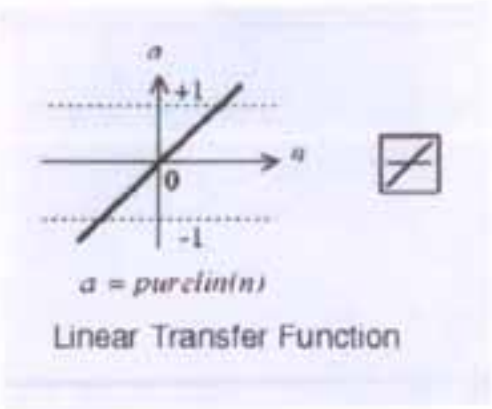
- Η συμπεριφορά των μονάδων εξόδου εξαρτάται από τη δραστηριότητα των κρυμμένων μονάδων εξόδου των βαρών μεταξύ των κρυμμένων μονάδων, των μονάδων εξόδου και τη σχέση εισόδου/ εξόδου που υπακούει η μονάδα εξόδου η οποία είναι γνωστή σαν συνάρτηση μεταφοράς και αναλύεται στην παράγραφο που ακολουθεί.

3.4 Συνάρτηση μεταφοράς

Η συμπεριφορά ενός ΤΝΔ (Τεχνητού Νευρικού Δικτύου) εξαρτάται πολύ από τις τιμές των βαρών και από την σχέση εισόδου-εξόδου (συνάρτηση μεταφοράς) που διέπουν τις μονάδες. Αυτή η σχέση, συνήθως έχει μια από τις ακόλουθες μορφές:

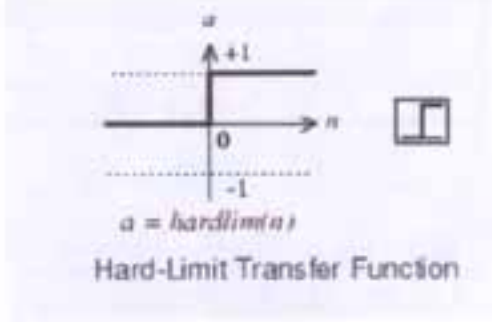
- γραμμική (ή κεκλιμένη ράμπα)
- περιοριστή κατωφλίου
- σιγμοειδής (sigmoid)

Η **γραμμική συνάρτηση μεταφοράς**, εξασφαλίζει ότι η τιμή εξόδου είναι ανάλογη της τιμής της εισόδου της.



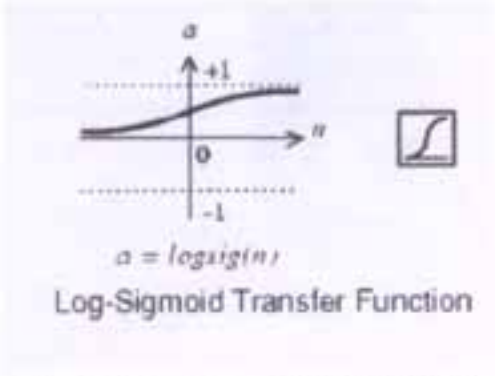
Σχήμα 5α: Γραμμική συνάρτηση μεταφοράς

Στον **περιοριστή κατωφλίου**, η έξοδος τίθεται σ' ένα από δύο επίπεδα, ανάλογα με το εάν η συνολική είσοδος είναι μεγαλύτερη ή μικρότερη της τιμής κατωφλίου.



Σχήμα 5β: Συνάρτηση μεταφοράς ενός περιοριστή κατωφλίου

Η **σιγμοειδής συνάρτηση μεταφοράς**, εξαναγκάζει την έξοδο να μεταβάλλεται μη γραμμικά καθώς αλλάζει η είσοδος. Οι σιγμοειδείς μονάδες μοιάζουν περισσότερο στη συνάρτηση μεταφοράς των πραγματικών νευρώνων από ότι οι γραμμικές ή οι περιοριστές κατωφλίου, αλλά και οι τρεις πρέπει να θεωρηθούν προσεγγίσεις, των συναρτήσεων μεταφοράς των πραγματικών νευρώνων.



Σχήμα 5γ: *Σιγμοειδής συνάρτηση μεταφοράς*

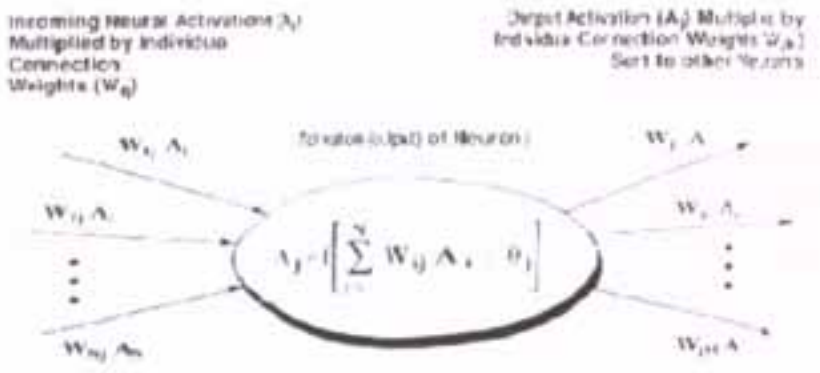
Για να δημιουργήσουμε ένα νευρωνικό δίκτυο που εκτελεί κάποια συγκεκριμένη εργασία, πρέπει να επιλέξουμε τον τρόπο που οι μονάδες συνδέονται η μία με την άλλη και πρέπει να δώσουμε κατάλληλες τιμές στα βάρη των συνδέσεων του δικτύου. Οι συνδέσεις καθορίζουν εάν είναι δυνατό μια μονάδα να επηρεάσει την άλλη. Τα βάρη καθορίζουν το μέγεθος της επιρροής.

Μπορούμε να εκπαιδεύσουμε ένα δίκτυο τριών-επιπέδων για να εκτελέσει μια συγκεκριμένη εργασία με τη χρήση της ακόλουθης διαδικασίας:

1. Τροφοδοτούμε το δίκτυο με χαρακτηριστικά παραδείγματα, τα οποία αποτελούνται από πρότυπα εισόδων μαζί με το επιθυμητό πρότυπο- μέρος των μονάδων εξόδου του δικτύου.
2. Υπολογίζουμε κατά πόσο η πραγματική έξοδος του δικτύου ταιριάζει με την επιθυμητή έξοδο.
3. Αλλάζουμε το βάρος κάθε σύνδεσης έτσι ώστε το δίκτυο να δίνει μια καλύτερη προσέγγιση της επιθυμητής εξόδου.

3.5 Οι τεχνικές μάθησης

Κάθε γνώση νευρωνικών δικτύων περιλαμβάνεται τις τιμές των συνδέσεων των βαρών. Η τροποποίηση της γνώσης που αποθηκεύεται στο δίκτυο ως λειτουργία της εμπειρίας υπονοεί έναν κανόνα εκμάθησης για τις τιμές των βαρών.



Σχήμα 6: *Είσοδοι και έξοδοι ενός νευρώνα*
Πηγή: *www.doc.ic.ac.uk ~nd surprise_96 journal vol4 cs11 report.html*

Οι πληροφορίες αποθηκεύονται στο πίνακα W που περιέχει τα βάρη ενός νευρωνικού δικτύου. Η μάθηση ουσιαστικά αποτελεί την διαδικασία προσδιορισμού των τιμών των βαρών. Ανάλογα με τον τρόπο που υλοποιείτε η διαδικασία μάθησης, μπορούμε να διακρίνουμε δύο σημαντικές κατηγορίες νευρωνικών δικτύων:

- **τα σταθερά δίκτυα** στα οποία τα βάρη δεν μπορούν να αλλάξουν, δηλαδή $dW / dt = 0$. Σε τέτοια δίκτυα, τα βάρη καθορίζονται από την αρχή ανάλογα με το πρόβλημα που λύνουν.
- **τα προσαρμοστικά δίκτυα** που είναι σε θέση να αλλάξουν τα βάρη τους, δηλαδή $dW/dt \neq 0$.

Όλες οι μέθοδοι μάθησης που χρησιμοποιούνται για τα **προσαρμοστικά νευρωνικά δίκτυα** μπορούν να ταξινομηθούν σε δύο σημαντικές κατηγορίες:

- **Την Εποπτευόμενη μάθηση** η οποία ενσωματώνει έναν εξωτερικό δάσκαλο, έτσι ώστε κάθε μονάδα εξόδου να γνωρίζει ποια είναι η επιθυμητή απάντησή στα σήματα εισόδου. Κατά τη διάρκεια της διαδικασίας μάθησης μπορεί να απαιτούνται σφαιρικές πληροφορίες. Τα παραδείγματα της εποπτευμένης εκμάθησης περιλαμβάνουν την λάθος-διόρθωση, την ενισχυτική μάθηση και την πιθανολογική μάθηση.

  Ένα σημαντικό ζήτημα σχετικά με την εποπτευμένη μάθηση είναι το πρόβλημα της σύγκλισης λάθους, δηλαδή η ελαχιστοποίηση του σφάλματος μεταξύ των επιθυμητών και υπολογισμένων τιμών των μονάδων. Ο στόχος είναι να καθοριστεί ένα σύνολο βαρών που να ελαχιστοποιεί το σφάλμα. Μια γνωστή μέθοδος, που είναι κοινή για πολλά παραδείγματα μάθησης είναι ο αλγόριθμος LMS.
- **Μη εποπτευόμενη μάθηση** που δεν χρησιμοποιεί κανέναν εξωτερικό δάσκαλο και είναι βασισμένη μόνο στις τοπικές πληροφορίες. Αναφέρεται επίσης ως αυτοοργάνωση, υπό την έννοια ότι το ΤΝΔ αυτοοργανώνει τα στοιχεία που παρουσιάζονται στο δίκτυο και ανιχνεύει τις προκύπτουσες συλλογικές ιδιότητές τους. Παράδειγμα τέτοιας μάθησης είναι η μάθηση Hebbian και η ανταγωνιστική χωρίς επίβλεψη εκμάθηση.

  Από τους ανθρώπινους νευρώνες στην πτυχή της τεχνητής εκμάθησης Neuronesther που αφορά τη διάκριση ή όχι μιας ξεχωριστής φάσης, κατά τη διάρκεια της οποίας το δίκτυο εκπαιδεύεται, για μια επόμενη φάση της λειτουργίας. Λέμε ότι ένα νευρωνικό δίκτυο μαθαίνει off –line εάν η φάση μάθησης και η φάση λειτουργίας είναι δύο ευδιάκριτες και ξεχωριστές φάσεις. Ένα νευρωνικό δίκτυο μαθαίνει on-line εάν μαθαίνει και λειτουργεί συγχρόνως. Συνήθως, η εποπτευόμενη μάθηση εκτελείται off-line ενώ η μάθηση χωρίς επίβλεψη εκτελείται on-line.

3.6 Παράδειγμα επεξήγησης της διαδικασίας εκπαίδευσης στην αναγνώριση χειρόγραφων ψηφίων

Μπορούμε να χρησιμοποιήσουμε χειρόγραφα ψηφία 0,1,...,9, 256 αισθητήρες σε διάταξη τετραγώνου, καθένας από τους οποίους καταγράφει την παρουσία ή την απουσία του μελανιού σε μια μικρή περιοχή ενός ενιαίου ψηφίου. Το δίκτυο επομένως θα χρειαζόταν 256 μονάδες εισόδου (μια για κάθε αισθητήρα), 10 μονάδες εξόδου (μια για κάθε ψηφίο) και έναν αριθμό από κρυμμένες μονάδες.

Για κάθε είδος ψηφίου που καταγράφεται από τους αισθητήρες, το δίκτυο θα πρέπει να δημιουργεί υψηλή δραστηριότητα στην κατάλληλη μονάδα εξόδου και χαμηλή δραστηριότητα στις άλλες μονάδες εξόδου.

Για να εκπαιδεύσουμε το δίκτυο, παρουσιάζουμε μια εικόνα ενός ψηφίου και συγκρίνουμε την πραγματική δραστηριότητα των 10 μονάδων εξόδου με την επιθυμητή δραστηριότητα. Υπολογίζουμε έπειτα το σφάλμα, το οποίο ορίζεται ως το τετράγωνο της διαφοράς μεταξύ των πραγματικών και επιθυμητών δραστηριοτήτων. Έπειτα αλλάζουμε το βάρος κάθε σύνδεσης ώστε να μειωθεί το σφάλμα. Επαναλαμβάνουμε αυτήν την διαδικασία εκπαίδευσης για πολλές διαφορετικές εικόνες για κάθε μια διαφορετική εικόνα του κάθε είδους ψηφίου μέχρι να ταξινομήσει το δίκτυο κάθε εικόνα σωστά.

Για να εφαρμόσουμε αυτήν την διαδικασία πρέπει να υπολογίσουμε την παράγωγο του σφάλματος ως προς τα βάρη των συνδέσεων του δικτύου προκειμένου να αλλάξει η τιμή του βάρους κατά ένα ποσό που είναι ανάλογο προς το ρυθμό αλλαγής του σφάλματος καθώς αλλάζει το βάρος. Ένας τρόπος να υπολογιστεί το σφάλμα βαρών είναι να διαταραχτεί ελαφρώς ένα βάρος και να παρατηρηθεί πώς αλλάζει το σφάλμα. Αλλά αυτή η μέθοδος είναι ανεπαρκής επειδή απαιτεί μια χωριστή διαταραχή για κάθε ένα από τα πολλά βάρη που συνήθως υπάρχουν σ'ένα δίκτυο.

Ένας άλλος τρόπος να υπολογιστεί το σφάλμα βαρών είναι να χρησιμοποιηθεί ο αλγόριθμος back-propagation που περιγράφεται πιο κάτω, και έχει γίνει σήμερα ένα από τα σημαντικότερα εργαλεία για την εκπαίδευση νευρωνικών δικτύων. Αναπτύχθηκε ανεξάρτητα από δύο ομάδες, μια στη Γαλλία (Fogelman- Soulie, Gallinari και LE Cun), και μια άλλη στις ΗΠΑ (Rumelhart, Hinton και Williams).

3.7 Ρυθμός Μάθησης

Πριν προχωρήσουμε στο ρυθμό μάθησης θα αναφέρουμε τον ***κανόνα μάθησης Hebbian*** *(Hebb,1949)*, σύμφωνα με τον οποίο αν δύο μονάδες j και i είναι ενεργές ταυτόχρονα τότε η μεταξύ τους διασύνδεση θα πρέπει να ενδυναμωθεί. Αν η j δέχεται είσοδο από την i η πιο απλή έκδοση του κανόνα αυτού συνιστά τη μετατροπή του βάρους w_{ji} με βάση τον ακόλουθο κανόνα:

$$\Delta w_{ji} = \eta y_j y_i \quad (3.1)$$

όπου η είναι μια θετική σταθερά που καθορίζει το ρυθμό μάθησης (learning rate) του δικτύου. Ένας άλλος κλασικός κανόνας χρησιμοποιεί όχι αυτή καθαυτή την ενεργοποίηση της μονάδας i αλλά τη διαφορά της πραγματικής από την επιθυμητή ενεργοποίηση για την προσαρμογή των βαρών:

$$\Delta w_{ji} = \eta y_j (d_i - yi) \quad (3.2)$$

όπου d_k είναι η επιθυμητή ενεργοποίηση η οποία δίνεται από κάποιο "δάσκαλο".

Η διαδικασία μάθησης απαιτεί η αλλαγή στο βάρος w να είναι ανάλογη της $\partial E^p / \partial w$ και τα βήματα να είναι πολύ μικρά. Άρα αυτό που ονομάζεται ρυθμός μάθησης είναι ουσιαστικά η σταθερά της αναλογικότητας.

Όσο μικρότερη είναι η παράμετρος της μάθηση η τόσο μικρότερες θα είναι οι αλλαγές στα βάρη, αλλά τότε η μάθηση θα κρατάει περισσότερο. Εάν η είναι πάρα πολύ μεγάλη, η διαδικασία της μάθηση μπορεί να γίνει ασταθής. Σύμφωνα με τον Sorensen (1994) είναι συνηθισμένη πρακτική να μειώνεται βαθμιαία η τιμή η από μια αρχική τιμή 0.1 σε 0. 001.

Ο αλγόριθμος back – propagation μπορεί να φθάσει μόνο σε ένα τοπικό ελάχιστο της εξίσωσης του σφάλματος και η αναζήτηση μπορεί να είναι αργή, ειδικά κοντά στο ελάχιστο. Ένας απλός τρόπος να βελτιωθεί η κατάσταση και να αποφευχθεί η αστάθεια, είναι να τροποποιηθεί ο κανόνας Δέλτα με την εισαγωγή ενός όρου *ορμής* στην εξίσωση μάθησης, ως ακολούθως:

$$\Delta w_{ji} = \eta \delta y_i + a \Delta w_{ji} \quad (3.3)$$

Εδώ το a είναι συνήθως μια θετική σταθερά αποκαλούμενη *σταθερά ορμής*. Η επίδραση είναι ένα φιλτράρισμα χαμηλών περασμάτων (low pass) των αυξήσεων των εκσυγχρονισμών του βάρους, μειώνοντας κατά συνέπεια τον κίνδυνο που υπάρχει σε ένα τοπικό ελάχιστο. Ακολουθούν κάποιοι εμπειροτεχνικοί κανόνες:

Για να είναι η μάθηση σταθερή, η σταθερά ορμής πρέπει να είναι μεταξύ

$$|a| \leq 1. \quad (3.4)$$

Όταν α είναι μηδέν ο back-propagation αλγόριθμος λειτουργεί χωρίς ορμή. Σημειώστε ότι το α μπορεί να είναι αρνητικό, αν και είναι απίθανο να χρησιμοποιηθεί στην πράξη. Σύμφωνα με τον Sorensen (1994) το α επιλέγεται τυπικά να είναι 0.2, 0.3, ..., 0.9.

Όταν οι μερικοί παράγωγοι $\partial\varepsilon / \partial w_{ji}$ της συνάρτησης σφάλματος έχουν το ίδιο πρόσημο στις διαδοχικές επαναλήψεις, το σύνολο Δw_{ji} αυξάνεται σε μέγεθος, και έτσι το βάρος παίρνει μια μεγάλη τιμή. Ως εκ τούτου ο όρος ορμή τείνει να επιταχύνει την κάθοδο.

Όταν οι μερικοί παράγωγοι $\partial\varepsilon / \partial w_{ji}$ έχουν αντίθετα πρόσημα στις διαδοχικές επαναλήψεις, το βάρος ρυθμίζεται σε μια μικρή τιμή. Ως εκ τούτου ο όρος ορμή έχει μια σταθερή μικρή επίδραση στις κατευθύνσεις που εναλλάσσονται στο πρόσημο.

Ο όρος ορμή μπορεί επίσης να αποτρέψει τον εγκλωβισμό κατά τη διαδικασία της μάθησης σε ένα ρηχό τοπικό ελάχιστο της επιφάνειας σφάλματος.

Η παράμετρος η της μάθησης είναι μια υποτιθέμενη σταθερά, αλλά στην πραγματικότητα πρέπει να είναι εξαρτώμενη με τις συνδέσεις. Μπορούμε στην πραγματικότητα να περιορίσουμε οποιοδήποτε αριθμό βαρών να παραμείνει σταθερό με το να κάνουμε απλά το ποσοστό μάθησης η_{ji} για βάρος w_{ji} να είναι ίσο με μηδέν.

Παρακάτω θα δούμε ένα παράδειγμα για το πόσο σημαντική ειναι η τιμή της παραμέτρου η στην επιτυχία ή αποτυχία του νευρωνικού δικτύου. Συγκεκριμένα θα δούμε πως θα χρησιμοποιήσουμε τις παρακάτω συναρτήσεις ώστε ένας γραμμικός νευρώνας να εκπαιδευτεί για να βρει τη βέλτιστη λύση σε ένα απλό πρόβλημα που χρησιμοποιείται ενας πολύ μεγάλος ρυθμός μάθησης. Οι συναρτήσεις που χρησιμοποιούμε είναι οι εξής:

Η συνάρτηση **NEWLIN** δημιουργεί ένα γραμμικό επίπεδο. Η σύνταξη της συνάρτησης αυτής στο περιβάλλοντος του MATLABείναι η εξής:

net = newlin(PR,S,ID,LR).

Η συνάρτηση NEWLIN δημιουργεί ένα νέο δίκτυο με ένα κουτί διαλόγου. Η συνάρτηση αυτή δέχεται ως ορίσματα εισόδου τα ακόλουθα:

PR - Rx2 Πίνακας των ελάχιστων και μέγιστων τιμών για τα στοιχεία εισόδου του R.

S - Ο αριθμός των στοιχείων στο διάνυσμα εξόδου.

ID – Διάνυσμα καθυστέρησης εισόδου (Input delay vector) με σφάλμα= [0].

LR – Ρυθμός μάθησης, με σφάλμα = 0.01;

Δίνοντας μας ένα νέο γραμμικό επίπεδο.

Η συνάρτηση NET = NEWLIN(PR,S,0,P). Η σύνταξη της NEWLIN δέχεται ως ορίσματα εισόδου:

P - Πίνακας των διανυσμάτων εισόδου, μας επιστρέφει ένα γραμμικό επίπεδο με το μέγιστο δυνατό ρυθμό μάθησης για εκπαίδευση με P εισόδους.

Η συνάρτηση **TRAIN** εκπαιδεύει ένα τεχνητό νευρωνικό δίκτυο. Η σύνταξη της συνάρτησης αυτής είναι η ακόλουθη:

[net,tr,Y,E,Pf,Af]=train(NET,P,T,Pi,Ai,VV,TV).

Η συνάρτηση TRAIN(NET,P,T,Pi,Ai) δέχεται ως ορίσματα εισόδου τα ακόλουθα:

NET – Δίκτυο.

P - Είσοδοι δικτύου.

T - Έξοδοι δικτύου, με μηδενικό σφάλμα.

Pi - Αρχική είσοδος καθυστέρησης με προϋπόθεση το μηδενικό σφάλμα.

Ai - Αρχικό επίπεδο καθυστέρησης με προϋπόθεση το μηδενικό σφάλμα.

VV - Δομή των διανυσμάτων επικύρωσης, με σφάλμα = [].

TV - Δομή των διανυσμάτων ελέγχου με σφάλμα = [].

Και επιστρέφει, ως ορίσματα εξόδου:

NET – Νέο δίκτυο.

TR - Εκπαιδευόμενο αρχείο (epoch and perf).

Y - Έξοδοι δικτύου

E - Σφάλματα δικτύου.

Pf - Προϋποθέσεις τελικών εισόδων καθυστέρησης.

Af - Προϋποθέσεις τελικών επιπέδων καθυστέρησης. (Τα ορίσματα μπορούν να έχουν τη μορφή πίνακα ή το διάνυσμα κελιών).

Η συνάρτηση **SIM** μιμείται ένα νευρωνικό δίκτυο. Τα ορίσματα που δέχεται είναι τα ακόλουθα:

T - Επιστρεφόμενο διάνυσμα χρόνου.

X - Επιστρεφόμενη έκφραση σε πίνακα.

Y - Επιστρεφόμενη έξοδο σε πίνακα.

Y1,...,Yn - Μπορούν να εκφραστούν μόνο για εμπόδια διαγραμμάτων.

'model' - Όνομα του μοντελοποιημένου διαγράμματος.

TIMESPAN: ένα από τα:

TFinal,

[TStart TFinal], or

[TStart OutputTimes TFinal].

OutputTimes είναι χρονικά σημεία που θα επιστρέψουν στο Τα, αλλά γενικά περιλαμβάνουν επιπρόσθετα χρονικά σημεία.

OPTIONS - Προαιρετικοί μιμητικοί παράμετροι.

UT - Προαιρετικοί εξωτερικοί είσοδοι. UT = [T, U1, ... Un]

Η συνάρτηση **MAXLINLR**, είναι ο μέγιστος ρυθμός μάθησης για ένα γραμμικό δίκτυο. Η σύνταξη της συνάρτησης αυτής είναι η ακόλουθη: lr = maxlinlr(P) lr = maxlinlr(P,'bias')
Η MAXLINLR(P) δέχεται ως διάνυσμα εισόδου ένα όρισμα,
P - RxQ πίνακας των διανυσμάτων εισόδου.
Και επιστρέφει ως όρισμα εξόδου το μέγιστο ρυθμό μάθησης για ένα γραμμικό επίπεδο χωρίς το bias το οποίο θα αποκτηθεί μόνο στα διανύσματα του P.
Ενώ η συνάρτηση MAXLINLR(P,'bias') επιστρέφει ως όρισμα εξόδου το μέγιστο ρυθμό μάθησης για ένα γραμμικό επίπεδο με bias.

Η συνάρτηση **ERRSURF** εκφράζει την επιφάνεια του σφάλματος ενός νευρώνα μιας εισόδου. Η σύνταξη της συνάρτησης αυτής είναι η ακόλουθη:
E = errsurf(P,T,WV,BV,F)
Η συνάρτηση ERRSURF(P,T,WV,BV,F) δέχεται τα εξής ορίσματα εισόδου:

P - 1xQ πίνακας διανυσμάτων εισόδου.

T - 1xQ πίνακας διανυσμάτων στόχου.

WV – Διάνυσμα γραμμής των τιμών της W.

BV- Διάνυσμα γραμμής των τιμών του B.

F - Συνάρτησης μεταφοράς (string).

Και επιστρέφει ως όρισμα εξόδου έναν πίνακα των τιμών του λάθους πάνω από το WV και το BV.

Η συνάρτηση **PLOTES** σχεδιάζει την επιφάνεια σφάλματος ενός νευρώνα μιας εισόδου. Η σύνταξη της συνάρτησης αυτής είναι η ακόλουθη:
plotes(wv,bv,es,v).
Η συνάρτηση PLOTES(WV,BV,ES,V) δέχεται τα εξής ως ορίσματα εισόδου:

WV - 1xN διάνυσμα γραμμής των τιμών του W.

BV - 1xM διάνυσμα γραμμής των τιμών του B.

ES - MxN πίνακας των διανυσμάτων των λαθών.

V - Σχέδιο, με σφάλμα = [-37.5, 30].

Ακόμη σχεδιάζει την επιφάνεια σφάλματος. Ενώ παράλληλα υπολογίζει την επιφάνεια σφάλματος ES με την συνάρτηση ERRSURF που προαναφέραμε.

Η συνάρτηση **GINPUT** αντιπροσωπεύει την γραφική είσοδο του ποντικιού.
Η συνάρτηση [X,Y] = GINPUT(N) παίρνει N σημεία από τους αρχικούς άξονες και επιστρέφει το X- και το Y-ισοδύναμα σε μήκος του N των διανυσμάτων X και Y. Ο κέρσορας μπορεί να τοποθετηθεί χρησιμοποιώντας το ποντίκι. Τα δεδομένα εισάγονται πατώντας το αριστερό κουμπί του ποντικιού ή οποιοδήποτε πλήκτρο του πληκτρολόγιου.

Η συνάρτηση [X,Y] = GINPUT συγκεντρώνει ένα απεριόριστο αριθμό σημείων μέχρι να πατηθεί το πλήκτρο επιστροφής.

Η συνάρτηση [X,Y,BUTTON] = GINPUT(N) επιστρέφει ένα τρίτο αποτέλεσμα, το BUTTON, που περιέχει ένα διάνυσμα ακέραιων καθορίζοντας ποιο κουμπί του ποντικιού (1,2,3 from left) χρησιμοποιήθηκε ή εάν τα πλήκτρα του πληκτρολογίου των αριθμών χρησιμοποιήθηκαν.

Η συνάρτηση **PLOTPERF**, σχεδιάζει την εκτέλεση ενός δικτύου. Η σύνταξη της συνάρτησης είναι η εξής: plotperf(tr,goal,name,epoch)

Η συνάρτηση PLOTPERF(TR,GOAL,NAME,EPOCH) δέχεται τα εξής ορίσματα εισόδου,

TR – Εκπαιδευόμενο αρχείο που επέστρεψε από την εκπαίδευση.

GOAL – Έξοδος εκπαίδευσης, με σφάλμα = NaN.

NAME – Όνομα της συνάρτησης εκπαίδευσης, με σφάλμα = ''.

EPOCH – Αριθμός των εποχών, με σφάλμα = το μήκος των εκπαιδευόμενων αρχείων.

Η παραπάνω συνάρτηση σχεδιάζει την διαδικασία εκπαίδευσης ενώ αν είναι απαραίτητο, μας δίνει την έξοδο της εκπαίδευσης, την επικύρωση της εκτέλεσης και τον έλεγχο της εκτέλεσης.

ΠΑΡΑΔΕΙΓΜΑ:

Ο νευρώνας εκπαιδεύεται με ρυθμό μάθησης μεγαλύτερο από αυτό που προτείνει η MAXLINLR.

Το P ορίζει δύο τύπους εισόδων με 1-στοιχείο (διανύσματα στηλών):

$$>> P = [+0.1 \; -1.2]; \quad (3.5)$$

Το T ορίζει τους συνδεδεμένους στόχους 1-στοιχείου (διανύσματα στηλών):

$$>> T = [+0.5 \; +1.0]; \quad (3.6)$$

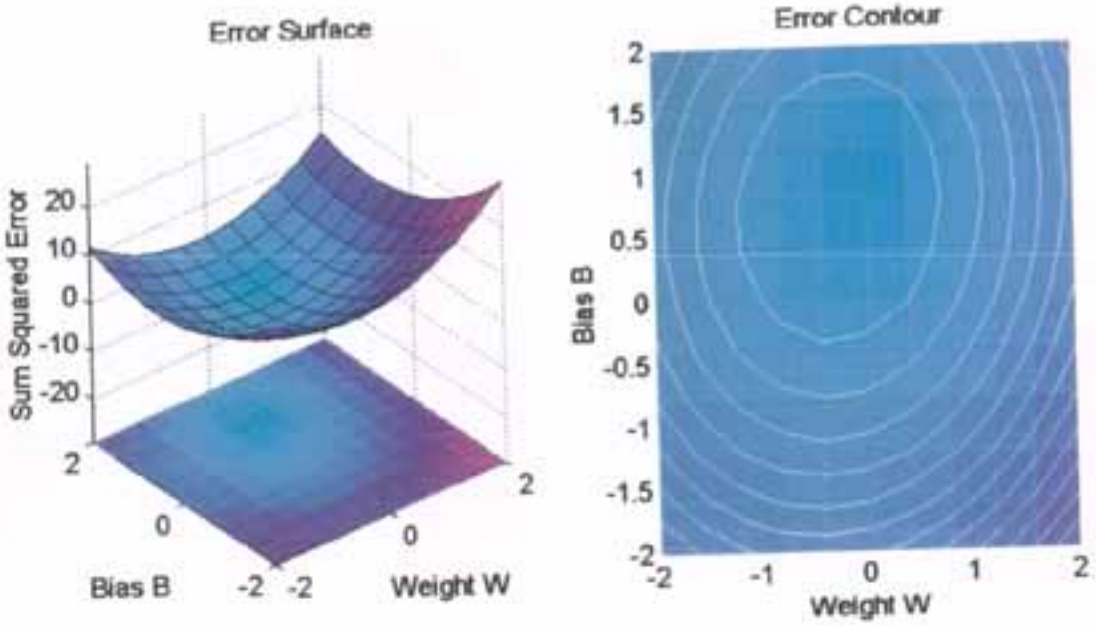
Η ERRSURF υπολογίζει σφάλματα για ένα νευρώνα με ένα εύρος πιθανών τιμών βάρους και bias. Η συνάρτηση PLOTES σχεδιάζει αυτή την επιφάνεια σφάλματος με ένα περιγραφικό διάγραμμα από κάτω.

$$>> w\_range = -2 : 0.4 : 2; \; b\_range = -2 : 0,4 : 2; \quad (3.7)$$

$$>> ES = errsurf(P,T,w\_range,b\_range,'purelin'); \quad (3.8)$$

$$>> plotes(w\_range,b\_range,ES); \quad (3.9)$$

Οι καλύτερες τιμές βάρους και του bias είναι αυτές που έχουν ως αποτέλεσμα το χαμηλότερο σημείο στη επιφάνεια σφάλματος.



Σχήμα 7: Γραφική παράσταση της επιφάνειας σφάλματος

Η MAXLINLR βρίσκει το γρηγορότερο δυνατό ρυθμό μάθησης για να εκπαιδεύσει ένα γραμμικό δίκτυο.

$$>> \max lr = \max linlr(P,'bias'); \quad (3.10)$$

Η NEWLIN δημιουργεί ένα γραμμικό νευρώνα. Για να δούμε τι συμβαίνει όταν ο ρυθμός μάθησης είναι πολύ μεγάλος, αυξάνουμε το ρυθμό μάθησης κατά 225% της προτεινόμενης τιμής.

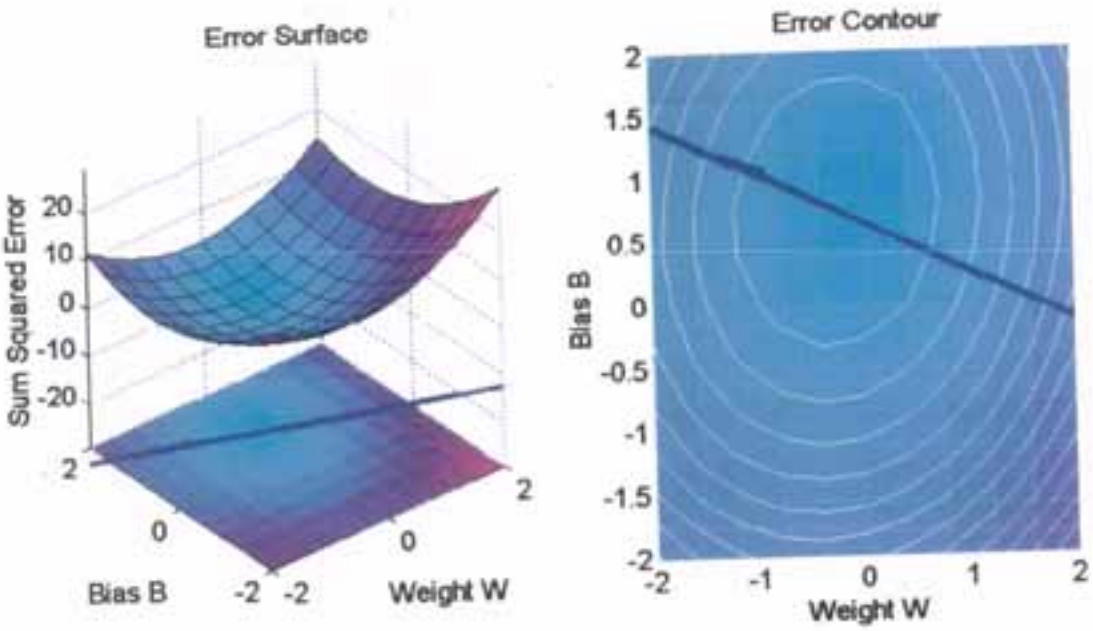
$$>> net = newlin([-22],1,[0],\max lr * 1.5); \quad (3.11)$$

Εξουδετερώνουμε τις λανθασμένες εκπαιδευόμενες παραμέτρους βάζοντας τον μέγιστο αριθμό των περιόδων. Αυτό επιβεβαιώνει ότι η εκπαίδευση θα σταματήσει

$$>> net.trainParam.epochs = 20; \quad (3.12)$$

Χρησιμοποιούμαι τη συνάρτηση plotes για να επανασχεδιάσουμε την επιφάνεια σφάλματος:

$$>> plotes(w\_range, b\_range, ES); \quad (3.13)$$



Σχήμα 8: *: Γραφική παράσταση της επιφάνειας σφάλματος*

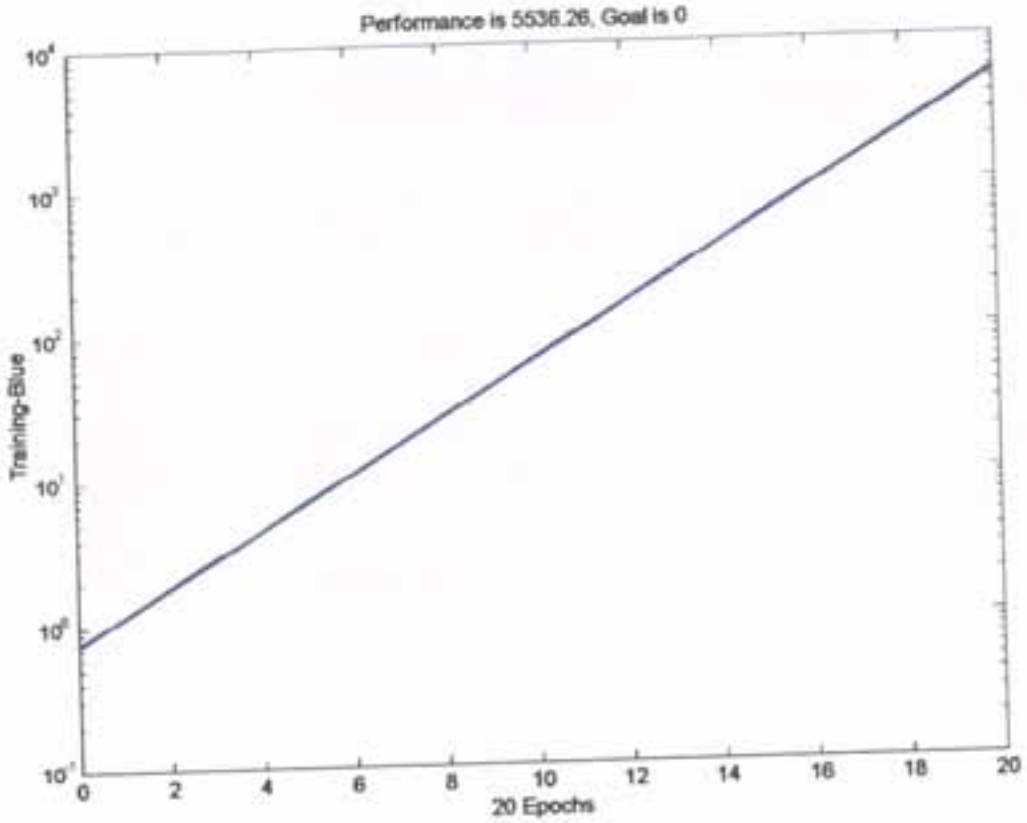
Παραμερίζουμε την τυχαιότητα του βάρους και του bias κάνοντας κλικ στο περιγραφικό διάγραμμα σφάλματος με τη GINPUT:

$>> subplot[1,2,2];[net.IW\{1,1\}net.b\{1\}] = ginput(1)$ (3.14)

Για να δούμε το μονοπάτι που ακολουθήθηκε κατά τη διαδικασία της εκπαίδευσης θα εκπαιδεύσουμε μόνο μια περίοδο κάθε φορά και θα αποκαλέσουμε PLOTEP κάθε εποχή (ο κώδικας δεν φαίνεται εδώ)

$>> net = train[net, P, T];$ (3.15)

Το διάγραμμα δείχνει την ιστορία της εκπαίδευσης. Κάθε τελεία από αυτές αντιπροσωπεύει μια περίοδο και οι μπλε γραμμές δείχνουν κάθε αλλαγή που έγινε από τον κανόνα μάθησης (με παραβίαση του κανόνα των Widrow-Hoff).



Σχήμα 9: Διάγραμμα της εκπαιδευτικής διαδικασίας

Η εξίσωση εκπαίδευσης έχει ως έξοδο το εκπαιδευμένο δίκτυο και την ιστορία της διαδικασίας εκπαίδευσης(tr). Εδώ τα λάθη διαγράφονται σε σχέση με τις περιόδους εκπαίδευσης:

$>> plotperf(tr, net.trainParam.goal);$ (3.16)

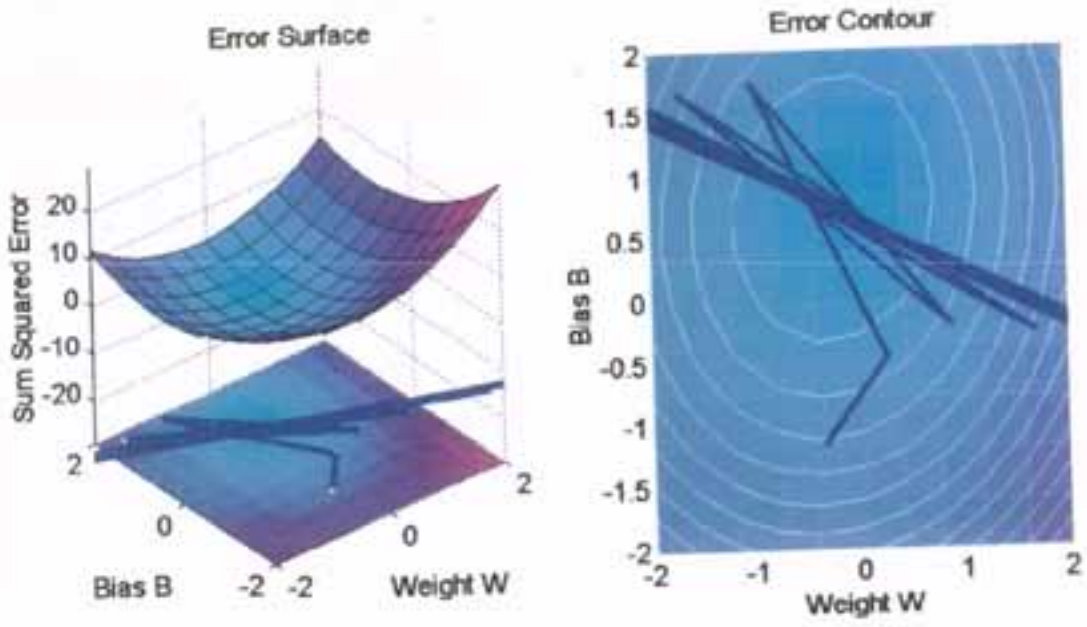
Μπορούμε τώρα να χρησιμοποιήσουμε τη συνάρτηση SIM για να τεστάρουμε τη συσχέτιση με μια από τις αυθεντικές εισόδους,-1.2, και βλέπουμε αν μας επιστρέφει τον στόχο 1.0.

$$>> \begin{matrix} p = -1.2; \\ a = sim(net, p) \end{matrix} \quad (3.17)$$

το αποτέλεσμα που απορρέει είναι το εξής: $a = 99.1466$ (3.18)

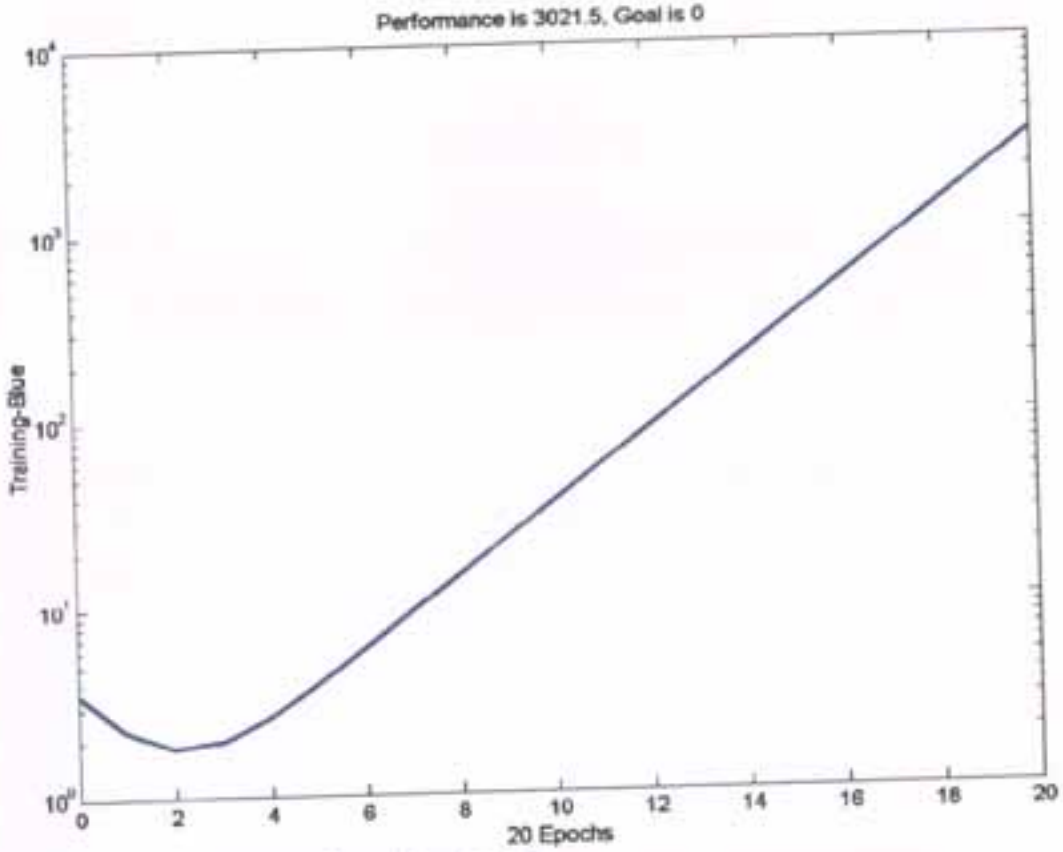
Το αποτέλεσμα δεν είναι πολύ κοντά στο 0,5. Αυτό συμβαίνει επειδή το δίκτυο εκπαιδεύτηκε με πολύ μεγάλο ρυθμό μάθησης.

Αυτή τη φορά κάνουμε κλικ με το ποντίκι στο περιγραφικό διάγραμμα σφάλματος χαμηλά.



Σχήμα 10 : *Γραφική παράσταση της επιφάνειας σφάλματος*

Η γραφική παράσταση που προκύπτει είναι η εξής:

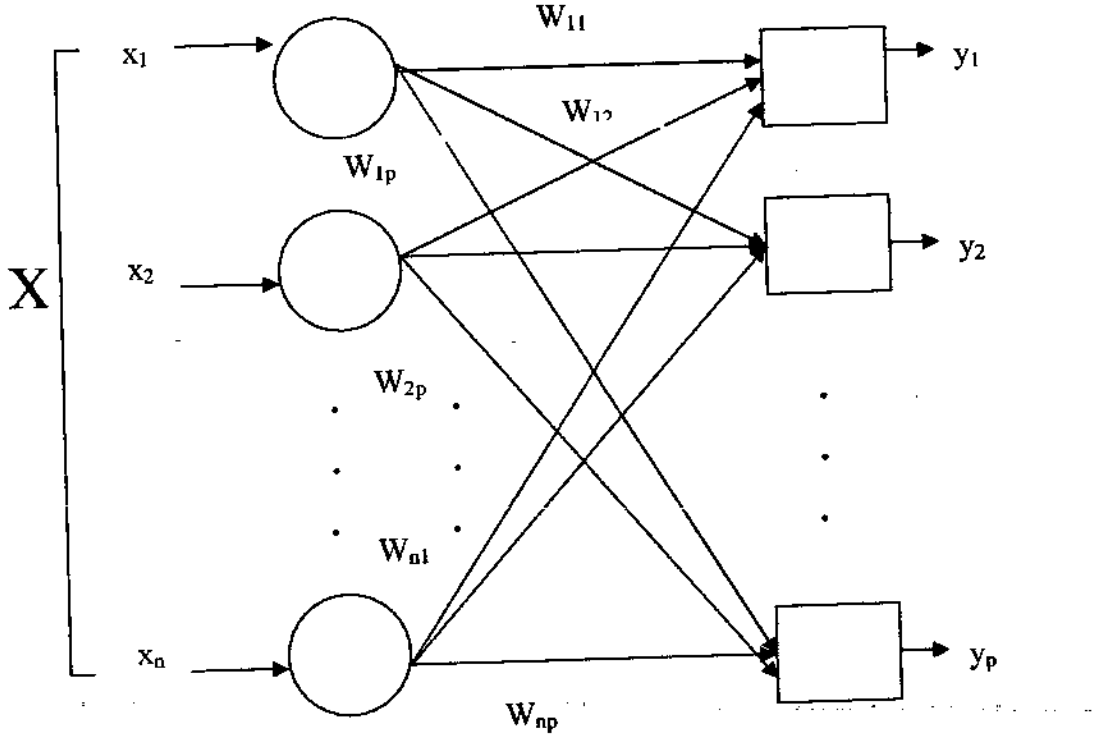


Σχήμα 11: *Διάγραμμα της εκπαιδευτικής διαδικασίας*

Το α αυτή τη φορά είναι $a = -71.508$ και σ' αυτή την περίπτωση το αποτέλεσμα είναι μακριά από το 0,5 λόγο του πολύ μεγάλου ρυθμού μάθησης.

3.8 Τεχνητά Νευρωνικά Δίκτυα ενός επιπέδου (Single-Layer)

Μολονότι ένας νευρώνας μόνος του μπορεί να εκτελέσει μερικές απλές λειτουργίες, που αφορούν την ανίχνευση απλών προτύπων, η υπολογιστική δύναμη των νευρώνων μπορεί να εκτιμηθεί μόνο όταν αυτοί συνδεθούν σε δίκτυο. Το πιο απλό δίκτυο αποτελείτε από μια ομάδα νευρώνων διατεταγμένων σε επίπεδο, όπως φαίνετε στο Σχήμα 12. Σημειώνουμε ότι, οι κύκλοι που φαίνονται στα αριστερά χρησιμοποιούνται μόνο για το μοίρασμα των τιμών εισόδου, δεν εκτελούν υπολογισμούς και έτσι δεν θεωρούνται ότι αποτελούν επίπεδο αντίθετα οι νεύρωνες που εκτελούν υπολογισμούς έχουν την μορφή τετραγώνων. Το σύνολο των τιμών εισόδου X έχει το κάθε του στοιχείο συνδεδεμένο σε κάθε ΤΝΔ μέσω διαφορετικών βαρών. Τα πρώτα ΤΝΔ δεν ήταν περισσότερο πολύπλοκα από ότι αυτό εδώ. Κάθε νευρώνας απλά παράγει ένα άθροισμα των τιμών εισόδου οι οποίες έχουν πολλαπλασιαστεί με τα αντίστοιχα βάρη. Στην πραγματικότητα στα τεχνητά και βιολογικά δίκτυα πολλές από τις συνδέσεις μπορεί να μην υπάρχουν, όμως φαίνονται όλες για λόγους γενικότητας, (μια σύνδεση μπορεί να καταστραφεί αν το βάρος της είναι 0).



Σχήμα 12: *Απλό δίκτυο*
Πηγή: *www.egnatia.ee.auth.gr/~imatz/ann_begin.html*

Είναι πιο βολικό να θεωρούμε τα βάρη σαν τα στοιχεία ενός πίνακα W. Οι διαστάσεις του πίνακα είναι P γραμμές και n στήλες, όπου n ο αριθμός των εισόδων και P ο αριθμός των νευρώνων όπως φαίνεται στο Σχήμα 12. Για παράδειγμα, το βάρος που συνδέει την τρίτη είσοδο με το δεύτερο νευρώνα το συμβολίζουμε W_{32}. Με αυτό τον τρόπο οι n έξοδοι των νευρώνων ενός δικτύου μπορούν να εκφραστούν με πολλαπλασιασμό πίνακα με διάνυσμα. Όπου Y και X είναι δυο διανύσματα στήλες που περιέχουν τα στοιχεία $y_1\ y_2 \ldots y_p$ και $x_1\ x_2 \ldots x_n$ αντίστοιχα.

3.9 Τεχνητά νευρωνικά δίκτυα πολλών επιπέδων (Multilayer)

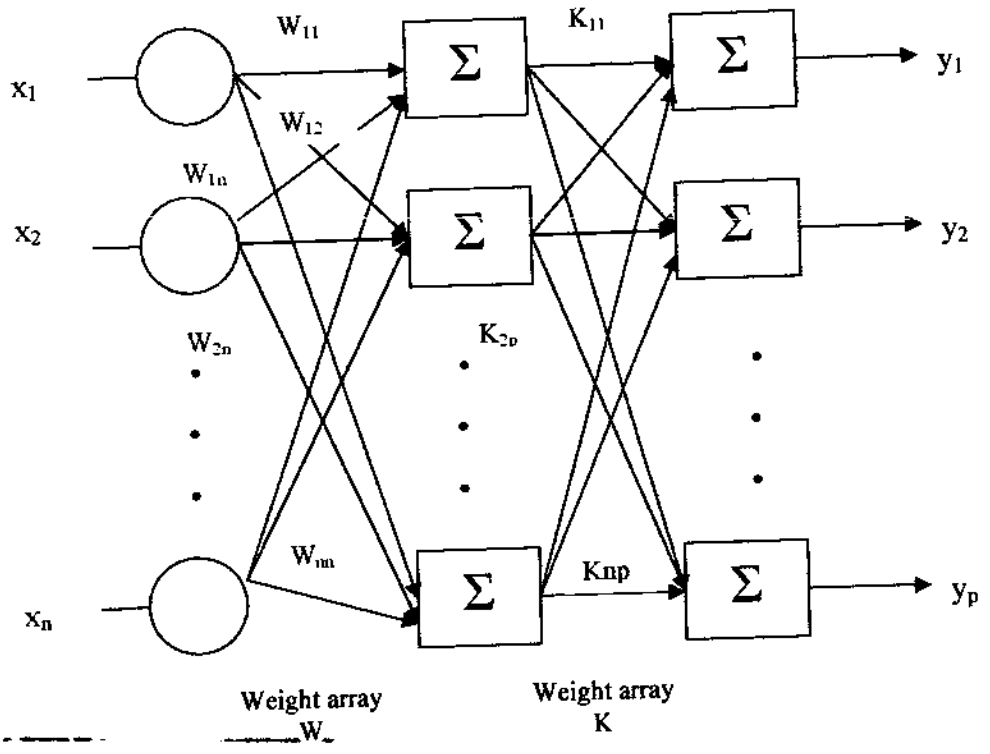
Μεγαλύτερα και περισσότερο πολύπλοκα δίκτυα, προσφέρουν μεγαλύτερη ικανότητα και δυνατότητα υπολογισμών. Μολονότι έχουν κατασκευαστεί δίκτυα με πολλούς διαφορετικούς τρόπους διάταξής των νευρώνων, διατάσσοντας τους νεύρωνες σε επίπεδα μιμούμαστε την στρωματική δομή των διάφορων τμημάτων του εγκεφάλου. Τα πολυεπίπεδα δίκτυα, έχει αποδειχθεί, ότι έχουν μεγαλύτερες ικανότητες από αυτές των μονόεπίπεδων δικτύων και έχουν αναπτυχθεί αποδοτικοί αλγόριθμοι για την εκπαίδευση τους.

Τα πολυεπίπεδα δίκτυα μπορούν να σχηματιστούν από ομάδες μονών επίπεδων δικτύων, όπου η έξοδος ενός επιπέδου αποτελεί την είσοδο του επομένου επιπέδου. Στο Σχήμα 13 φαίνετε ένα τέτοιο δίκτυο, όπου έχουν σχεδιαστεί όλες οι δυνατές διασυνδέσεις. Η έξοδος ενός τέτοιου ΤΝΔ μπορεί να εκφραστεί από τη σχέση Y = KWX όπου:

$$X = [x_1 x_2 \ldots x_n]^t \ldots,\quad Y = [y_1 y_2 \ldots y_n]^t \cdots$$

K ένας $n \times p$ πίνακας του οποίου τα στοιχεία περιέχουν τις τιμές των βαρών K_{ij} και

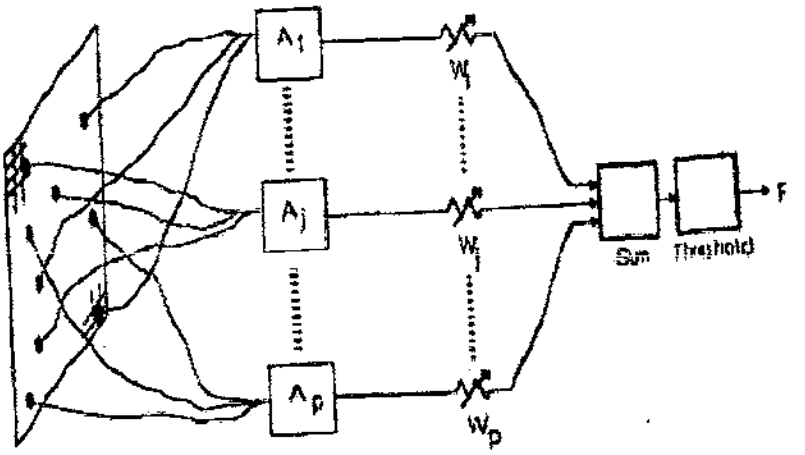
W ένας $p \times n$ πίνακας τα στοιχεία του οποίου περιέχουν τις τιμές των βαρών W_{ij}



Σχήμα 13: Πολυεπίπεδο δίκτυο
Πηγή: www.egnatia.ee.auth.gr/~imatz/ann_begin.html

3.10 Perceptrons

Η εργασία με την περισσότερη επίδραση στα νευρωνικά δίκτυα στη δεκαετία του '60 ονομάστηκε "perceptrons" ένας όρος που προτάθηκε από το Frank Rosenblatt. Το Perceptron (Σχήμα 14) αποδεικνύεται με ένα MCP πρότυπο (νευρώνιο με σταθμισμένες εισαγωγές) και με κάποια, σταθερή, προεπεξεργασία. Οι μονάδες που συμβολίζονται ως A1, A2,...,Ap καλούνται μονάδες ένωσης και στόχος τους είναι να εξαγάγουν συγκεκριμένο, αποτέλεσμα/ χαρακτηριστικό από τις εικόνες εισαγωγής. Τα Perceptrons μιμούνται τη βασική λειτουργία α του οπτικού συστήματος. Χρησιμοποιήθηκαν κυρίως στην αναγνώριση προτύπων ακόμα κι αν οι ικανότητές τους επεκτάθηκαν πολύ περισσότερο. (www.iau.dtu.dk/~jj/pubs/nnet.pdf)

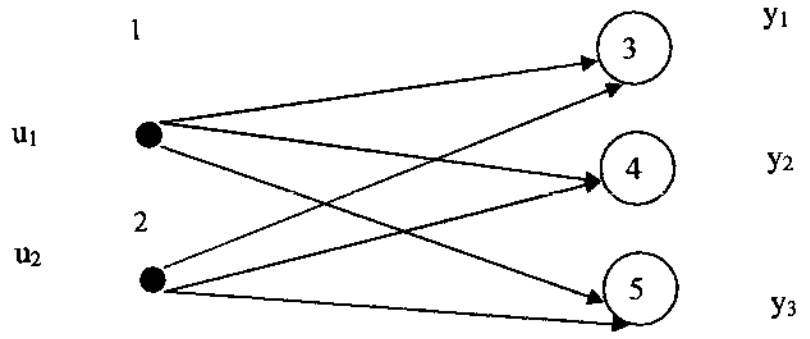


Σχήμα 14: MCP πρότυπο
Πηγή: *www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html*

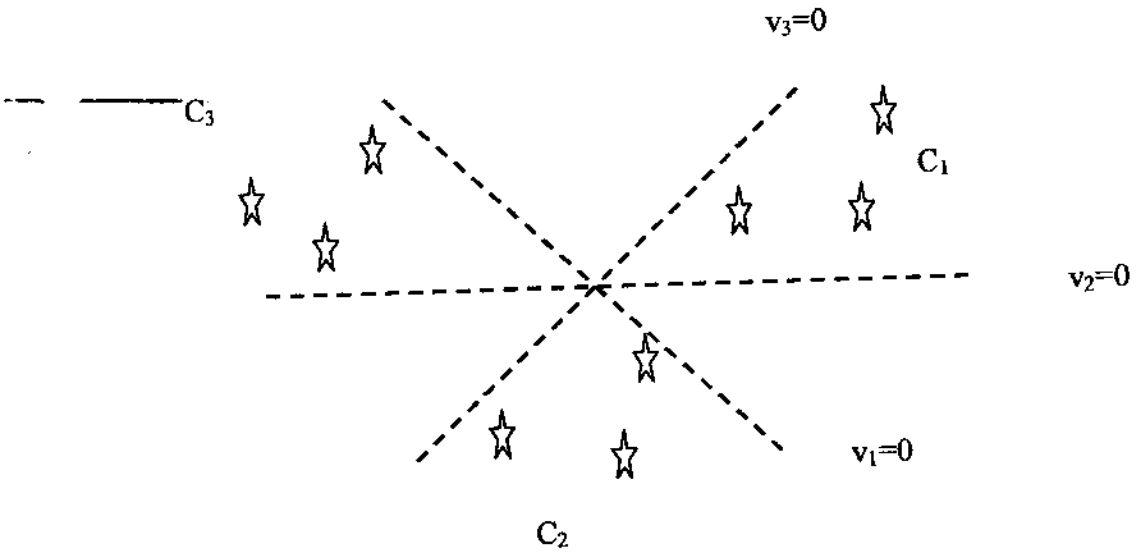
Το 1969 οι Minsky και Papert έγραψαν ένα βιβλίο που περιέγραφε τους περιορισμούς του Perceptron ενός επιπέδου. Η επίδραση που είχε το βιβλίο ήταν τεράστια. Το βιβλίο ήταν καλογραμμένο και παρουσίαζε από μαθηματική άποψη ότι τα ενός επιπέδου Perceptrons δεν μπορούν να κάνουν μερικές βασικές διαδικασίες αναγνώρισης προτύπων όπως τον καθορισμό της ισότητας ενός σχήματος/ τύπο ή τον καθορισμό εάν ένα σχήμα συνδέεται ή όχι.

3.10.1 Single layer perceptron

Λαμβάνοντας υπόψη ένα πρόβλημα που απαιτεί περισσότερες από δύο κατηγορίες, διαφορετικά perceptrons μπορούν να συνδυαστούν σε ένα δίκτυο. Η απλούστερη μορφή ενός δικτύου με *επίπεδα* έχει ακριβώς ένα επίπεδο εισόδου των κόμβων πηγής που συνδέονται με ένα επίπεδο εξόδου των νευρωνίων, αλλά όχι αντίστροφα. Το Σχήμα 15 παρουσιάζει ένα perceptron, ενός επιπέδου με πέντε κόμβους ικανούς να αναγνωρίσουν τρείς γραμμικές διαχωριζόμενες κατηγορίες/ τάξεις (τρεις κόμβους εξόδου) με τη βοήθεια δύο χαρακτηριστικών γνωρισμάτων (δύο κόμβων εισόδου). Κάθε νευρώνας εξόδου καθορίζει ένα διάνυσμα βάρους που καθορίζει στη συνέχεια τα υπερεπίπεδα (hyperplane). Τα υπερεπίπεδα χωρίζουν τις κατηγορίες όπως φαίνεται στο Σχήμα 16.



Σχήμα 15: perceptron ενός επιπέδου με τρεις εξόδους νευρώνων
Πηγή: www.iau.dtu.dk/~jj/pubs/nnet.pdf, σελ.10



Σχήμα 16: Τρεις ξεχωριστές τάξεις επιπέδων
Πηγή: www.iau.dtu.dk/~jj/pubs/nnet.pdf, σελ.10

Η εξίσωση ενεργοποίησης θα μπορούσε να έχει διάφορες μορφές ανάλογα με την εφαρμογή, εδώ βλέπουμε έξι διαφορετικούς τύπους. Οι μαθηματικές εκφράσεις για τα τρία στην κάτω σειρά είναι, αντίστοιχα

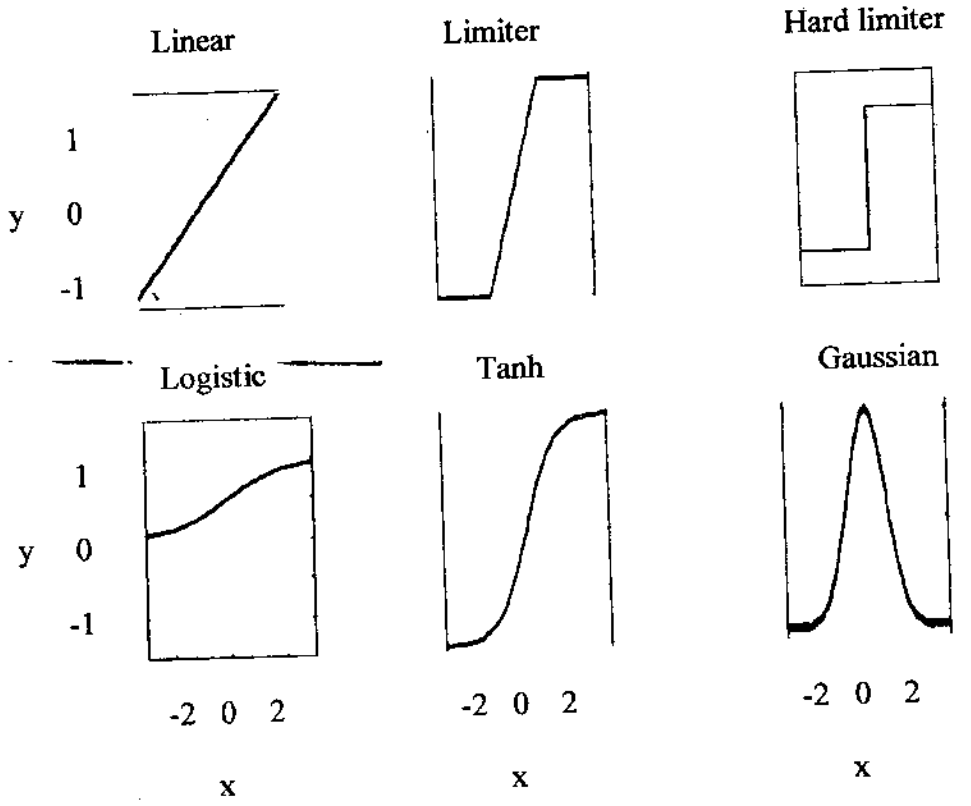
$$f(x) = \frac{1}{1+\exp(-ax)} \qquad (3.21)$$

$$f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} = \tanh(x) \qquad (3.22)$$

$$f(x) = \exp(-x^2) \; (3.23) \Longrightarrow$$

$$\frac{1}{\sqrt{2\pi * \sigma}} = -e\frac{x^2}{Z\sigma^2}$$

Η πρώτη είναι η *λογιστική* συνάρτηση, ένα παράδειγμα της ευρέως χρησιμοποιημένης *sigmoid* συνάρτησης, γενικά sigmoid είναι η s - διαμορφωμένη λειτουργία, που αυξάνεται και είναι ασυμπτωτική. Η παράμετρος α χρησιμοποιείται για να ρυθμίσει την κλίση στο όριο καθώς η α τείνει στο άπειρο. Η δεύτερη είναι η εξίσωση της *υπερβολικής εφαπτομένης, η οποία* είναι μια συνάρτηση αυτή συμμετρική. Η τρίτη εξίσωση είναι η εξίσωση Gausian μιας συνάρτησης του x, μέσης τιμής 0 και διασποράς 1(Gaussian function).



Σχήμα 17: *Παραδείγματα 6 συναρτήσεων ενεργοποίησης ενός νευρώνα*
Πηγή: *www.iau.dtu.dk/~jj/pubs/nnet.pdf, σελ.11*

(Για την γραμμική συνάρτηση ενεργοποίησης περιορίσεων δες σελ. 15)

3.10.2 Multi-layer perceptron

Το ενός επιπέδου perceptron μπορεί να ταξινομήσει μόνο γραμμικά διαχωριζόμενα προβλήματα. Για τα μη-διαχωριζόμενα προβλήματα είναι απαραίτητο να χρησιμοποιηθούν περισσότερα επίπεδα. Ένα *πολυεπίπεδο* (feedforward) δίκτυο έχει ένα ή περισσότερα κρυμμένα επίπεδα των οποίων τα νευρώνια καλούνται κρυμμένα νευρώνια. Το δίκτυο *συνδέεται πλήρως*, επειδή κάθε κόμβος σε κάθε επίπεδο συνδέεται με όλους τους κόμβους

στο επόμενο επίπεδο. Εάν μερικές από τις συνδέσεις λείπουν, το δίκτυο *συνδέεται μερικώς*. Όταν λέμε ότι ένα δίκτυο έχει n επίπεδα, μετράμε μόνο τα κρυμμένα επίπεδα και το επίπεδο εξόδου, το επίπεδο εισόδου δεν προσμετράτε, επειδή οι κόμβοι του δεν εκτελούν υπολογισμούς. Ένα ενός επιπέδου perceptron αναφέρεται έτσι σαν ένα δίκτυο με ακριβώς ένα επίπεδο εξόδου.

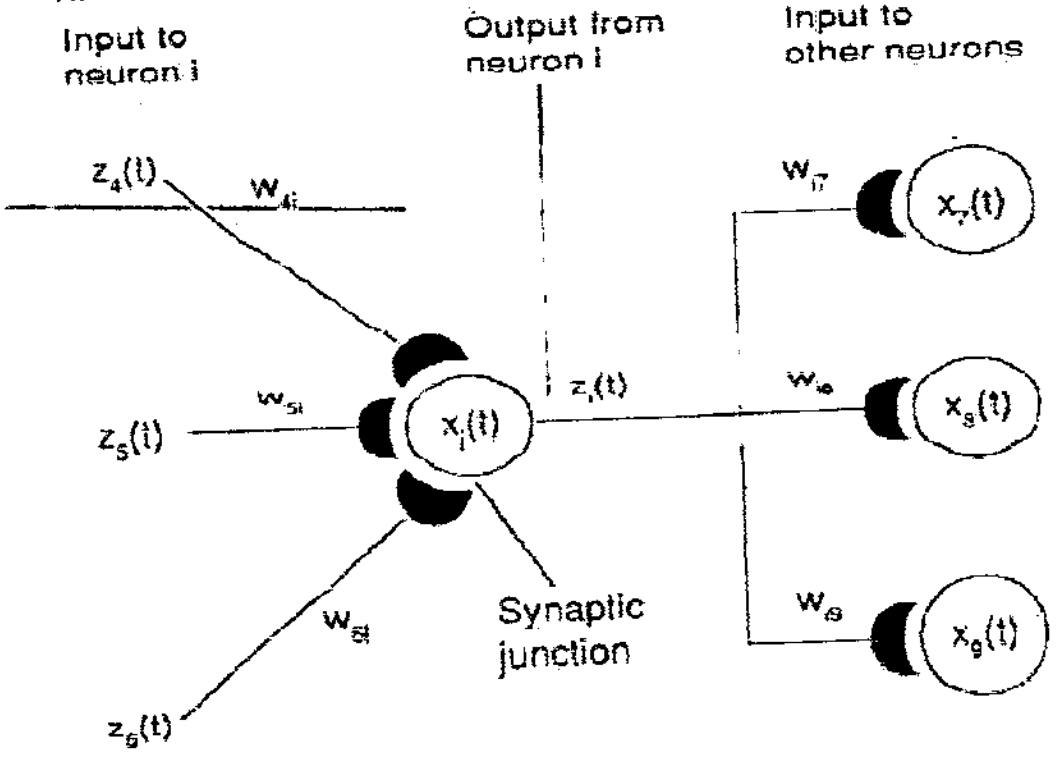
4 Αλγόριθμοι Μάθησης

4.1 Ο αλγόριθμος οπισθοδρόμησης (Back- Propagation)

Προκειμένου να εκπαιδευθεί ένα νευρωνικό δίκτυο για να εκτελέσουμε κάποιο έργο, πρέπει να ρυθμίσουμε τα βάρη κάθε μονάδας κατά τέτοιο τρόπο ώστε το σφάλμα μεταξύ της επιθυμητής εξόδου και της πραγματικής να γίνει ελάχιστο. Αυτή η διαδικασία προϋποθέτει ότι το νευρωνικό δίκτυο υπολογίζει την παράγωγο της συνάρτησης του σφάλματος ως προς τα βάρη. Με άλλα λόγια, πρέπει να υπολογίσει πώς το σφάλμα αλλάζει καθώς κάθε βάρος αυξάνεται ή μειώνεται ελαφρώς. Ο αλγόριθμος Back- Propagation είναι η ευρύτατα χρησιμοποιούμενη μέθοδος για τον προσδιορισμό του σφάλματος των βαρών.
Ο αλγόριθμος back- propagation είναι ευκολότερο να κατανοηθεί εάν όλες οι μονάδες στο δίκτυο είναι γραμμικές. Ο αλγόριθμος υπολογίζει καθένα σφάλμα των βαρών αφού πρώτα υπολογίσει το **ΕΑ**, δηλ. ο ρυθμός κατά τον οποίο το σφάλμα αλλάζει καθώς αλλάζει η δραστηριότητα μιας μονάδας ενός επίπεδου. Για τις μονάδες εξόδου, το **ΕΑ** είναι απλά η διαφορά μεταξύ της πραγματικής και επιθυμητής εξόδου. Για να υπολογίσουμε το **ΕΑ** μιας κρυμμένης μονάδας στο επίπεδο πριν από αυτό της εξόδου, προσδιορίζουμε αρχικά όλα τα βάρη μεταξύ εκείνης της κρυμμένης μονάδας και των μονάδων εξόδου με τις οποίες συνδέεται. Πολλαπλασιάζουμε έπειτα εκείνα τα βάρη με τα **ΕΑ** εκείνων των μονάδων εξόδου και προσθέτουμε τα γινόμενα. Αυτό το άθροισμα εκφράζει τα **ΕΑ** για την επιλεγμένη κρυμμένη μονάδα. Μετά τον υπολογισμό όλων των επιμέρους **ΕΑ** στο κρυμμένο επίπεδο, μπορούμε να υπολογίσουμε τα **ΕΑ** για άλλα επίπεδα/ στρώματα, που κινούνται από στρώμα σε στρώμα σε μια κατεύθυνση αντίθετη από τον τρόπο που οι δραστηριότητες διαδίδονται μέσα στο δίκτυο. Μόλις υπολογιστεί το **ΕΑ** για μια μονάδα, είναι απλό να υπολογιστεί το σφάλμα των βαρών για κάθε εισερχόμενη σύνδεση της μονάδας. Το σφάλμα των βαρών είναι το προϊόν της **ΕΑ** και της δραστηριότητας μέσω της εισερχόμενης σύνδεσης.

4.2 Η μαθηματική προσέγγιση του αλγορίθμου οπισθοδρόμησης (back – propagation)

Οι μονάδες συνδέονται η μια με την άλλη. Οι συνδέσεις αντιστοιχούν στις άκρες της υφιστάμενης κατευθυνόμενης γραφικής παράστασης. Υπάρχει ένας πραγματικός αριθμός που συνδέεται με κάθε σύνδεση, η οποία καλείται βάρος της σύνδεσης. Συμβολίζουμε με Wij το βάρος της σύνδεσης από τη μονάδα u_i στη μονάδα uj. Στη συνέχεια είναι απλό να εκφραστεί ο τύπος της σύνδεσης στο δίκτυο από ένα πίνακα βαρών W του οποίου τα στοιχεία είναι τα βάρη Wij. Δύο τύποι συνδέσεων διακρίνονται συνήθως: ο ενισχυτικός και ο ανασταλτικός. Ένα θετικό βάρος σε μια σύνδεση αντιπροσωπεύει μια ενισχυτική σύνδεση ενώ ένα αρνητικό βάρος αντιπροσωπεύει μια ανασταλτική σύνδεση. Ο τύπος της σύνδεσης χαρακτηρίζει την αρχιτεκτονική του δικτύου.



Σχήμα 18: Αρχιτεκτονική του δικτύου
Πηγή: www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

Η έξοδος μιας μονάδας στο επίπεδο των εξόδων καθορίζει τη δραστηριότητά της ακολουθώντας μια διαδικασία δύο βημάτων.

- Κατ' αρχάς, υπολογίζει τη συνολική σταθμισμένη είσοδο x_j, από τη σχέση:

$$x_j = \sum_i y_i w_{ij} \quad (4.1)$$

όπου το y_i είναι το επίπεδο δραστηριότητας της μονάδας jth στο προηγούμενο στρώμα και το Wij είναι το βάρος της σύνδεσης μεταξύ της i-στης και της ς j-στης μονάδας.

● Στη συνέχεια, η μονάδα υπολογίζει την έξοδο y_j χρησιμοποιώντας κάποια συνάρτηση της συνολικής σταθμισμένης εισόδου. Τυπικά χρησιμοποιείτε τη σιγμοειδή συνάρτηση που είδαμε στην παράγραφο 3.10.1 και η οποία για ευκολία επαναλαμβάνεται στη συνέχεια:

$$y_i = \frac{1}{1+e^{-x_j}} \quad (4.2)$$

Μόλις καθοριστούν οι δραστηριότητες όλων των μονάδων εξόδου, το δίκτυο υπολογίζει το σφάλμα E, το οποίο καθορίζεται από τον τύπο:

$$E = \frac{1}{2}\sum_i (y_i - d_i)^2 \quad (4.3)$$

όπου το y_j είναι το επίπεδο δραστηριότητας της j-στης μονάδας στο κορυφαίο επίπεδο και το dj είναι η επιθυμητή έξοδος της j-στης μονάδας.

Ο αλγόριθμος back-propagation αποτελείται από τέσσερα βήματα:

1. Αρχικά υπολογίζετε πόσο επηρεάζεται το σφάλμα καθώς αλλάζει η δραστηριότητα μιας μονάδας εξόδου. Αυτή η παράγωγος λάθους ορίζεται σαν τη διαφορά μεταξύ της πραγματικής και επιθυμητής δραστηριότητας.

$$EA = \frac{dE}{dy_i} = y_i - d_j \quad (4.4)$$

2. Στη συνέχεια υπολογίζετε πόσο γρήγορα αλλάζει το σφάλμα καθώς η συνολική είσοδος που παραλαμβάνεται από μια μονάδα εξόδου αλλάζει.. Αυτή η ποσότητα (EI) είναι η απάντηση από το βήμα 1 που πολλαπλασιάζεται με το ρυθμό στο οποίο η έξοδος μιας μονάδας αλλάζει καθώς η αλλάζει συνολική είσοδος.

$$\mathrm{EI}_j = \frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \times \frac{dy_j}{dx_j} = EA_j y_j (1 - y_j) \quad (4.5)$$

3. Έπειτα υπολογίζετε πόσο γρήγορα γίνονται οι αλλαγές του σφάλματος καθώς αλλάζει ένα βάρος της σύνδεση σε μια μονάδα εξόδου. Αυτή η ποσότητα (EW) είναι η απάντηση από το βήμα 2 που πολλαπλασιάζεται με το επίπεδο δραστηριότητας της μονάδας από την οποία προέρχεται η σύνδεση.

$$EW_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial W_{ij}} = EI_j y_i \quad (4.6)$$

4. Τέλος υπολογίζετε πόσο γρήγορα γίνονται οι αλλαγές του σφάλματος καθώς η δραστηριότητα μιας μονάδας στο προηγούμενο στρώμα/ επίπεδο αλλάζει. Αυτό το κρίσιμο βήμα επιτρέπει στον back – propagation να εφαρμοστεί σε πολυεπίπεδα δίκτυα. Όταν η δραστηριότητα μιας μονάδας στο προηγούμενο στρώμα αλλάζει, επηρεάζει τις ενέργειες όλων των μονάδων εξόδου με τις οποίες συνδέεται. Έτσι για να υπολογίσουμε τη γενική επίδραση στο σφάλμα, προσθέτουμε μαζί όλα τα επιμέρους αποτελέσματα στις μονάδες

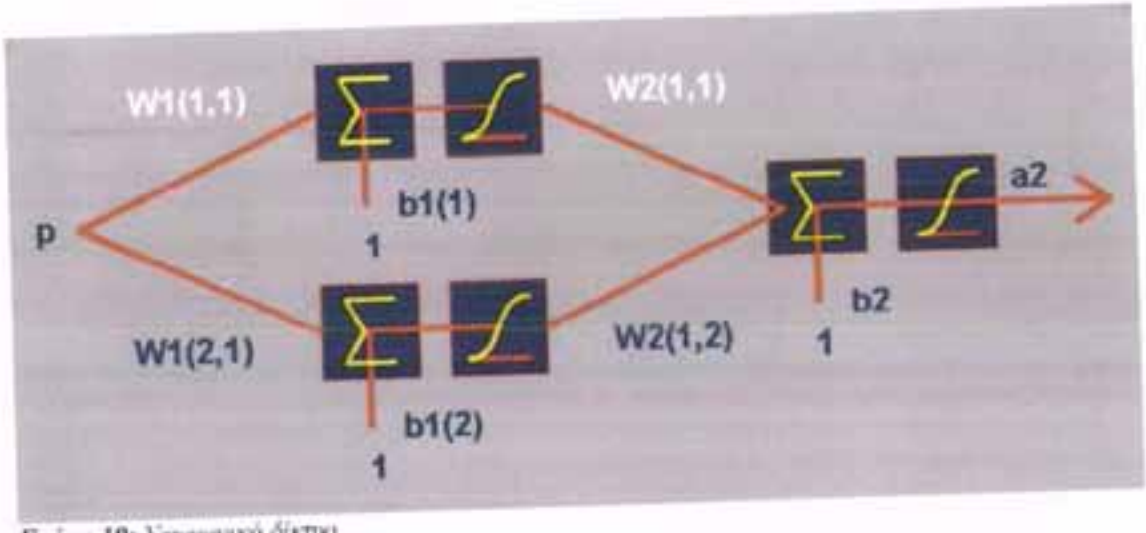
εξόδου. Αλλά ο υπολογισμός κάθε επίδρασης είναι απλός. Αυτή είναι η απάντηση στο βήμα 2 που πολλαπλασιάζεται με το βάρος στη σύνδεση σε εκείνη την μονάδα εξόδου, δηλαδή:

$$EA_i = \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial y_i} = \sum_j EI_j W_{ij} \quad (4.7)$$

Με τη χρήση των βημάτων 2 και 4, μπορούμε να μετατρέψουμε το ΕΑ (δές σελίδα 33) ενός στρώματος των μονάδων σε ΕΑ για το προηγούμενο στρώμα. Αυτή η διαδικασία μπορεί να επαναληφθεί για να πάρει τα ΕΑ για όσα προηγούμενα στρώματα επιθυμούμε. Μόλις ξέρουμε το ΕΑ μιας μονάδας, μπορούμε να χρησιμοποιήσουμε τα βήματα 2 και 3 για να υπολογίσουμε τα σφάλματα των βαρών στις εισερχόμενες συνδέσεις της.

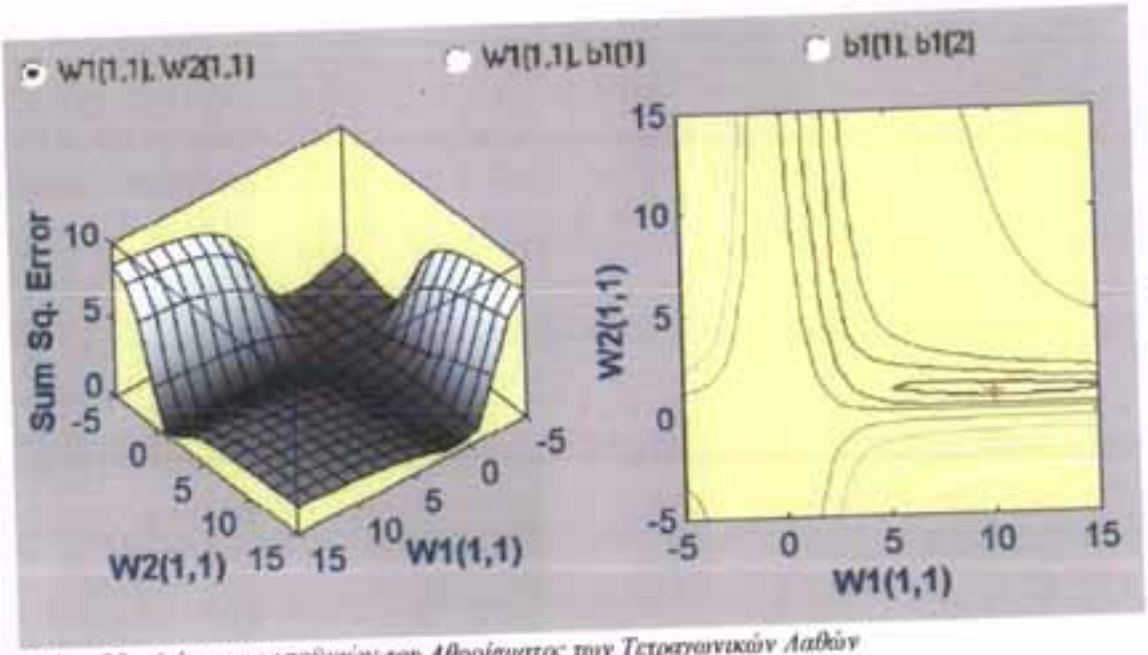
Το παράδειγμα που ακολουθεί αναφέρεται στο Steepest Descent Back-propagation (δες παράγραφος 4.3.4):

Το νευρωνικό δίκτυο που θα χρησιμοποιήσουμε είναι το εξής:



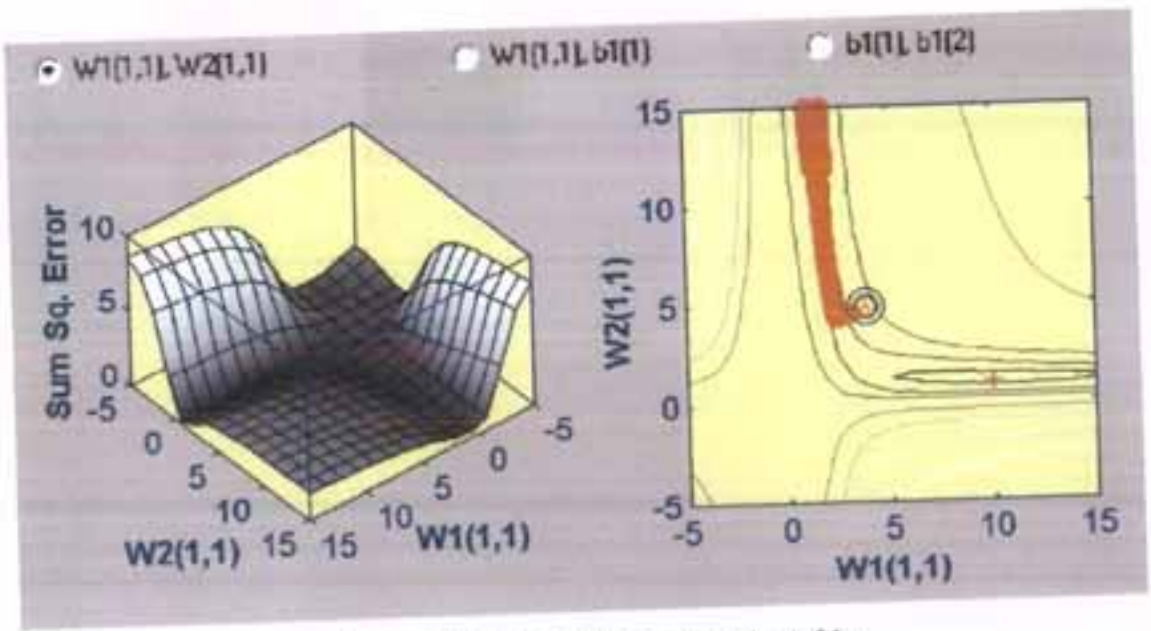
Σχήμα 19: Νευρωνικό δίκτυο

Έχοντας επιλέξει τα βάρη η αντίστοιχη επιφάνεια λάθους και το διάγραμμα ισοϋψών φαίνονται στο παρακάτω σχήμα(δεξιά και αριστερά αντίστοιχα):

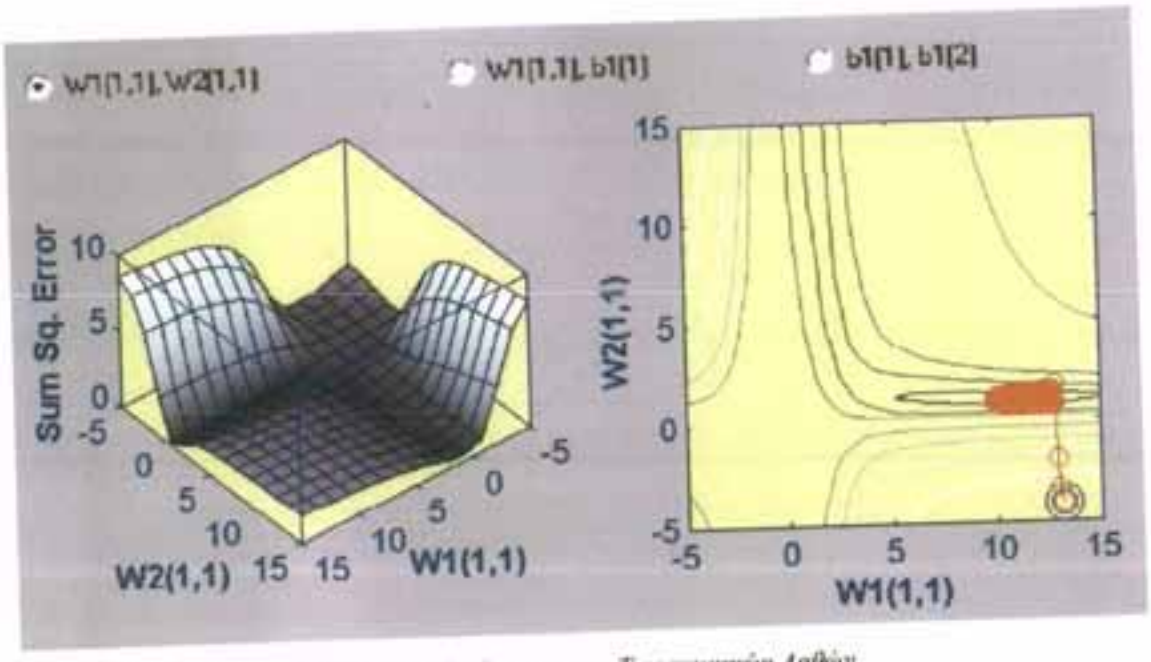


Σχήμα 20: Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών

Πατώντας σε οποιοδήποτε σημείο του περιγραφικού σχεδιαγράμματος αρχίζει ο αλγόριθμος μάθησης Steepest Descent από κάτω φαίνονται δύο παραδείγματα, μια που κάνουμε κλικ κοντά στο στόχο και μια που κάνουμε μακριά, ενώ ο Steepest Descent στο τέλος μας βρίσκει τις καλύτερες τιμές γύρω από το στόχο.

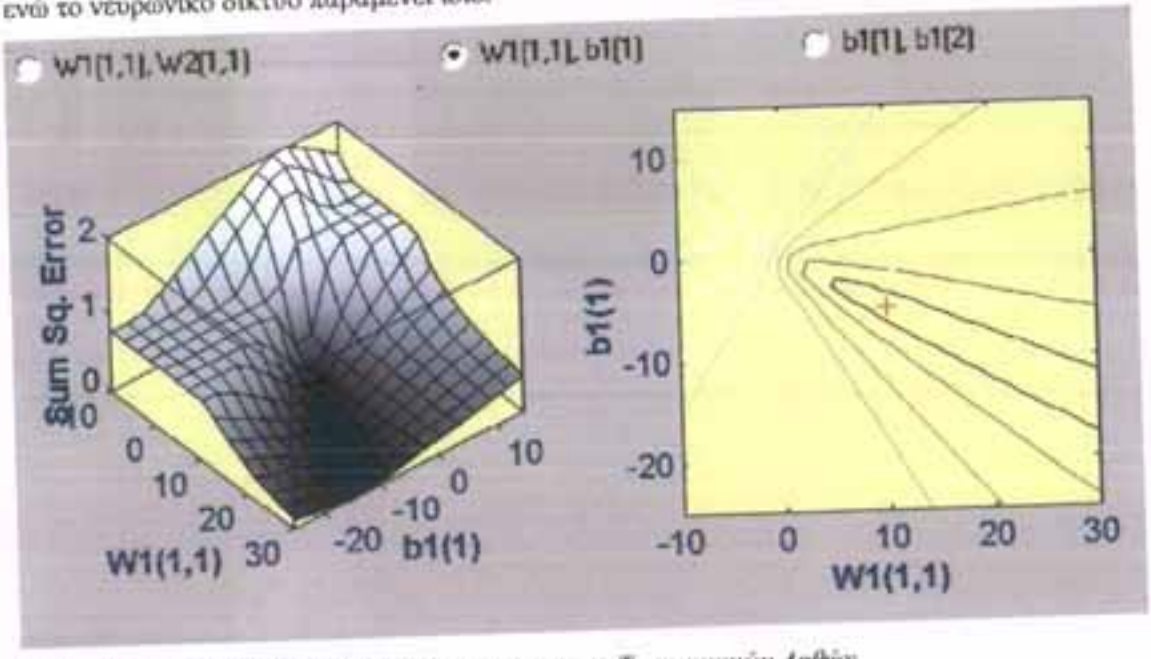


Σχήμα 21: Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών



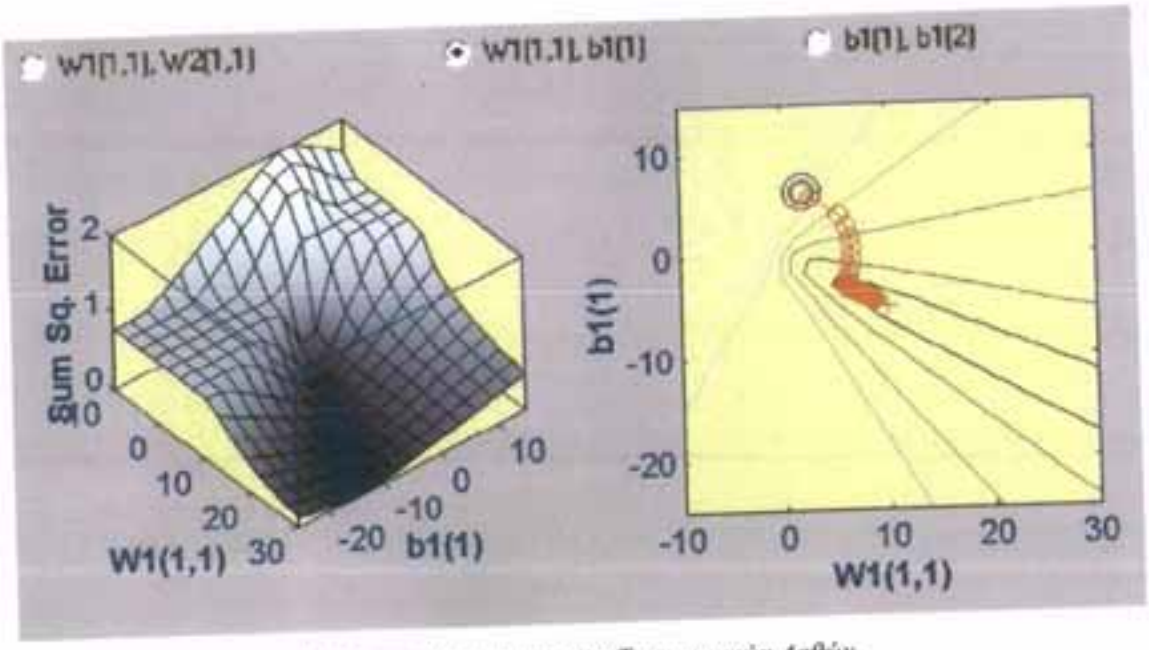
Σχήμα 22: Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών

Τώρα επιλέγοντας το βάρος και το bias, βλέπουμε ότι το περιγραφικό διάγραμμα αλλάζει, ενώ το νευρωνικό δίκτυο παραμένει ίδιο.

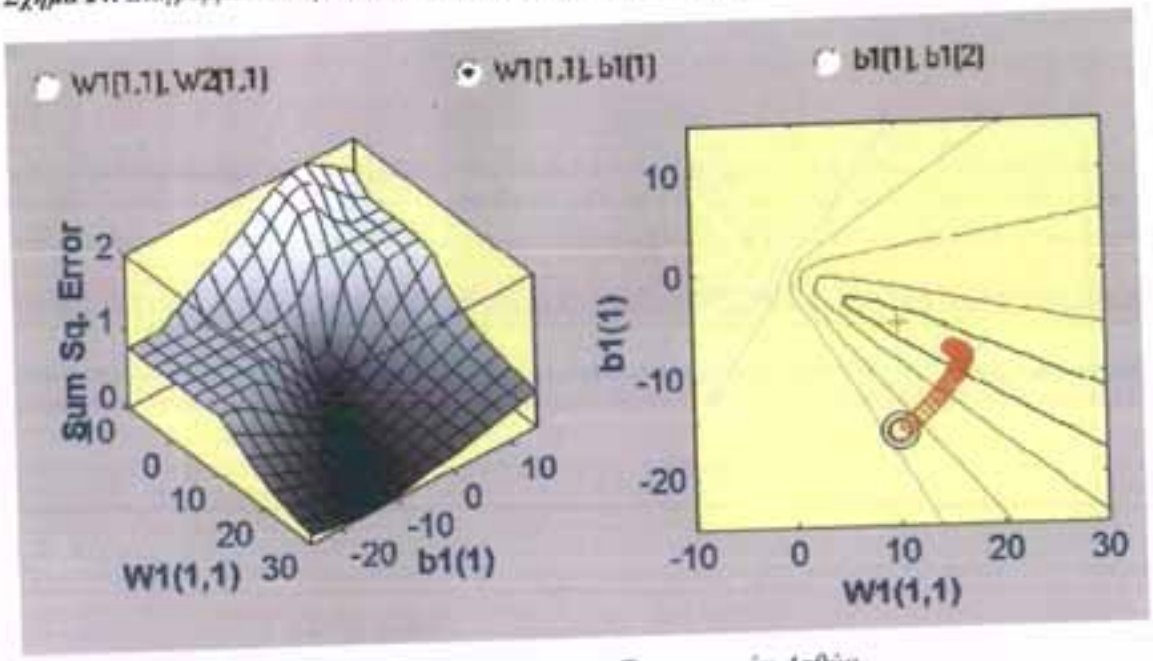


Σχήμα 23: Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών

Τώρα θα εστιάσουμε σε ένα άλλο σημείο και θα διαπιστώσουμε πως ο αλγόριθμος μάθησης Steepest Descent προσπαθεί να βρει τις καλύτερες δυνατές τιμές κοντά στο στόχο.

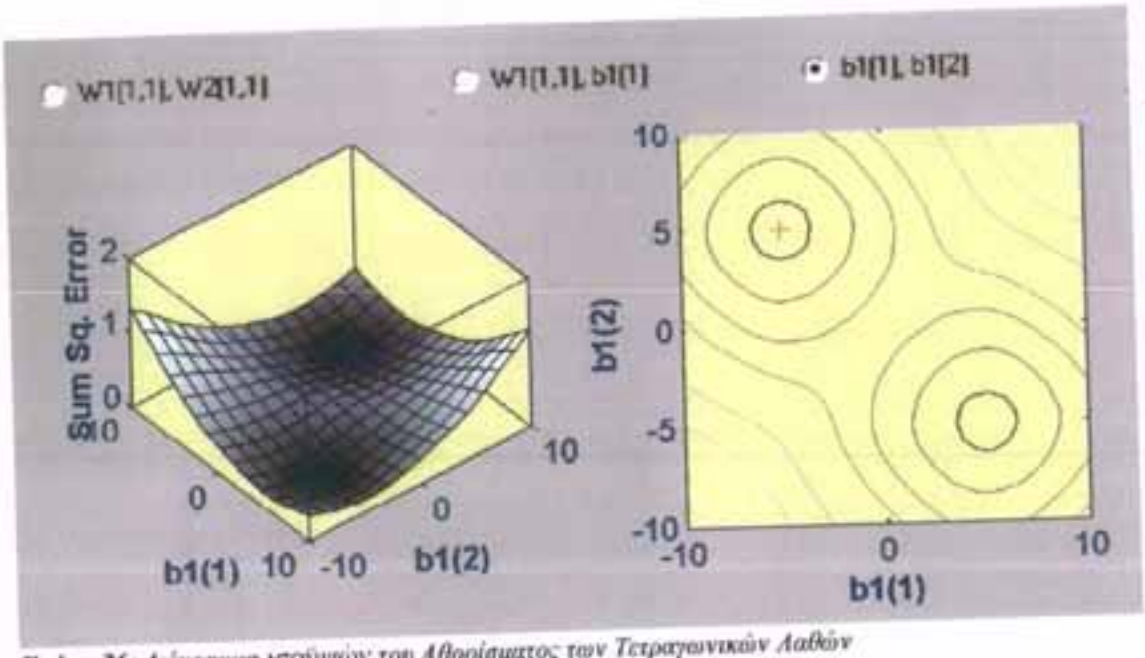


Σχήμα 24: Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών

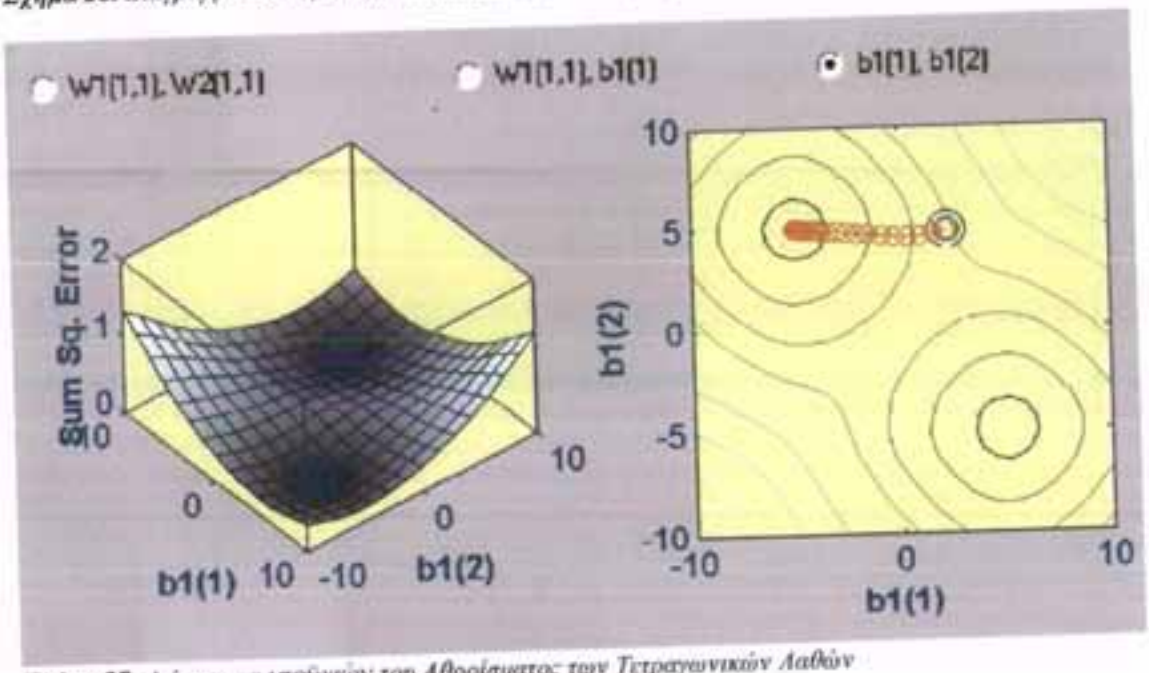


Σχήμα 25: Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών

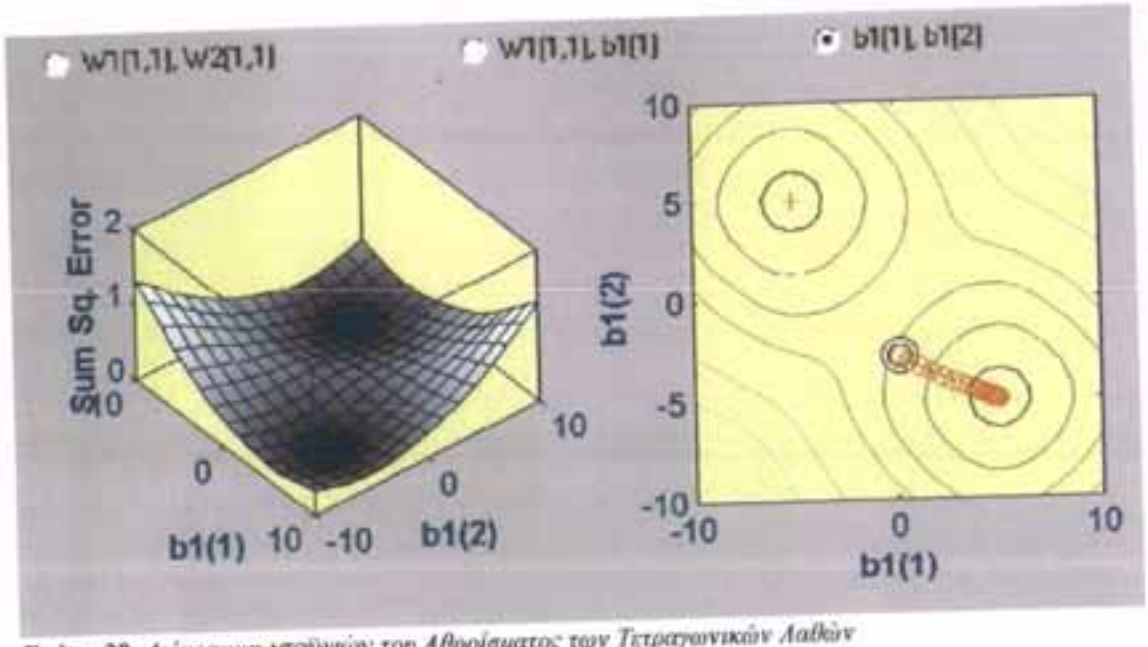
Αυτή τη φορά θα είναι επιλεγμένα μόνο τα bias και πάλι θα αλλάζει μόνο το δεξί περιγραφικό διάγραμμα ενώ θα δούμε πάλι δύο παραδείγματα κάνοντας κλικ μια κοντά στο στόχο και μία μακριά από το στόχο.



Σχήμα 26: Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών



Σχήμα 27: Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών



Σχήμα 28: *Διάγραμμα ισοϋψών του Αθροίσματος των Τετραγωνικών Λαθών*

Ενδιαφέρον είναι και η γραμμική προσαρμογή ενός μη γραμμικού προβλήματος, καθώς πολύ συχνά καλούμαστε στην πράξη να αντιμετωπίσουμε μη γραμμικά προβλήματα με τη βοήθεια γραμμικών συστημάτων. Παρακάτω θα δούμε ένα τέτοιο παράδειγμα. Εδώ θα χρησιμοποιήσουμε εκτός από τις γνωστές μας συναρτήσεις[1] ακόμη μία νέα :

Η συνάρτηση **SUBPLOT** μας παρέχει την δυνατότητα πολλών γραφικών παραστάσεων σε ένα σχήμα.

Συγκεκριμένα η H = SUBPLOT(m,n,p), ή η SUBPLOT(mnp), χωρίζει το παράθυρο σχήματος σε έναν πίνακα mxn γραφικών παραστάσεων, ενώ το p καθορίζει την p-th από τις mxn γραφικές παραστάσεις και ως εκ τούτου παίρνει τιμές από 1 έως mxn. Οι άξονες μετριούνται κατά μήκος της κορυφής του σχήματος που εμφανίζεται στο παράθυρο, μετά συνεχίζει η δεύτερη σειρά κ.λ.π.

Με τις γνωστές συναρτήσεις ένας γραμμικός νευρώνας εκπαιδεύεται για να βρει τη γραμμική προσαρμογή του ελάχιστου αθροίσματος του τετραγωνικού σφάλματος. Σε ένα πρόβλημα μη γραμμικής εισόδου / εξόδου.

Το P ορίζει τέσσερις τρόπους/ σχέδια εισόδων 1- στοιχείου (στήλη διανυσμάτων):

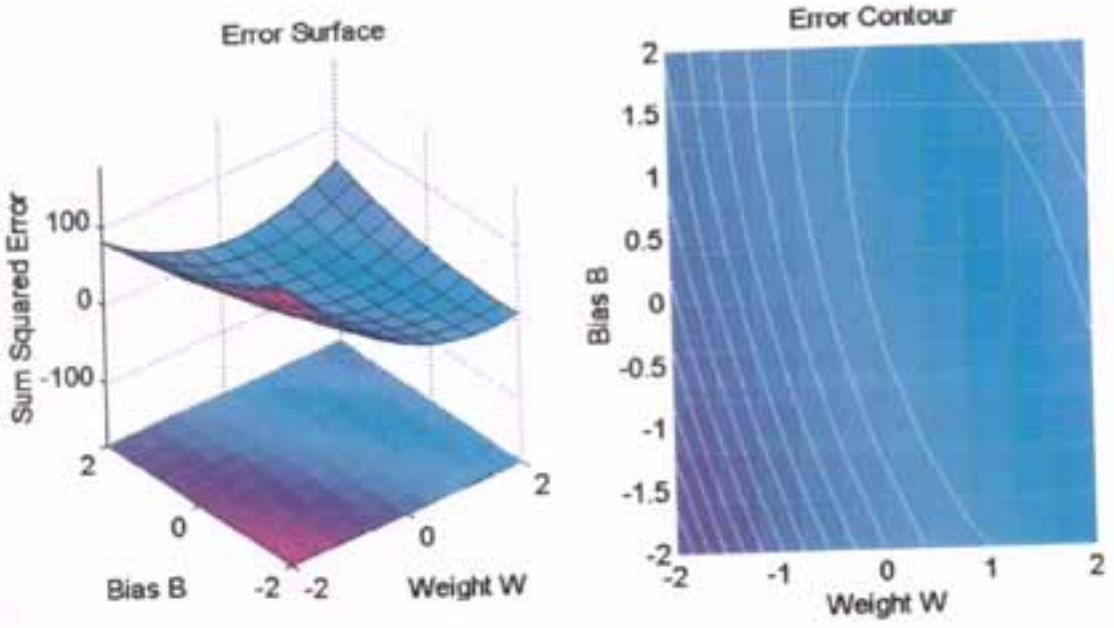
$$>> P = \left[+1.0 +1.5 +3.0 -1.2\right]; \quad (4.8)$$

Το T ορίζει τους συνδεδεμένους στόχους 1- στοιχείου (στήλη διανυσμάτων):

$$>> T = \left[+0.5 +1.1 +3.0 -1.0\right]; \quad (4.9)$$

[1] Τις συναρτήσεις αυτές μπορούμε να τις θυμηθούμε αν ανατρέξουμε στην παράγραφο 3.7 Ρυθμός μάθησης και συγκεκριμένα στο παράδειγμα του πολύ μεγάλου ρυθμού μάθησης

Πρέπει να τονίσουμε ότι η σχέση ανάμεσα στις τιμές του P και T είναι μη γραμμική. Δεν υπάρχει W και B έτσι ώστε είναι P* W+B=T για τα τέσσερα σετ των παραπάνω τιμών του P και T.



Σχήμα 29:Γραφική παράσταση της επιφάνειας σφάλματος του Αθροίσματος των Τετραγωνικών Λαθών

Η συνάρτηση ERRSURF, όπως είδαμε, υπολογίζει σφάλματα για ένα νευρώνα με μια κλίμακα πιθανών τιμών βάρους και bias. Η συνάρτηση PLOTES σχεδιάζει τις ισοϋψείς της επιφάνεια σφάλματος με ένα περιγραφικό διάγραμμα από κάτω.

$>> w\_range = -2:0.4:2;$ (4.10)

$>> ES = errsurf(P,T,w\_range,b\_range,'purelin');$ (4.11)

$>> plotes(w\_range,b\_range,ES);$ (4.12)

Οι άριστες τιμές βάρους και του bias είναι αυτές που καταλήγουν στο ελάχιστο σημείο της επιφάνειας σφάλματος. Σημειώστε ότι επειδή μια τέλεια γραμμική προσαρμογή δεν είναι δυνατή, η ελάχιστη τιμή αντιστοιχεί σε έχει ένα σφάλμα μεγαλύτερο του μηδενός.

Η συνάρτηση MAXLINLR βρίσκει το γρηγορότερο δυνατό ρυθμό μάθησης για να εκπαιδεύσει ένα γραμμικό δίκτυο.

$>> \max lr = \max linlr(P,'bias');$ (4.13)

Η NEWLIN δημιουργεί ένα γραμμικό νευρώνα

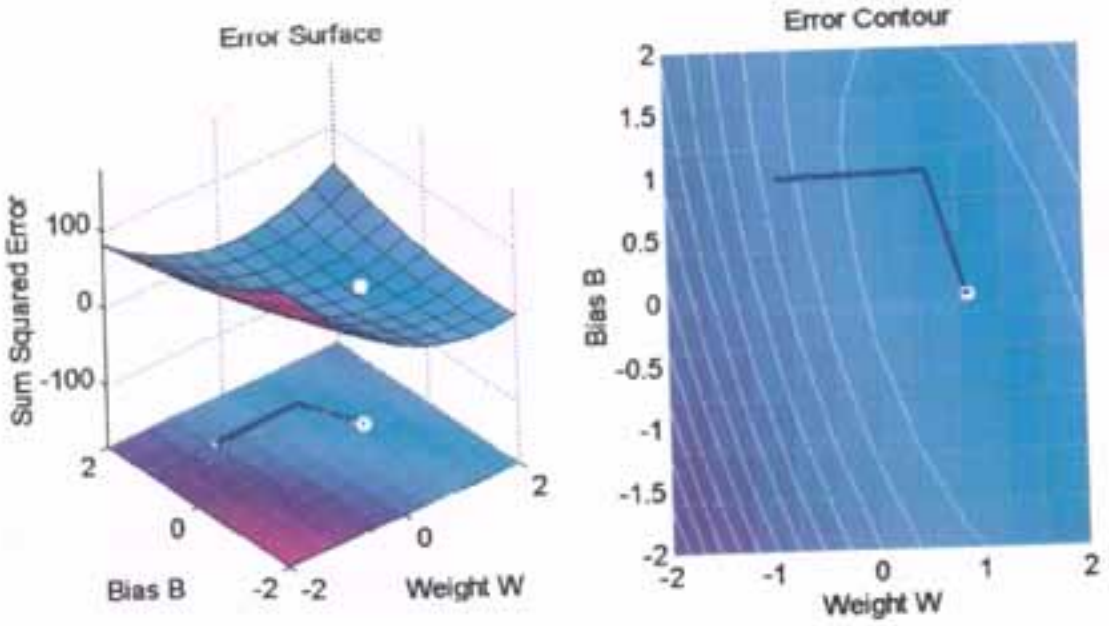
$>> net = newlin([-2\ 2],1,[0],\max lr);$ (4.14)

Παραμερίζουμε/ εξουδετερώνουμε τα λάθη των παραμέτρων εκπαίδευσης δίνοντας τον μέγιστο αριθμό περιόδων. Αυτό επιβεβαιώνει ότι η εκπαίδευση θα σταματήσει:

$>> net.trainParam.epochs = 15$ (4.15)

Χρησιμοποιούμε το PLOTES για να ξανασχεδιάσουμε την επιφάνεια σφάλματος:

$>> plotes[w\_range, b\_range, ES];$ (4.16)



Σχήμα 30: Γραφική παράσταση της επιφάνειας σφάλματος του Αθροίσματος των Τετραγωνικών Λαθών

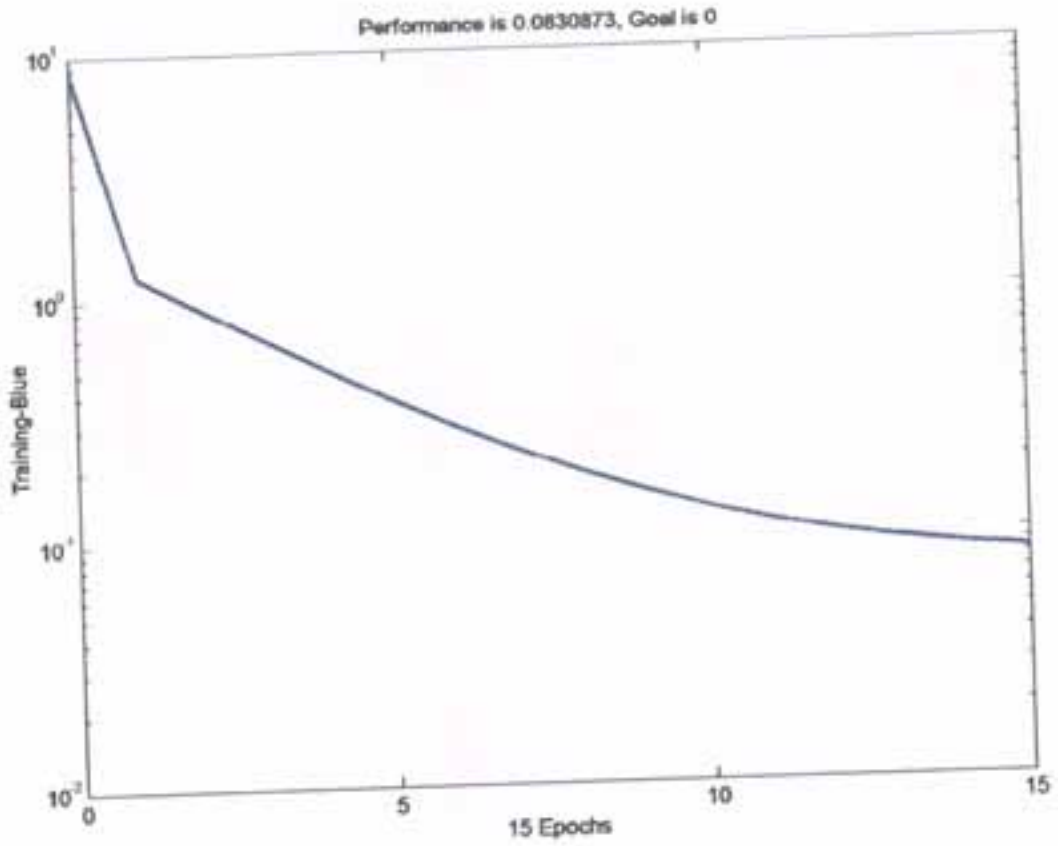
Παραμερίζουμε/ εξουδετερώνουμε την τυχαία δημιουργία βάρους και bias κάνοντας ένα κλικ στο περιγραφικό διάγραμμα σφάλματος με το GINPUT

$>> subplot[1,2,2];[net.IW\{1,1\} net.b\{1\} = ginput(1)$ (4.17)

Για να δείξουμε το μονοπάτι της εκπαίδευσης, εκπαιδεύουμε μόνο μια περίοδο κάθε φορά και αποκαλούμε PLOTEP κάθε εποχή (ο κώδικας δεν φαίνεται εδώ)

$>> net = train[net, P, T];$ (4.18)

Το διάγραμμα δείχνει την ιστορία της εκπαίδευσης. Κάθε τελεία αντιπροσωπεύει μια περίοδο και οι μπλε γραμμές δείχνουν κάθε αλλαγή που έγινε από τον κανόνα μάθησης (με παραβίαση του Windrow- Hoff).



Σχήμα 31: Διάγραμμα της εκπαιδευτικής διαδικασίας

Η εξίσωση εκπαίδευσης έχει ως έξοδο το εκπαιδευμένο δίκτυο και την ιστορία της διαδικασίας εκπαίδευσης(tr). Εδώ τα σφάλματα έχουν σχεδιαστεί σε σχέση στις περιόδους εκπαίδευσης:

$>> plotperf(tr, nettrainParam.goal);$ (4.19)

Ας σημειώσουμε εδώ ότι το σφάλμα δεν φτάνει ποτέ στο μηδέν. Αυτό το πρόβλημα είναι μη γραμμικό και γι' αυτό ένα μηδενικό σφάλμα γραμμικής λύσης δεν είναι δυνατό.

Τώρα χρησιμοποιούμε τη SIM για να ελέγξουμε κατά πόσο το δίκτυο επιστρέφει την τιμή στόχο 1, όταν δώσουμε σαν αρχικές εισόδους -1.2.

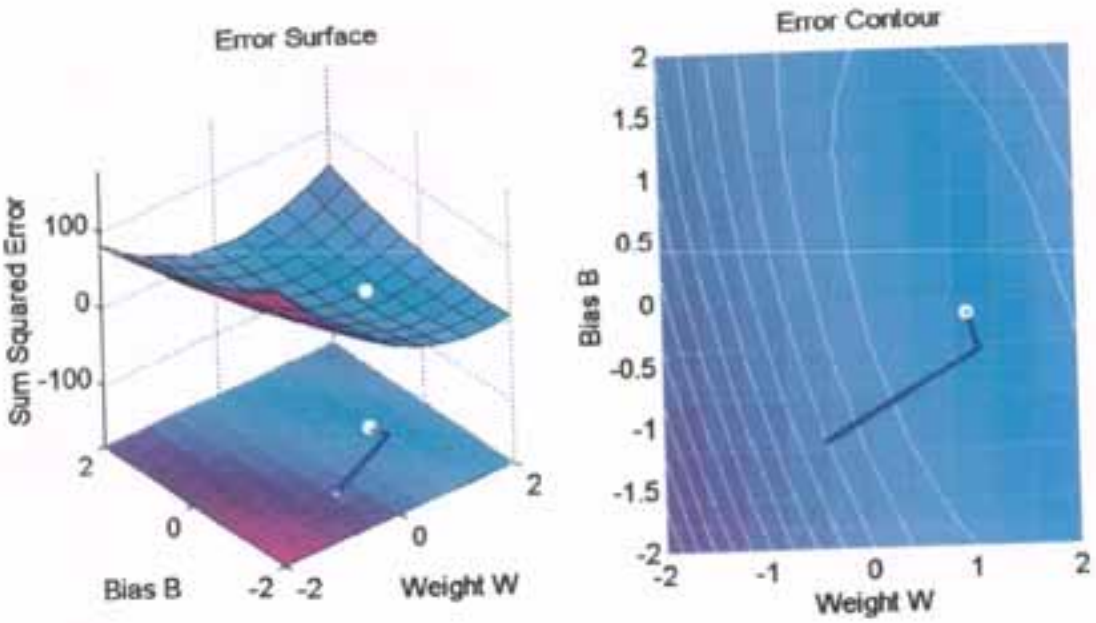
$>> p = -1.2;$ (4.20)

$>> a = sim(net, p);$ (4.21)

Το αποτέλεσμα που προκύπτει είναι: $a = -1.0444$ (4.22)

Το αποτέλεσμα δεν είναι πολύ κοντά στο 0,5. Όταν συμβαίνει αυτό το δίκτυο είναι η καλύτερη γραμμική προσαρμογή σε ένα μη γραμμικό πρόβλημα.

Τώρα κάνουμε κλικ χαμηλά στο περιγραφικό διάγραμμα σφάλματος.



Σχήμα 32: Γραφική παράσταση της επιφάνειας σφάλματος του Αθροίσματος των Τετραγωνικών Λαθών

Το σχεδιάγραμμα αλλάζει όπως και το α, που τώρα γίνεται: $a = -1.281$, που αποτελεί και την καλύτερη γραμμική προσαρμογή σε ένα μη γραμμικό πρόβλημα.

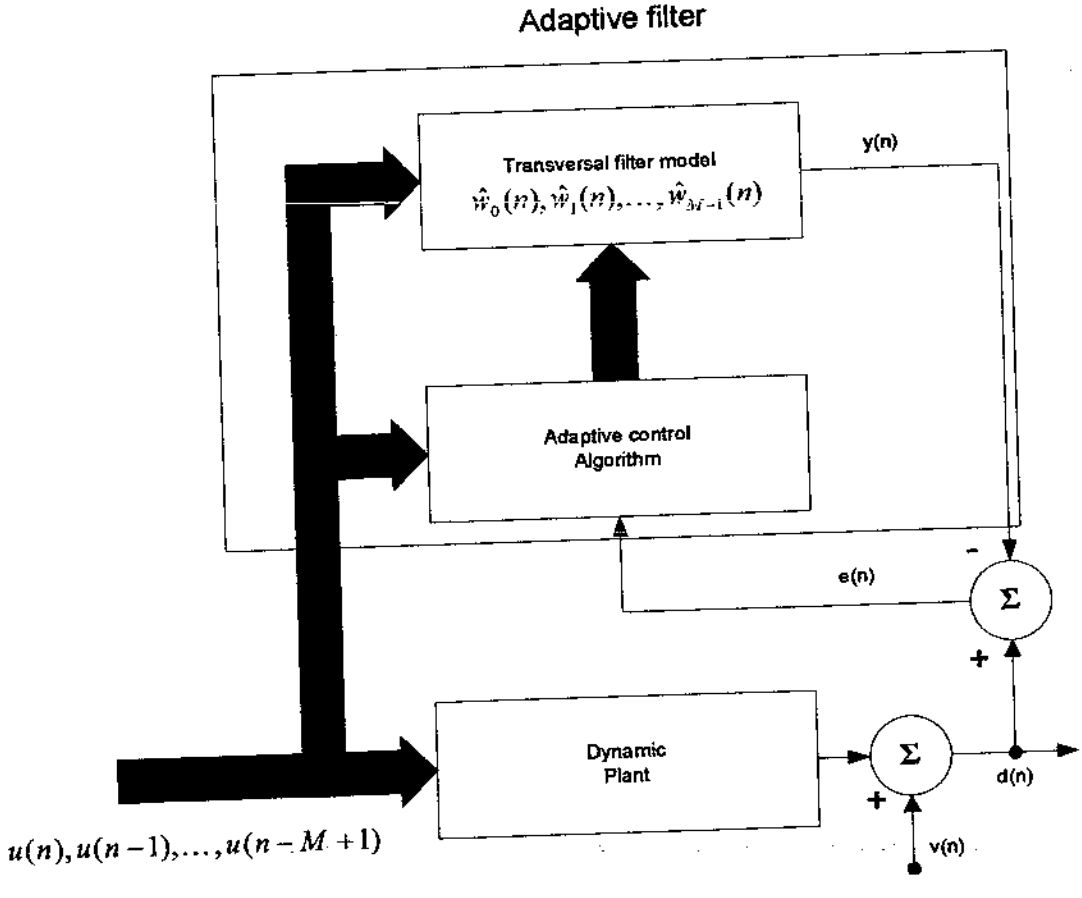
4.3 Προσαρμοστικές μέθοδοι

4.3.1Γενική περιγραφή

Ένα προσαρμοσμένο φίλτρο μπορεί να χρησιμοποιηθεί στην μοντελοποίηση, δηλαδή, να μιμηθεί την συμπεριφορά φυσικών δυναμικών συστημάτων που μπορούν να θεωρηθούν ως άγνωστα μαύρα κουτιά (black boxes) και που έχουν μία ή πολλές εισόδους και μία ή πολλές εξόδους. Η μοντελοποίηση ενός δυναμικού συστήματος μιας εισόδου-μιας εξόδου, φαίνεται στο Σχήμα 33. Το άγνωστο σύστημα καθώς και το προσαρμοσμένο φίλτρο έχουν την ίδια είσοδο. Το προσαρμοσμένο φίλτρο προσαρμόζει τον εαυτό του με σκοπό να κάνει την έξοδό του να μοιάζει όσο το δυνατόν περισσότερο μ' αυτή του άγνωστου συστήματος, γενικότερα να κάνει την έξοδό του να ταιριάζει μ' αυτή του συστήματος κατά την έννοια των ελαχίστων τετραγώνων. Εάν υπάρχει αρκετή ευελιξία στο προσαρμοσμένο σύστημα, και αν είναι επαρκώς προσαρμοζόμενο και περιέχει αρκετούς «βαθμούς ελευθερίας», μπορεί να υπάρξει μια σύγκλιση του συστήματος και του φίλτρου ή ακόμα και ένα τέλειο ταίριασμα. Κατά την σύγκλιση η δομή και οι τιμές των παραμέτρων του προσαρμοζόμενου συστήματος μπορεί ή δεν μπορεί να μοιάζουν μ' αυτές του άγνωστου συστήματος, αλλά η σχέση εισόδου-εξόδου θα μοιάζουν. Μ' αυτήν την έννοια το προσαρμοζόμενο σύστημα γίνεται ένα μοντέλο του

άγνωστου συστήματος. Εάν η είσοδος έχει ευρύ φάσμα τιμών και αν η δομή του προσαρμοστικού συστήματος είναι τέτοια ώστε όταν οι προσαρμοζόμενες παράμετροι επιλέγονται κατάλληλα είναι δυνατόν να υπάρξει ένα ακριβές ταίριασμα, και προσαρμόζοντας το σύστημα ώστε να ελαχιστοποιήσει το μέσο τετραγωνικό σφάλμα θα προκαλέσει ένα ακριβές ταίριασμα σε κάθε λεπτομέρεια.

Αν υποτεθεί ότι υπάρχει ένα άγνωστο σύστημα το οποίο είναι γραμμικό και μεταβάλλεται χρονικά. Το σύστημα χαρακτηρίζεται από ένα πραγματικό σύνολο τιμών από μετρήσεις τιμών διακριτού χρόνου το οποίο περιγράφει τις μεταβολές των εξόδων του συστήματος ως αποτέλεσμα μιας γνωστής και στάσιμης εισόδου. Ο στόχος είναι να κατασκευαστεί ένα on-line εγκάρσιο φίλτρο γι' αυτό το σύστημα όπως περιγράφεται στο Σχήμα 33. Το μοντέλο αποτελείται από ένα πεπερασμένο αριθμό από στοιχεία καθυστέρησης και από ένα αντίστοιχο σύνολο από μεταβαλλόμενες παραμέτρους (tap-weights).



Σχήμα 33: Αναγνώριση συστήματος

Έστω ότι το διαθέσιμο σήμα εισόδου την χρονική στιγμή n συμβολίζεται από το σύνολο των δειγμάτων: $u(n), u(n-1), \ldots, u(n-M+1)$, όπου M είναι ο αριθμός των μεταβαλλόμενων

παραμέτρων του μοντέλου. Αυτό το σήμα εισόδου εφαρμόζεται ταυτόχρονα στο σύστημα προς αναγνώριση και στο μοντέλο. Οι αντίστοιχες έξοδοι του συστήματος και του μοντέλου συμβολίζονται με *d(n)* και *y(n)*. Η έξοδος του συστήματος *d(n)* εξυπηρετεί τον σκοπό της επιθυμητής απόκρισης του προσαρμοζόμενου αλγορίθμου που χρησιμοποιείται για την ρύθμιση των παραμέτρων του μοντέλου. Η έξοδος του μοντέλου δίνεται από την ακόλουθη σχέση:

$$y(n) = \sum_{k=0}^{M-1} \hat{w}_k(n) u(n-k) \quad (4.23)$$

όπου τα $\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)$ είναι οι εκτιμούμενες παράμετροι του μοντέλου. Η έξοδος του μοντέλου *y(n)* συγκρίνεται με την έξοδο του συστήματος *d(n)*. Η διαφορά μεταξύ τους, *d(n)-y(n)*, ορίζει το λάθος εκτίμησης του μοντέλου (model estimation error). Το λάθος αυτό συμβολίζεται με *e(n)*.

Τυπικά, την χρονική στιγμή *n*, το λάθος εκτίμησης *e(n)* είναι μη μηδενικό, υπονοώντας ότι το μοντέλο αποκλίνει από το σύστημα. Σε μια προσπάθεια να περιοριστεί αυτή η απόκλιση, το λάθος *e(n)* εφαρμόζεται σε έναν προσαρμοζόμενο αλγόριθμο ελέγχου. Τα δείγματα του σήματος εισόδου, $u(n), u(n-1), \ldots, u(n-M+1)$, επίσης εφαρμόζονται σ' αυτόν τον αλγόριθμο. Ο συνδυασμός του εγκάρσιου φίλτρου μαζί με τον προσαρμοστικό αλγόριθμο ελέγχου κάνουν τον προσαρμοστικό αλγόριθμο φιλτραρίσματος (adaptive filtering algorithm). Ο αλγόριθμος σχεδιάζεται για να ελέγχει τις αλλαγές που γίνονται στις τιμές των παραμέτρων του μοντέλου. Σαν αποτέλεσμα, οι παράμετροι του μοντέλου παράγουν ένα νέο σύνολο από τιμές που χρησιμοποιούνται στην επόμενη επανάληψη του αλγορίθμου. Συνεπώς, την χρονική στιγμή *n+1*, υπολογίζεται η έξοδος του νέου μοντέλου και μαζί μ' αυτή μια νέα τιμή για το λάθος εκτίμησης. Η λειτουργία που περιγράφηκε παραπάνω επαναλαμβάνεται στην συνέχεια. Αυτή η διαδικασία συνεχίζεται για ένα αρκετά μεγάλο αριθμό από επαναλήψεις (ξεκινώντας από την χρονική στιγμή *n=0*), μέχρι η απόκλιση του μοντέλου από το σύστημα προς αναγνώριση, μετρημένη από το πλάτος του λάθους εκτίμησης, να γίνει σημαντικά μικρή σύμφωνα με κάποια στατιστική έννοια.

Σε πολλές πρακτικές περιπτώσεις, το σύστημα που πρόκειται να μοντελοποιηθεί περικλείει και θόρυβο, δηλαδή εσωτερικά στό σύστημα δρουν τυχαίες δυνάμεις που αλλάζουν τις τιμές της εξόδου. Σ' αυτές τις περιπτώσεις όταν το προσαρμοστικό σύστημα έχει αρκετή ευελιξία ώστε να ταιριάζει με την δυναμική απόκριση του άγνωστου συστήματος, η έξοδός του θα μοιάζει απόλυτα με αυτή του άγνωστου συστήματος αν αφαιρεθεί ο προσθετικός θόρυβος *v(n)* όπως φαίνεται στο Σχήμα 33. Ο εσωτερικός θόρυβος του συστήματος φαίνεται στην έξοδό του και συχνά αναπαριστάται σαν να ήταν προσθετικός θόρυβος. Αυτός ο θόρυβος

είναι γενικά ασυσχέτιστος με την είσοδο του συστήματος. Σε μια τέτοια περίπτωση αν το προσαρμοστικό μοντέλο είναι ένα γραμμικό σύστημα του οποίου τα βάρη προσαρμόζονται ώστε να ελαχιστοποιήσουν το *μέσο τετραγωνικό σφάλμα*, μπορεί να δειχθεί ότι η λύση των ελαχίστων τετραγώνων δεν θα επηρεαστεί από την παρουσία του θορύβου του συστήματος. Αυτό δεν σημαίνει ότι η σύγκλιση της προσαρμοστικής διαδικασίας δεν θα επηρεαστεί από τον θόρυβο του συστήματος αλλά ότι μόνο το εκτιμώμενο διάνυσμα των βαρών του προσαρμοστικού μοντέλου μετά την σύγκλιση δεν θα επηρεαστεί. Η λύση των ελαχίστων τετραγώνων θα πρέπει να καθοριστεί αρχικά από την κρουστική απόκριση του συστήματος που θα μοντελοποιηθεί. Αυτό επίσης μπορεί να επηρεαστεί σημαντικά και από τις στατιστικές ιδιότητες του σήματος εισόδου.

Όταν το σύστημα είναι χρονικά μεταβαλλόμενο, η έξοδος του είναι μη στάσιμη καθώς επίσης και η επιθυμητή απόκριση που παρουσιάζεται από τον προσαρμοστικό αλγόριθμο φιλτραρίσματος. Σε τέτοια περίπτωση, ο αλγόριθμος έχει την αρμοδιότητα όχι μόνο να κρατήσει το λάθος εκτίμησης μικρό αλλά και να παρακολουθεί τις χρονικές μεταβολές των δυναμικών του συστήματος.

4.3.2 Συνάρτηση Μέσου Τετραγωνικού Σφάλματος

Σύμφωνα με την προηγούμενη ενότητα το λάθος εκτίμησης την χρονικής στιγμή n δίνεται από την σχέση:

$$e(n) = d(n) - y(n) \quad (4.24)$$

Επίσης η έξοδος του συστήματος μπορεί να αναπαρασταθεί ως γινόμενο διανυσμάτων ως εξής:

$$y(n) = \mathbf{u}^T(n)\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}^T(n)\mathbf{u}(n) \quad (4.25)$$

όπου $\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$ και $\hat{\mathbf{w}}(n) = [\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)]^T$

Η 4.24 γίνεται με αντικατάσταση της 4.25 σ' αυτή:

$$e(n) = d(n) - \mathbf{u}^T(n)\hat{\mathbf{w}}(n) = d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n) \quad (4.26)$$

Για να εμφανιστεί το μέσο τετραγωνικό σφάλμα η 4.26 υψώνεται στο τετράγωνο και λαμβάνεται η μέση τιμή[2] και στα δύο μέλη της εξίσωσης οπότε γίνεται:

$$E\left[e^2(n)\right] = E\left[d^2(n)\right] + \hat{\mathbf{w}}^T\mathbf{R}\hat{\mathbf{w}} - 2\mathbf{p}^T\hat{\mathbf{w}} \quad (4.27)$$

όπου $\boldsymbol{R}$ είναι ο πίνακας αυτοσυσχέτισης των δεδομένων εισόδου και ισούται με

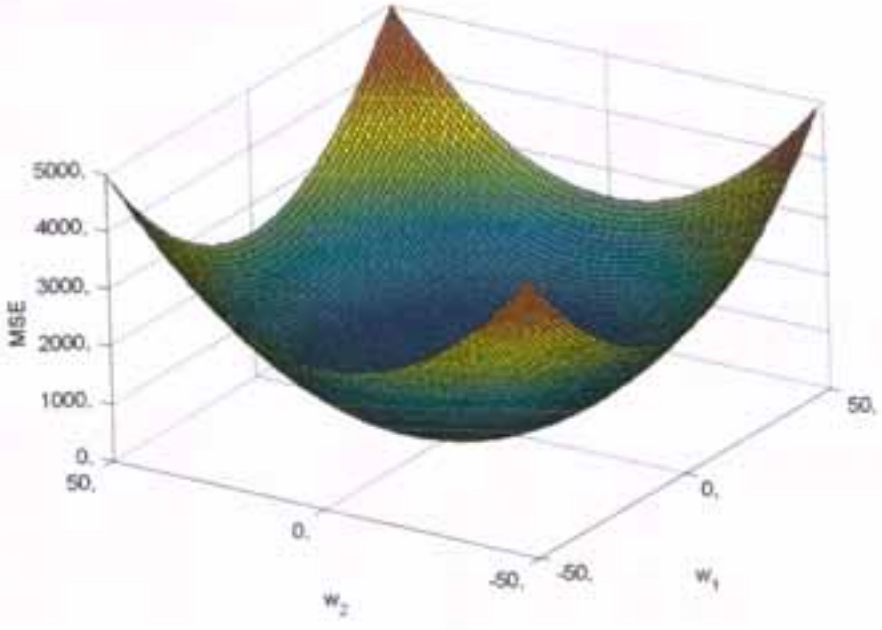
$$\mathbf{R} = E\left[\mathbf{u}(n)\mathbf{u}^T(n)\right] \quad (4.28)$$

[2] Τα $\varepsilon(n)$, $d(n)$ και $U(n)$ θεωρούνται ότι είναι στατιστικά στάσιμες διαδικασίες

και $\boldsymbol{p}$ είναι το διάνυσμα ετεροσυσχέτισης των δεδομένων εισόδου και εξόδου του συστήματος και ισούται με:

$$\mathbf{p} = E\left[d(n)\mathbf{u}(n)\right] \quad (4.29)$$

Έτσι λοιπόν από την εξίσωση 4.27 το μέσο τετραγωνικό σφάλμα μπορεί να αναπαρασταθεί ως συνάρτηση των παραμέτρων του μοντέλου καθώς και των στατιστικών ιδιοτήτων των δεδομένων της εισόδου και της εξόδου του συστήματος. Στο Σχήμα 34 φαίνεται μια δισδιάστατη απεικόνιση της συνάρτησης μέσου τετραγωνικού σφάλματος. Ο κάθετος άξονας αναπαριστά το μέσο τετραγωνικό σφάλμα και οι οριζόντιοι τις τιμές των δύο παραμέτρων του μοντέλου. Από την εικόνα αυτή φαίνεται ότι η συνάρτηση έχει σχήμα παραβολοειδές και στην περίπτωση περισσοτέρων από 2 παραμέτρων υπερ-παραβολοειδές. Η επιφάνεια αυτή ονομάζεται και *επιφάνεια σφάλματος (error surface)* και με βάση αυτή οι προσαρμοστικοί αλγόριθμοι προσπαθούν να βρουν τις βέλτιστες παραμέτρους των μοντέλων ώστε να ελαχιστοποιείται το μέσο τετραγωνικό σφάλμα και να προσεγγίζεται όσο το δυνατόν καλύτερα το σύστημα απ' αυτό.



Σχήμα 34: Επιφάνεια μέσου τετραγωνικού σφάλματος

4.3.3 Μέθοδοι ανίχνευσης της επιφάνειας σφάλματος

Για την εύρεση του ελάχιστου της εξίσωση (4.27) η σχέση αυτή παραγωγίζεται και η λύση που προκύπτει είναι η επίλυση του συστήματος των εξισώσεων Wiener-Hoff:

$$\mathbf{R}\hat{\mathbf{w}}_{opt} = \mathbf{p} \quad (4.30)$$

όπου $\hat{\mathbf{w}}_{opt}$ είναι το διάνυσμα που περιέχει τις παραμέτρους που ελαχιστοποιούν το μέσο τετραγωνικό σφάλμα της εξίσωσης 4.27.

Η επίλυση του συστήματος της σχέσης 4.30 είναι μια διαδικασία η οποία περιέχει αρκετές υπολογιστικές δυσκολίες, ειδικά όταν το μοντέλο αποτελείται από μεγάλο αριθμό παραμέτρων και ο ρυθμός δεδομένων εισόδου είναι μεγάλος.

Για την αποφυγή των προβλημάτων που επιφέρει η επίλυση του συστήματος υπάρχουν επαναληπτικές μέθοδοι οι οποίες κατορθώνουν και βρίσκουν μια αρκετά καλή προσέγγιση του ελάχιστου τετραγωνικού σφάλματος. Αυτές οι μέθοδοι καλούνται *gradient* και χρησιμοποιούν μια εκτίμηση της κλίσης της επιφάνειας σε κάθε επανάληψη για να βρουν το ελάχιστό της. Τέτοιες μέθοδοι είναι η μέθοδος ανίχνευσης του *Newton* και η *steepest descent.*

Η μέθοδος του Newton αν και έχει μεγάλη μαθηματική σημασία είναι δύσκολη να υλοποιηθεί στην πράξη. Είναι μια μέθοδος η οποία προκαλεί την αλλαγή των παραμέτρων του μοντέλου σε κάθε βήμα της διαδικασίας ανίχνευσης. Οι αλλαγές των παραμέτρων γίνονται στην κατεύθυνση του ελάχιστου της επιφάνειας σφάλματος. Για την δημιουργία σωστών αποτελεσμάτων θα πρέπει αυτή η επιφάνεια να έχει μοναδικό ελάχιστο.

Η μέθοδος της steepest descent είναι εύκολα υλοποιήσιμη και έχει αποδείξει την αξία της σε μια ποικιλία από πρακτικές εφαρμογές. Όπως και η Newton είναι μια μέθοδος gradient search η οποία αλλάζει τις παραμέτρους του μοντέλου σε κάθε βήμα. Η διαφορά με την προηγούμενη μέθοδο είναι ότι αυτή την φορά οι αλλαγές γίνονται στην κατεύθυνση της αρνητικής κλίσης της επιφάνειας σφάλματος. Αυτό δεν σημαίνει απαραίτητα ότι κατευθύνεται στο ελάχιστο της επιφάνειας, αφού η αρνητική κλίση τείνει στο ελάχιστο μόνο όταν ξεκινά από έναν από τους κύριους άξονες της επιφάνειας.

Βέβαια υπάρχουν και άλλες μέθοδοι για την εύρεση του βέλτιστου της επιφάνειας σφάλματος οι οποίες λειτουργούν καλύτερα ανά περίπτωση. Δύο απ' αυτές που θα παρουσιαστούν παρακάτω στο κείμενο είναι η μέθοδος των ελαχίστων τετραγώνων (LMS) και το γενετική ανίχνευση μία κατηγορία των τυχαίων μεθόδων ανίχνευσης.

4.3.4 Η Μέθοδος της Steepest-Descent

Η μέθοδος της steepest-descent είναι μία από τις παλαιότερες μεθόδους βελτιστοποίησης. Για να βρεθεί η ελάχιστη τιμή του μέσου τετραγωνικού σφάλματος, J_{min}, με την μέθοδο της steepest-descent ακολουθείται η παρακάτω διαδικασία:

1. Πρώτα ξεκινά με μια αρχική τιμή $w(0)$ των παραμέτρων, η οποία παρέχει μια αρχική εκτίμηση στο πού θα βρίσκεται το σημείου του ελάχιστου της επιφάνειας σφάλματος. Αν δεν υπάρχει κάποια εκ των προτέρων γνώση για την αρχική τιμή του $w(0)$, χρησιμοποιείται το μηδενικό διάνυσμα.
2. Χρησιμοποιώντας την αρχική ή την παρούσα εκτίμηση του διανύσματος w, υπολογίζεται το διάνυσμα κλίσης (gradient vector), το πραγματικό και το φανταστικό μέρος του οποίου ορίζονται ως η παράγωγος του μέσου τετραγωνικού σφάλματος $J(n)$, που υπολογίζεται έχοντας υπόψη το πραγματικό και φανταστικό μέρος του διανύσματος των παραμέτρων $w(n)$ την στιγμή n.
3. Στην συνέχεια υπολογίζεται η επόμενη εκτίμηση του διανύσματος των παραμέτρων με το να αλλάξει η παρούσα εκτίμηση στην αντίθετη κατεύθυνση του διανύσματος κλίσης.
4. Τέλος η διαδικασία επαναλαμβάνεται πηγαίνοντας στο βήμα 2.

Είναι αρκετά λογικό ότι οι διαδοχικές διορθώσεις των παραμέτρων του διανύσματος στην κατεύθυνση του αρνητικού διανύσματος κλίσης της επιφάνειας θα πρέπει να οδηγούν στο ελάχιστο μέσο τετραγωνικό σφάλμα J_{min} όπου σ' αυτό το σημείο το διάνυσμα των παραμέτρων προσεγγίζει το w_{opt}.

Έστω $\nabla J(n)$ να δηλώνει την τιμή του διανύσματος κλίσης στην χρονική στιγμή n. Έστω $w(n)$ είναι η τιμή του διανύσματος των παραμέτρων του μοντέλου την χρονική στιγμή n. Σύμφωνα με την μέθοδο του steepest descent η ενημερωμένη τιμή του διανύσματος των παραμέτρων την χρονική στιγμή $n+1$ υπολογίζεται χρησιμοποιώντας την απλή αναδρομική σχέση:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu\left[-\nabla J(n)\right] \quad (4.31)$$

όπου το μ είναι μια θετική αρνητική σταθερά. Ο παράγοντας ½ χρησιμοποιείται για την ακύρωση του παράγοντα 2 που εμφανίζεται στην εξίσωση για το $\nabla J(n)$.

Το διάνυσμα κλίσεως δίνεται από την εξίσωση:

$$\nabla J(n) = \begin{bmatrix} \frac{\partial J(n)}{\partial a_0(n)} + j\frac{\partial J(n)}{\partial b_0(n)} \\ \frac{\partial J(n)}{\partial a_1(n)} + j\frac{\partial J(n)}{\partial b_1(n)} \\ \vdots \\ \frac{\partial J(n)}{\partial a_{M-1}(n)} + j\frac{\partial J(n)}{\partial b_{M-1}(n)} \end{bmatrix} \quad (4.32)$$
$$= -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)$$

όπου, εσωτερικά του διανύσματος τα $\partial J(n)/\partial a_k(n)$ και $\partial J(n)/\partial b_k(n)$ είναι οι μερικές παράγωγοι της συνάρτησης σφάλματος *J(n)* ως προς το πραγματικό μέρος $a_k(n)$ και το φανταστικό μέρος $b_k(n)$ της k-οστής παραμέτρου $w_k(n)$, *k=1,2,...,M-1*. Για την εφαρμογή του αλγόριθμου steepest-descent, από την εξίσωση 4.32 ο πίνακας αυτοσυσχέτισης ***R*** και το διάνυσμα ετεροσυσχέτισης ***p*** είναι γνωστά έτσι ώστε να υπολογιστεί το διάνυσμα κλίσης $\nabla J(n)$ για μια συγκεκριμένη τιμή του διανύσματος *w(n)*. Έτσι, αντικαθιστώντας την εξίσωση 4.32 στην εξίσωση 4.31, υπολογίζεται η νέα τιμή του διανύσματος παραμέτρων *w(n+1)* σύμφωνα με την εξίσωση:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\left[\mathbf{p} - \mathbf{R}\mathbf{w}(n)\right], \quad n = 1, 2, \ldots \quad (4.33)$$

Η σταθερά μ ελέγχει το μέγεθος της διόρθωσης που εφαρμόζεται στο διάνυσμα των παραμέτρων καθώς προχωράει από το ένα βήμα στο επόμενο. Η σταθερά αυτή αναφέρεται και ως *παράμετρος μεγέθους βήματος* (*step-size parameter*). Η εξίσωση 4.33 περιγράφει την μαθηματική τυποποίηση του αλγόριθμου του steepest-descent.

Σύμφωνα με την εξίσωση 4.33 η διόρθωση *δw(n)* που εφαρμόζεται στο διάνυσμα των παραμέτρων την χρονική στιγμή *n+1* ισούται με $\mu\left[\mathbf{p} - \mathbf{R}\mathbf{w}(n)\right]$. Αυτή η διόρθωση μπορεί να εκφραστεί ως μ φορές την μέση τιμή του εσωτερικού γινομένου του διανύσματος εισόδου *u(n)* και του σφάλματος εκτίμησης *e(n)*.

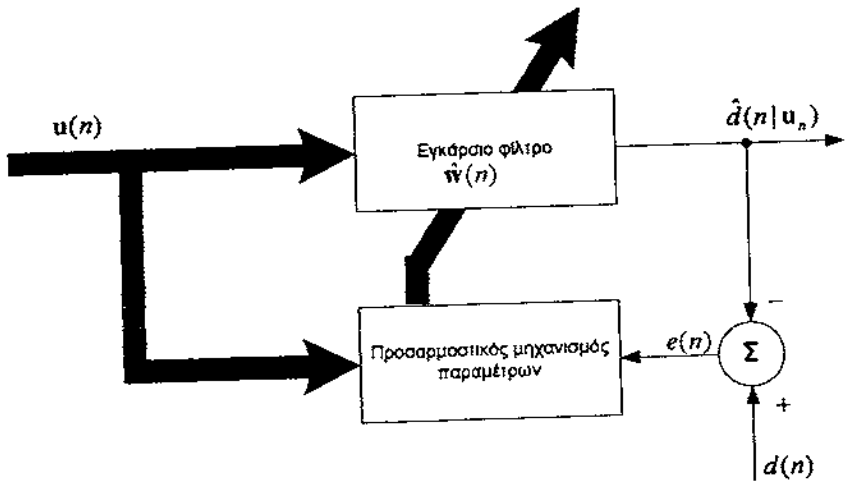
4.3.5 Ο Αλγόριθμος LMS

Ο αλγόριθμος LMS (least mean squares) είναι μια παραλλαγή του αλγορίθμου steepest descent ο οποίος ανήκει στην κατηγορία των *stochastic gradient αλγορίθμων*. Ο όρος stochastic gradient χρησιμοποιείται για να ξεχωρίσει τον LMS αλγόριθμο από την μέθοδο του steepest descent η οποία χρησιμοποιεί την ακριβή κλίση της επιφάνειας λάθους σε αντίθεση με τον LMS οποίος χρησιμοποιεί μια εκτίμηση της.

Ο αλγόριθμος LMS είναι ένας γραμμικός προσαρμοστικός αλγόριθμος φιλτραρίσματος ο οποίος αποτελείται από δύο βασικές διαδικασίες:

1. Μια διαδικασία φιλτραρίσματος, η οποία περιλαμβάνει
   i. τον υπολογισμό της εξόδου του φίλτρου που παράγεται από ένα σύνολο παραμέτρων
   ii. την δημιουργία ενός σφάλματος εκτίμησης που βγαίνει από την σύγκριση της εξόδου με την επιθυμητή απόκριση.
2. Μια διαδικασία προσαρμογής, η οποία περιλαμβάνει την αυτοματοποιημένη ρύθμιση των παραμέτρων του φίλτρου με συμφωνία ως προς το σφάλμα εκτίμησης.

Ο συνδυασμός αυτών των διαδικασιών σχηματίζει ένα βρόγχο ανάδρασης (feedback loop) γύρω από τον LMS αλγόριθμο, όπως φαίνεται στο Σχήμα 35. Πρώτα υπάρχει ένα εγκάρσιο φίλτρο γύρω από το οποίο φτιάχνεται ο αλγόριθμος LMS, αυτή η συνιστώσα ευθύνεται για την εκτέλεση της διαδικασίας φιλτραρίσματος. Έπειτα, υπάρχει ένας μηχανισμός για την εκτέλεση της διαδικασίας προσαρμογής των παραμέτρων του φίλτρου, γι' αυτό και η επωνυμία «προσαρμοστικός μηχανισμός παραμέτρων» στο Σχήμα 35.



Σχήμα 35: *Μπλοκ διάγραμμα προσαρμοστικού εγκάρσιου φίλτρου*

Αν ήταν δυνατό να έβγαιναν ακριβές μετρήσεις του διανύσματος κλίσης $\nabla J(n)$ για κάθε επανάληψη n και αν η παράμετρος βήματος μ ήταν κατάλληλα επιλεγμένη, τότε το διάνυσμα των παραμέτρων που θα υπολογιζόταν με τη μέθοδο του steepest descent θα σύγκλινε στη βέλτιστη Wiener λύση. Στην πραγματικότητα, ωστόσο, ακριβές μετρήσεις του διανύσματος κλίσης δεν είναι δυνατές γιατί απαιτούν από πριν γνώση του πίνακα αυτοσυσχέτισης $\boldsymbol{R}$ των

δεδομένων της εισόδου και του διανύσματος ετεροσυσχέτισης $\boldsymbol{p}$ των δεδομένων της εισόδου και της επιθυμητής εξόδου. Συνεπώς, το διάνυσμα κλίσης θα πρέπει να εκτιμηθεί από τα διαθέσιμα δεδομένα.

Για να αναπτυχθεί μια εκτίμηση του διανύσματος κλίσης $\nabla J(n)$, η πιο προφανής στρατηγική είναι να αντικατασταθεί με τις εκτιμώμενες τιμές του πίνακα αυτοσυσχέτισης $\boldsymbol{R}$ και του πίνακα ετεροσυσχέτισης $\boldsymbol{p}$ η εξίσωση 4.32, η οποία φαίνεται και εδώ για ευκολία:

$$\nabla J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n) \quad (4.34)$$

Η απλούστερη επιλογή των εκτιμητών για το $\boldsymbol{R}$ και το $\boldsymbol{p}$ είναι να χρησιμοποιηθούν στιγμιαίες εκτιμήσεις οι οποίες βασίζονται στα δείγματα των διανυσμάτων εισόδου και επιθυμητής απόκρισης, οι παρακάτω εξισώσεις ορίζουν τους εκτιμητές αυτούς:

$$\hat{\mathbf{R}}(n) = \mathbf{u}(n)\mathbf{u}^H(n) \quad (4.35)$$

και

$$\hat{\mathbf{p}}(n) = \mathbf{u}(n)d^*(n) \quad (4.36)$$

Κατά συνέπεια, η στιγμιαία εκτίμηση του διανύσματος κλίσης είναι:

$$\hat{\nabla} J(n) = -2\mathbf{u}(n)d^*(n) + 2\mathbf{u}(n)\mathbf{u}^H(n)\hat{\mathbf{w}}(n) \quad (4.37)$$

Αντικαθιστώντας την εκτίμηση της εξίσωσης 4.37 για το διάνυσμα κλίσης $\nabla J(n)$ στον αλγόριθμο steepest descent που περιγράφηκε στην προηγούμενη εξίσωση 4.31, δημιουργείται μια νέα σχέση για την διόρθωση του διανύσματος των παραμέτρων του φίλτρου:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)\left[d^*(n) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n)\right] \quad (4.38)$$

Εδώ χρησιμοποιείται το σύμβολο ^ πάνω από το διάνυσμα των παραμέτρων για να ξεχωριστεί από το διάνυσμα των παραμέτρων που υπολογίζεται από την μέθοδο του steepest descent. Ισοδύναμα, οι παραπάνω σχέσεις μπορούν να γραφούν στην μορφή των ακόλουθων τριών βασικών σχέσεων:

1. Έξοδος του φίλτρου:

$$y(n) = \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \quad (4.39)$$

2. Σφάλμα εκτίμησης:

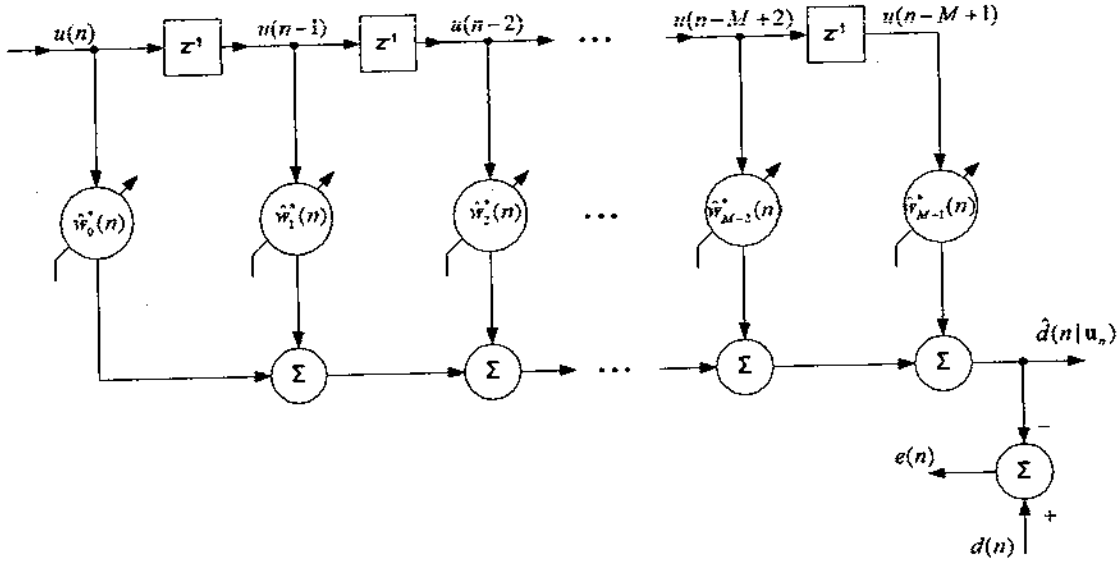
$$e(n) = d(n) - y(n) \quad (4.40)$$

3. Προσαρμογή των παραμέτρων:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n) \quad (4.41)$$

Οι εξισώσεις 4.39,4.40 ορίζουν το σφάλμα εκτίμησης *e(n)*, ο υπολογισμός του οποίου βασίζεται στην τρέχουσα εκτίμηση του διανύσματος παραμέτρων $\hat{\mathbf{w}}(n)$. Επίσης ο όρος

$\mu \mathbf{u}(n)e^*(n)$, στο δεξί μέρος της εξίσωσης 4.41, αναπαριστά την διόρθωση η οποία εφαρμόζεται στην τρέχουσα εκτίμηση του διανύσματος παραμέτρων $\hat{w}(n)$. Η επαναληπτική διαδικασία ξεκινά με μια αρχική εκτίμηση $\hat{\mathbf{w}}(0)$.

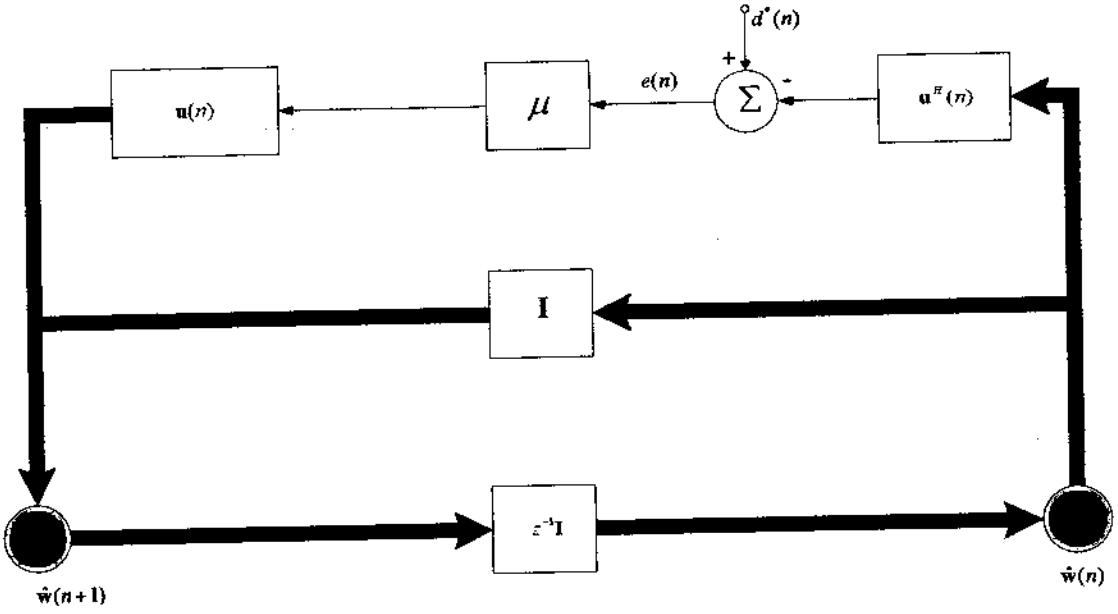


Σχήμα 36: Λεπτομερής δομή του εγκάρσιου φίλτρου

Ο αλγόριθμος που περιγράφεται από τις εξισώσεις 4.39 έως 4.41, είναι η μιγαδική μορφή του προσαρμοστικού αλγόριθμου των ελάχιστων μέσων τετραγώνων (least-mean-square LMS). Σε κάθε επανάληψη ή χρονική στιγμή, χρειάζεται γνώση των πιο πρόσφατων τιμών: *u(n)*, *d(n)* και $\hat{w}(n)$. Ο LMS αλγόριθμος είναι μέλος των stochastic gradient αλγόριθμων. Ειδικότερα, όταν ο LMS αλγόριθμος λειτουργεί σε στοχαστικές εισόδους, το επιτρεπόμενο σύνολο των κατευθύνσεων κατά μήκος που μπορεί να μεταβεί από έναν κύκλο επανάληψης στον επόμενο είναι αρκετά τυχαίο έτσι δεν μπορεί να θεωρηθεί ότι δίνει πραγματικές κατευθύνσεις στις κλίσεις.

Το σχήμα 37 δείχνει μια αναπαράσταση του LMS αλγόριθμου στην μορφή ενός μοντέλου ανάδρασης. Το γράφημα του σχήματος 37 δείχνει την απλότητα του αλγόριθμου LMS. Συγκεκριμένα, φαίνεται από την εικόνα ότι ο LMS αλγόριθμος χρειάζεται μόνο *2M+1* μιγαδικούς πολλαπλασιασμούς και *2M* μιγαδικές προσθέσεις ανά επανάληψη, όπου *M* είναι ο αριθμός των παραμέτρων του φίλτρου. Με άλλα λόγια η υπολογιστική πολυπλοκότητα του αλγορίθμου είναι *O(M)*.

Οι στιγμιαίες εκτιμήσεις των $\boldsymbol{R}$ και $\boldsymbol{p}$ των εξισώσεων 4.35 και 4.36 έχουν σχετικά μεγάλες αποκλίσεις. Με την πρώτη ματιά, μπορεί να φαίνεται ότι ο αλγόριθμος LMS να μην είναι ικανός να έχει καλή απόδοση αφού χρησιμοποιεί αυτές τις στιγμιαίες αποκλίσεις. Ωστόσο, πρέπει να ληφθεί υπόψη ότι ο LMS είναι αναδρομικός από την φύση του και με αυτό μπορεί να υπολογίζει ικανοποιητικά αυτές τις εκτιμήσεις κατά την διάρκεια εκτέλεσής του.



Σχήμα 37: Διάγραμμα ροής του αλγόριθμου LMS

4.4 Misadjustment

Το πλεονάζον MSE είναι το μέσο MSE μείον το ελάχιστο MSE. Είναι ένα μέτρο της διαφοράς ανάμεσα στην πραγματική και στη βέλτιστη επίδοση. Ένα άλλο μέτρο της διαφοράς αυτής, εξαιρετικά χρήσιμο είναι το misadjustment M. Αυτό ορίζεται από το "υπερβάλλον" MSE προς το ελάχιστο MSE:

$$\mathrm{M} = \frac{\text{excess MSE}}{\xi_{\min}} \qquad (4.42)$$

Το misadjustment είναι ένας καθαρός αριθμός της διαφοράς ανάμεσα στην προσαρμοστική και στη βέλτιστη επίδοση ως αποτέλεσμα του θορύβου της εκτίμησης της παραγώγου. Παρατηρούμε πως το M δεν περιλαμβάνει την διαταραχή, η οποία οφείλεται στη διακύμανση των βαρών παρά εξαιτίας του θορύβου. (Ταμπάκης Π., Ιούνιος 2004, σελ. 76-77)
Το M στην περίπτωση του Newton θα είναι:

$$M \approx \frac{(L+1)\lambda_{av.}(1/\lambda)_{av.}}{8NP\tau}$$

ή (4.43)

$$M \approx \frac{(L+1)^2\lambda_{av.}(1/\lambda)_{av.}}{8NPT_{MSE}}$$

Ενώ για τη μέθοδο απότομης κατάβασης το misadjustment υπολογίζεται από τον τύπο:

$$M \approx \frac{(L+1)^2}{8P}\left(\frac{1}{T_{MSE}}\right)_{av.} \quad (4.44)$$

5. Matlab

Το MATLAB, ένα υπολογιστικό περιβάλλον στο οποίο μπορούμε να εκτελέσουμε εύκολα και γρήγορα τεχνικούς υπολογισμούς. Ενσωματώνει υπολογισμό, προγραμματισμό και οπτικοποίηση σε ένα ευκολόχρηστο περιβάλλον, όπου τα προβλήματα και οι λύσεις εκφράζονται με οικεία μαθηματική σημειογραφία.

Το όνομα MATLAB είναι τα αρχικά των λέξεων matrix laboratory (εργαστήριο πινάκων). Η MATLAB αρχικά δημιουργήθηκε για να παρέχει εύκολη πρόσβαση στο λογισμικό των πινάκων που αναπτύχθηκε από τα πακέτα LINPACK και EISPACK. Σήμερα, το MATLAB χρησιμοποιεί λογισμικό που αναπτύχθηκε από τα πακέτα LAPACK και ARPACK, που και τα δύο αντιπροσωπεύουν τη κορυφή(state-of-the-art) του λογισμικού πού χρησιμοποιείται σε υπολογισμούς πινάκων.

Το MATLAB έχει εξελιχθεί με την πάροδο του χρόνου αφού χρησιμοποιείται από πολλούς χρήστες. Στην εκπαίδευση και την έρευνα αποτελεί ένα από τα βασικότερα εργαλεία. Στη βιομηχανία, επίσης το MATLAB είναι ένα από τα εργαλεία για υψηλή παραγωγική έρευνα, ανάπτυξη και ανάλυση.

6.Απλές εφαρμογές στο Matlab

Σήμερα, πολλά από τα σημαντικά προβλήματα που δεν μπορούσαν να αντιμετωπιστούν παλαιότερα με πρακτικό τρόπο, μπορούν να αντιμετωπιστούν ικανοποιητικά με τη βοήθεια των νευρωνικών δικτύων. Παρακάτω θα δούμε μερικές εφαρμογές νευρωνικών δικτύων.

6.1 AppLin1: Γραμμική πρόβλεψη

Τα γραμμικά δίκτυα μπορούν να χρησιμοποιηθούν για να προβλέψουν μελλοντικές τιμές των χρονοσειρών. Σε αυτήν την εφαρμογή, σχεδιάζεται, ένα γραμμικό επίπεδο με την βοήθεια της συνάρτησης **solvelin1,** για να προβλέψουμε την επόμενη τιμή ενός σήματος, δεδομένων των προηγούμενων πέντε τιμών του.

Θα χρησιμοποιήσουμε ένα σήμα t το οποίο διαρκεί 5 δευτερόλεπτα και καθορίζεται από ένα ρυθμό δειγματοληψίας 40 δειγμάτων ανά δευτερόλεπτο.

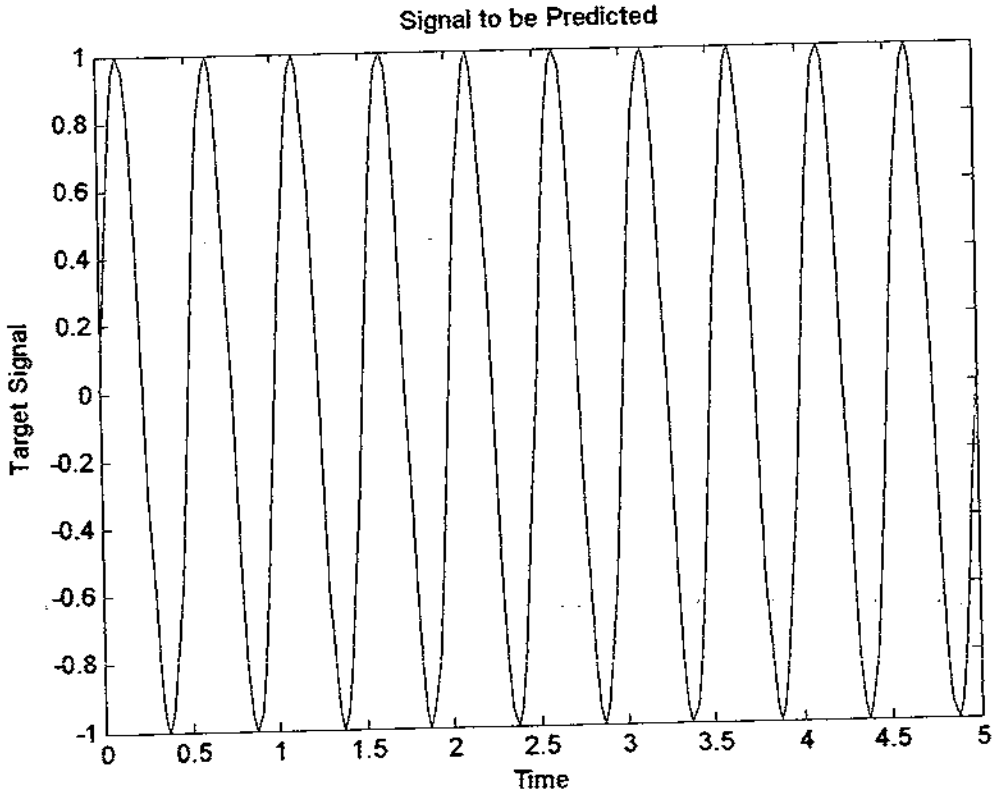
time = 0:0.025:5; (6.1)

t = sin (time*4*pi); (6.2)

Σε κάθε δεδομένο χρονικό βήμα, στο δίκτυο δίνονται οι πέντε προηγούμενες τιμές του **t,** και αναμένεται από το δίκτυο να προβλέψει την επόμενη τιμή. Οι είσοδοι **P** βρίσκονται με την καθυστέρηση του σήματος t από το 1ο στο 5ο χρονικό βήμα με την βοήθεια της συνάρτησης **delaysig**.

P = delaysig (t,1,5); (6.3)

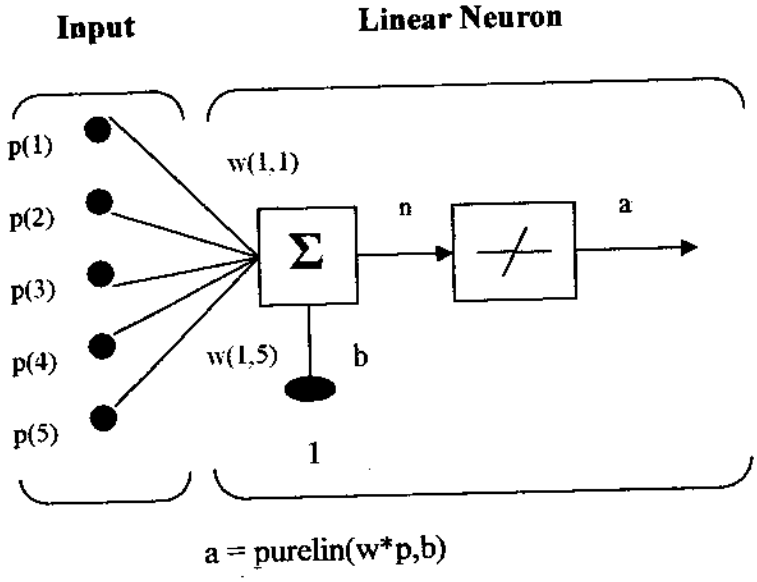
Στο σχήμα που ακολουθεί υπάρχει μια γραφική παράσταση του σήματος t που θα πρέπει να προβλέπεται από το δίκτυο.



Σχήμα 38: Γραφική παράσταση του σήματος t που θα πρέπει να προβλέπεται από το δίκτυο

Επειδή το σήμα είναι χρονικά αμετάβλητο (η σχέση μεταξύ των παλαιών και των μελλοντικών τιμών του σήματος δεν αλλάζει), το δίκτυο μπορεί να σχεδιαστεί απ' ευθείας χρησιμοποιώντας την **solvelin**.

Το πρόβλημα όπως έχει οριστεί ανωτέρω, έχει 5 εισόδους, 5 καθυστερημένες τιμές του σήματος και μία έξοδο, την επόμενη (προβλεφθείσα) τιμή σήματος. Συνεπώς η λύση του δικτύου πρέπει να αποτελείται από ένα μοναδικό νευρώνα με πέντε εισόδους.



***Σχήμα 39:** Δίκτυο με ένα νευρώνα*

Εδώ η **solvelin** βρίσκει τα βάρη και τα biases, για τον ανωτέρω νευρώνα, που ελαχιστοποιεί το άθροισμα τετραγωνικού σφάλματος γι' αυτό το πρόβλημα.

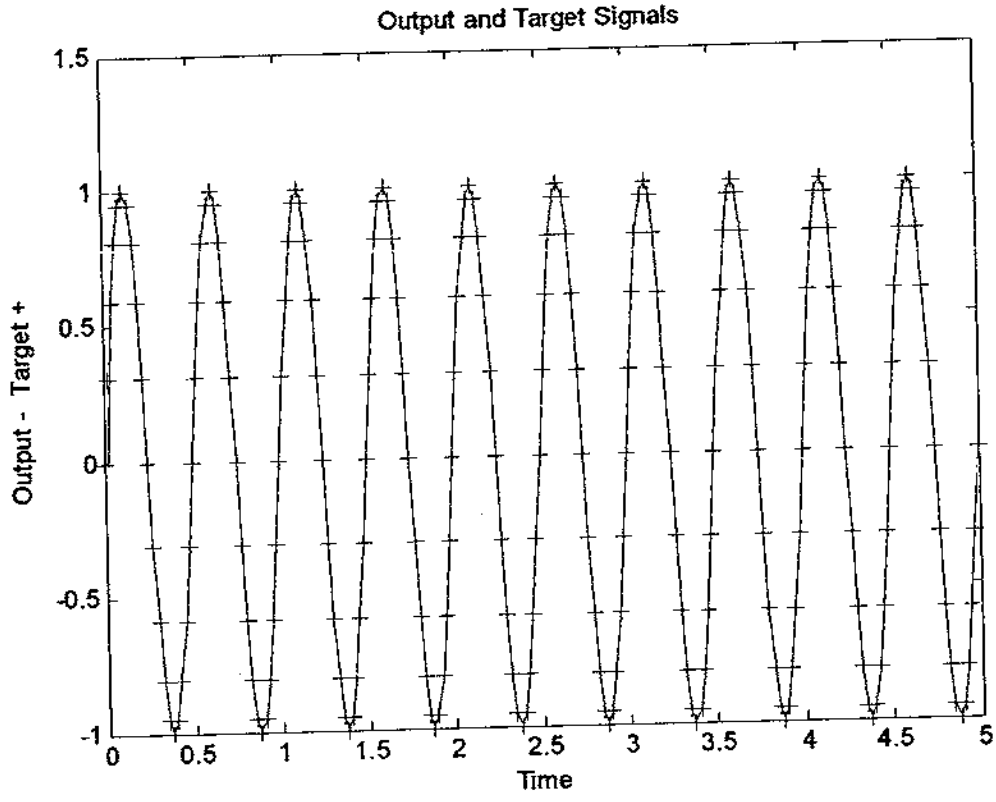
[w,b] = solvelin (P, T) (6.4)

Αυτά τα βάρη και τα biases μπορούν τώρα να ελεγχθούν.

Για να ελέγξουμε το δίκτυο, η έξοδος **a** υπολογίζεται για τα πέντε καθυστερημένα σήματα **P** και συγκρίνονται με το πραγματικό σήμα **t** το οποίο θα έπρεπε να έχει προβλεφθεί.

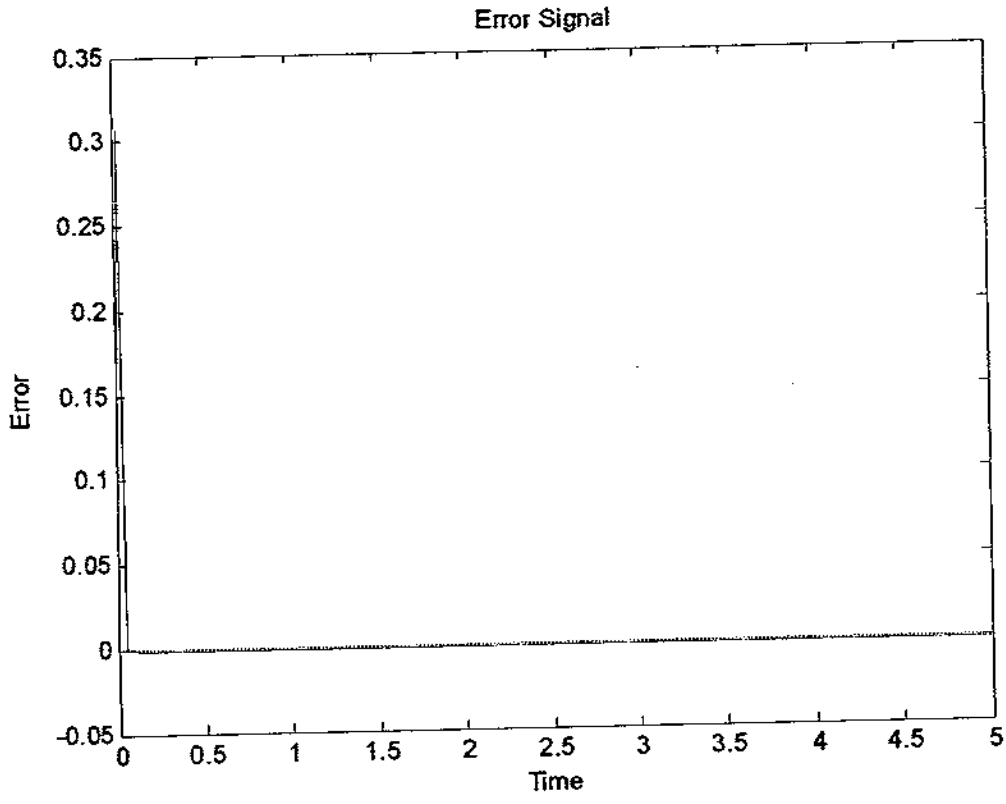
a = simulin(P,w,b); (6.5)

Στη συνέχεια παρατίθεται το διάγραμμα του **a** συναρτήσει του **t**.



Σχήμα 40: *Διάγραμμα του* **a** *σε σύγκριση με το* ***t***

Η πρόβλεψη του δικτύου **a** και το πραγματικό σήμα **t** φαίνεται να ταιριάζουν απόλυτα. Αλλά, για να σιγουρευτούμε, ας δούμε τη γραφική παράσταση της συνάρτησης σφάλματος **e=t-a**.



Σχήμα 41: Σχεδιάγραμμα λάθους

Το δίκτυο, έκανε ένα μικρό λάθος για τα πέντε πρώτα χρονικά βήματα. Αυτό συνέβη επειδή το δίκτυο δεν είχε στην πραγματικότητα 5 καθυστερημένες τιμές σημάτων στη διάθεση του δικτύου μέχρι το 5° χρονικό βήμα. Ωστόσο, μετά το 5° χρονικό βήμα το σφάλμα είναι αμελητέο. Το δίκτυο επιτέλεσε σωστά το έργο του.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Το γραμμικό δίκτυο έκανε μια εξαιρετική δουλειά στην πρόβλεψη ενός γραμμικού κυματισμού, τι θα συνέβαινε όμως αν το σήμα ήταν μη γραμμικό; Προκύπτει όμως ότι ένας γραμμικός προβλεπτής μπορεί να είναι χρήσιμος ακόμη και για μη γραμμικά προβλήματα. Αν και η **solvelin** δεν μπορεί να βρει λύση μηδενικού σφάλματος για μη γραμμικά προβλήματα, ωστόσο μπορεί να ελαχιστοποιεί το άθροισμα του μέσου τετραγωνικού σφάλματος. Σε πολλές περιπτώσεις η λύση αυτή, μπορεί να μοντελοποιήσει καλά ένα μη γραμμικό σήμα και να αντιμετωπίσει συγκεκριμένες εφαρμογές. Δίνοντας στο γραμμικό δίκτυο πολλές τιμές καθυστερημένων δειγμάτων του σήματος , δίνει πολλές πληροφορίες με τις οποίες βρίσκει τη γραμμική εφαρμογή του ελάχιστου σφάλματος για ένα μη γραμμικό πρόβλημα.

6.2 AppLin 2: Προσαρμοστική πρόβλεψη

Σε αυτήν την εφαρμογή, ένα γραμμικό δίκτυο εκπαιδεύεται προσαρμοστικά με το **adaptwh** για να προβλέψει μια χρονοσειρά. Επειδή το γραμμικό δίκτυο εκπαιδεύεται προσαρμοστικά, μπορεί να ανταποκριθεί σε αλλαγές στις σχέσης μεταξύ των προηγουμένων και των μελλοντικών τιμών του σήματος.

Το σήμα **t** που πρέπει να προβλεφθεί διαρκεί έξι δευτερόλεπτα με ένα ρυθμό δειγματοληψίας 20 δειγμάτων ανά δευτερόλεπτο. Ωστόσο, μετά από τέσσερα δευτερόλεπτα η συχνότητα του σήματος ξαφνικά διπλασιάζεται.

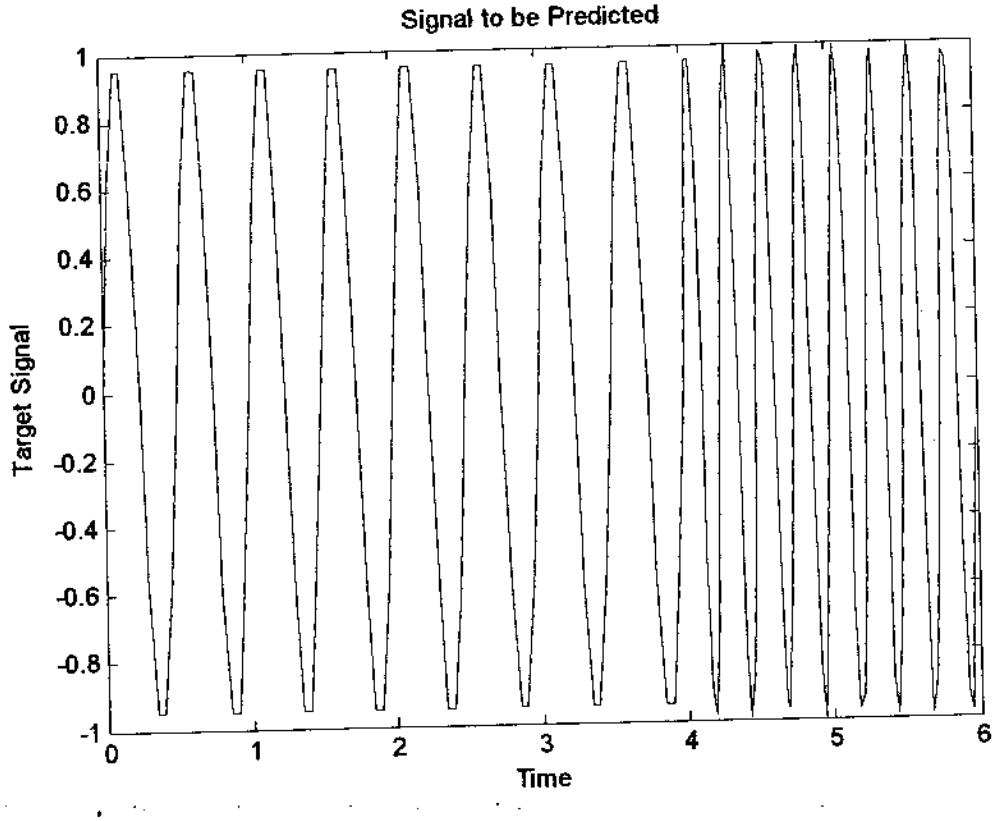
time1 = 0:0.05:4; (6.6)

time2 = 4.05:0.024:6; (6.7)

time = [time1 time2]; (6.8)

t = [sin(time1*4*pi) sin(time2*8*pi)]; (6.9)

Στο σχήμα που ακολουθεί φαίνεται η μορφή του σήματος t:

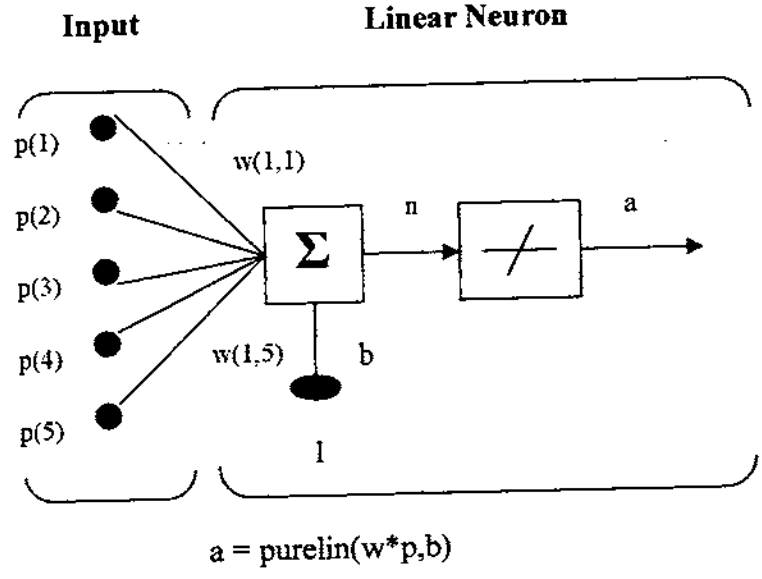


Σχήμα 42: Γραφική παράσταση του σήματος

Σε κάθε χρονική στιγμή, οι τελευταίες 5 τιμές του σήματος **t** δίνονται στο δίκτυο σαν είσοδοι, και προσπαθεί να δώσει ως έξοδο την επόμενη τιμή του **t**. Η συνάρτηση **delaysig** χρησιμοποιείται για να αποκτήσει διαμόρφωση των καθυστερημένων δειγμάτων του σήματος **P** και επιστρέφουν το όρισμα **P** που δίνεται με τη σειρά του ως όρισμα εισόδου στο δίκτυο.

P = delaysig (t,1,5); (6.10)

Το δίκτυο έχει μόνο ένα νευρώνα, καθώς μόνο μια τιμή εξόδου του σήματος t γεννιέται σε κάθε χρονικό βήμα. Αυτός ο νευρώνας έχει πέντε εισόδους, τις 5 καθυστερημένες τιμές του σήματος t.

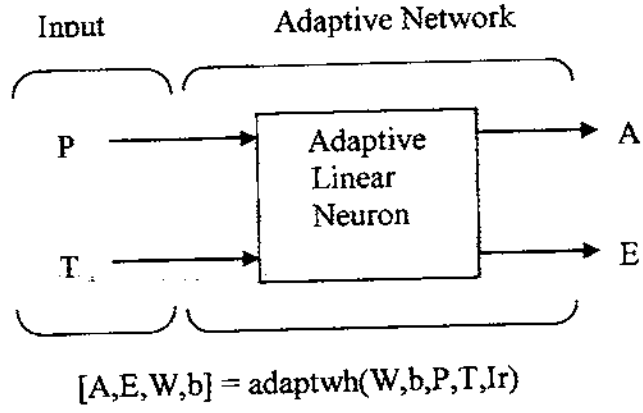


Σχήμα 43: Δίκτυο με ένα νευρώνα

Η συνάρτηση **initlin** δημιουργεί τα αρχικά βάρη και τα biases για το νευρώνα που φαίνονται στο Σχήμα 43

[w,b] = initlin(P,t) (6.11)

Ο ανωτέρω νευρώνας θα εκπαιδευτεί προσαρμοστικά με την χρήση της συνάρτησης **adaptwh** όπως φαίνεται στο παρακάτω σχήμα.

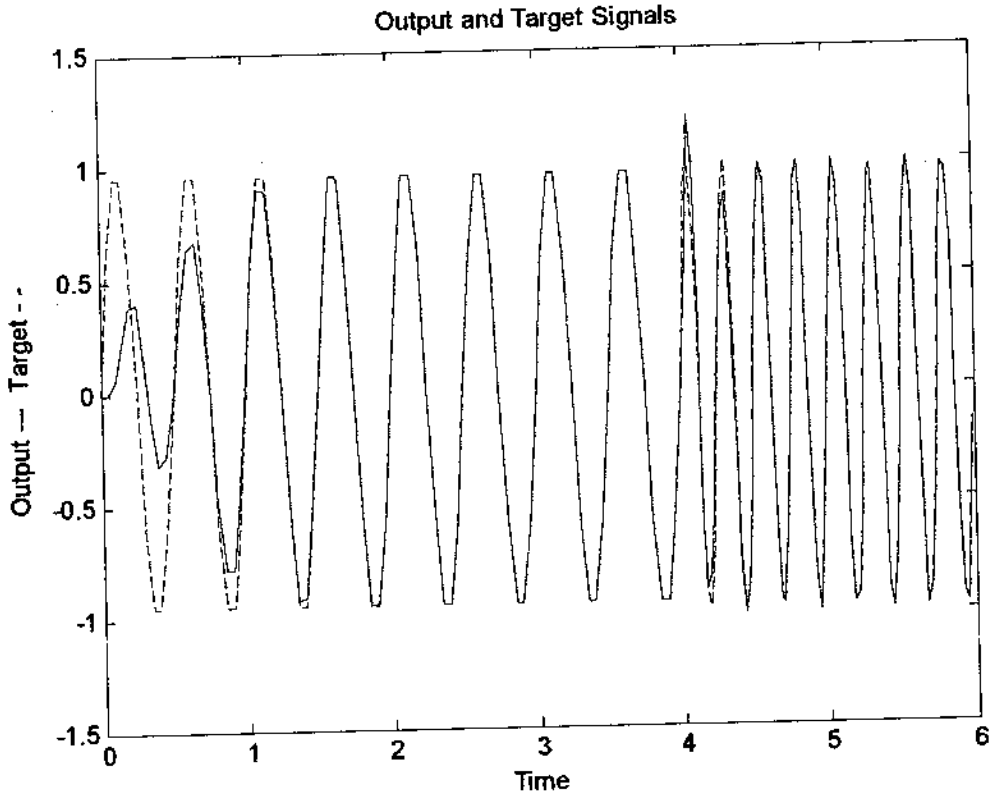


Σχήμα 44:Προσαρμοστική εκπαίδευση του νευρώνα

Εδώ φαίνεται η προσαρμοστική εκπαίδευση του νευρώνα με είσοδο/ στόχο σημάτων **P** και **t** χρησιμοποιώντας ρυθμό μάθησης 0.1. γίνεται με την ακόλουθη εντολή

[a,e,w,b] = adaptwh (w, b, P, t, 0.1); (6.12)

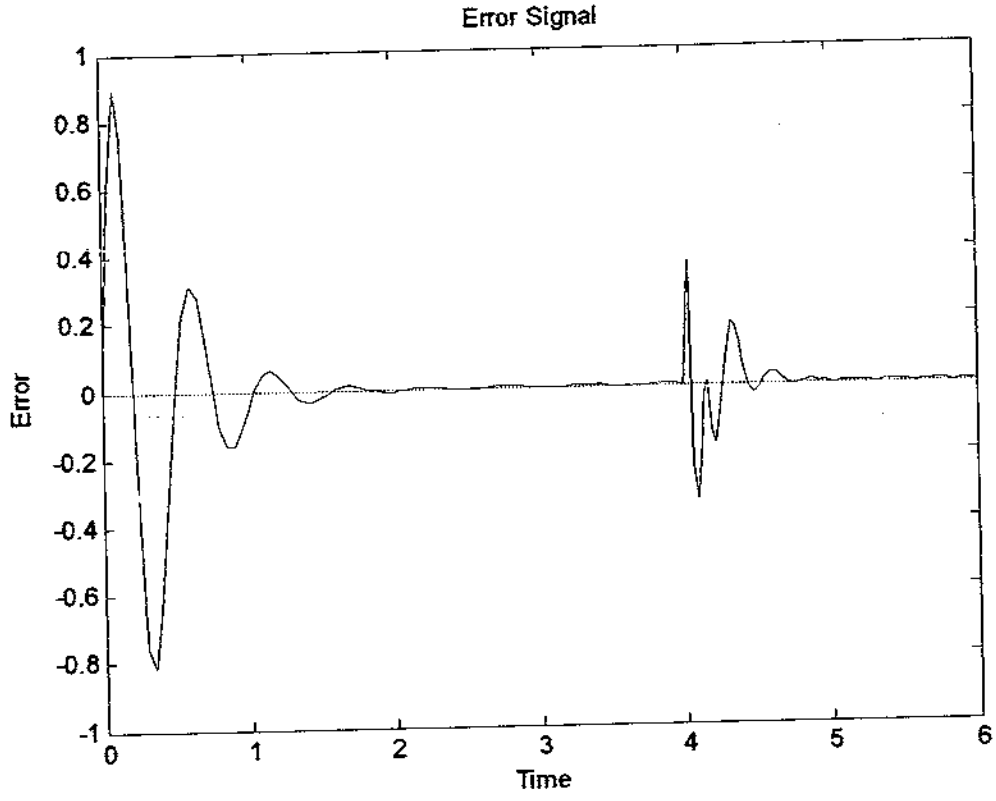
Μόλις το δίκτυο έχει προσαρμοστεί, μπορούμε να σχεδιάσουμε το σήμα εξόδου και να το συγκρίνουμε με το σήμα στόχου.



Σχήμα 45: Διάγραμμα σύγκρισης του σήματος εξόδου και του σήματος στόχου

Αρχικά, παίρνει για το δίκτυο ενάμιση δευτερόλεπτο (30 δείγματα) για να ανιχνεύσει το σήμα στόχου. Έπειτα οι προβλέψεις είναι ακριβείς μέχρι το τέταρτο δευτερόλεπτο όταν το σήμα στόχου ξαφνικά αλλάζει συχνότητα. Ωστόσο, το προσαρμοστικό δίκτυο μαθαίνει να ανιχνεύει το νέο σήμα σε ένα ακόμη μικρότερο διάστημα καθώς έχει ήδη μάθει τη συμπεριφορά ενός παρόμοιου σήματος.

Ένα διάγραμμα του σφάλματος κάνει να γίνουν αυτές οι επιδράσεις πιο ορατές.



Σχήμα 46: *Διάγραμμα του σήματος σφάλματος*

6.3 AppLin 3: Ταυτοποίηση γραμμικού συστήματος

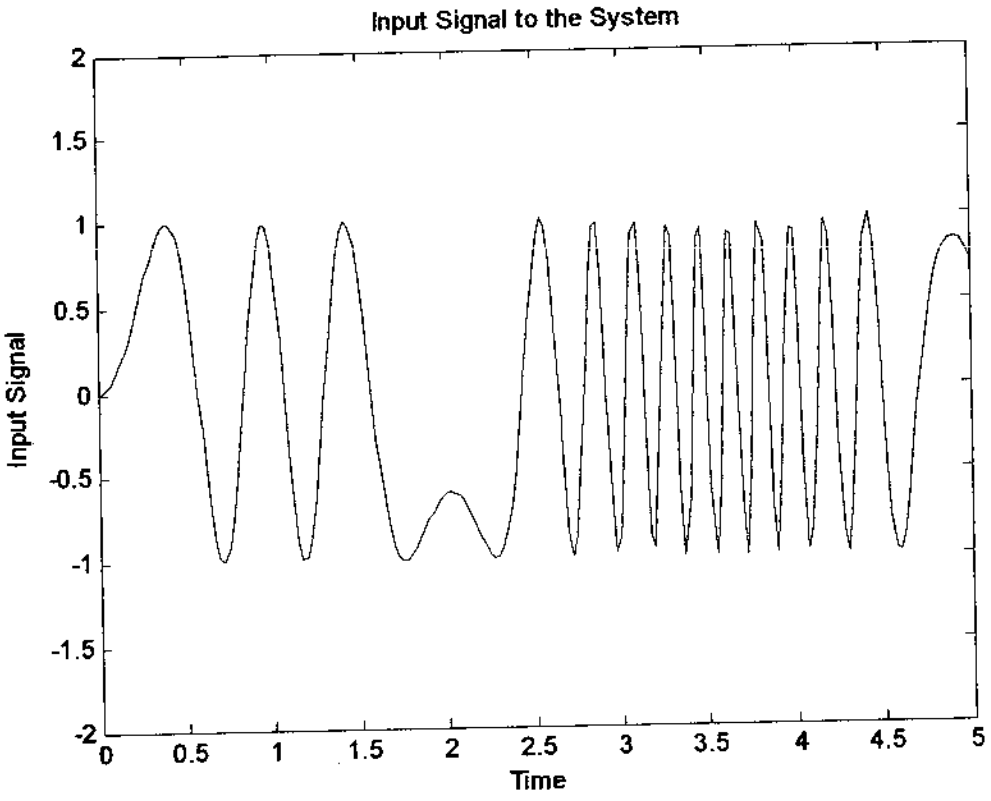
Τα γραμμικά συστήματα μπορεί να χρησιμοποιούνται για να μοντελοποιήσουν πραγματικά συστήματα. Αν το πραγματικό σύστημα είναι γραμμικό ή σχεδόν γραμμικό τότε το γραμμικό δίκτυο μπορεί να λειτουργήσει σαν μοντέλο πολύ χαμηλού σφάλματος.

Στο παράδειγμα που ακολουθεί ένα σήμα εισόδου x που ορίζεται ως:

time=0:0.025:5; (6.13)

x=sin(sin(time).*time*10); (6.14)

Στα Σχήματα 47 και 48 φαίνονται οι γραφικές παραστάσεις των σημάτων εισόδου και εξόδου του συστήματος αντίστοιχα.



Σχήμα 47: Γραφική παράσταση του σήματος εισόδου

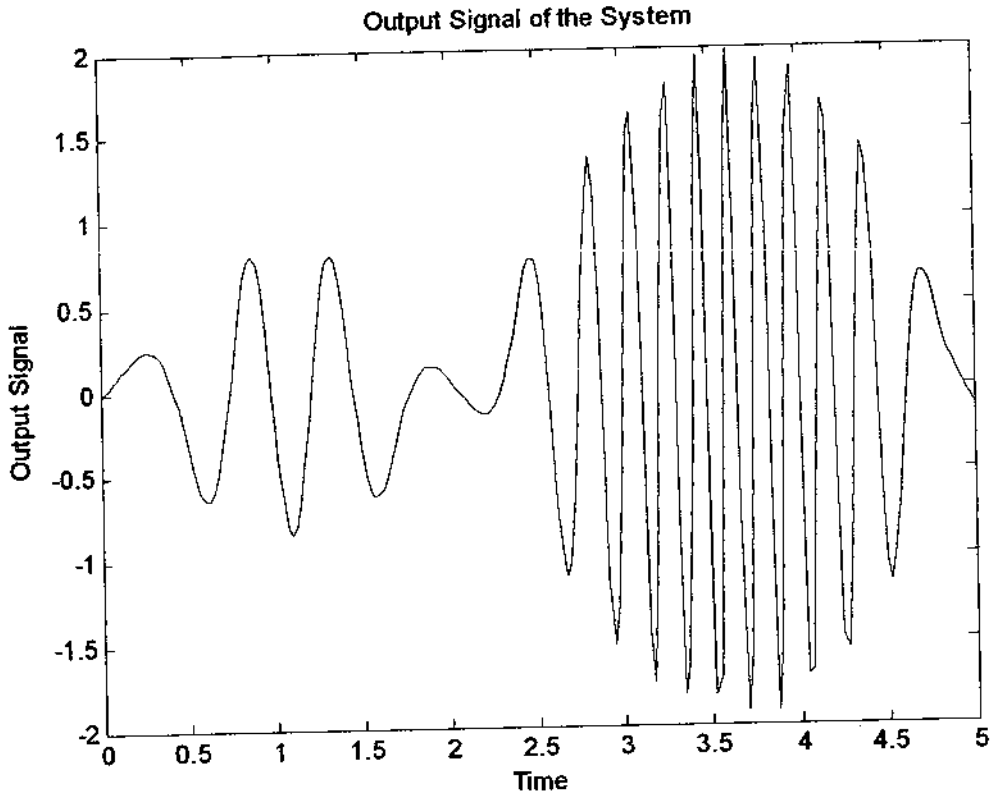
Η έξοδος ενός συστήματος οδηγείται στην είσοδο ενός γραμμικού δικτύου το οποίο περιγράφεται από την εξίσωση διαφορών

$y(n)=x(n)+x(n-1)-1.5x(n-2)$ (6.15)

και υπολογίζεται με την εντολή

t=filter([1 0.5 -1.5],1,x); (6.16)

του MATLAB.



Σχήμα 48: Διάγραμμα του σήματος εξόδου

Το δίκτυο προσπαθεί να προβλέψει την έξοδο t του συστήματος δεδομένων της τρέχουσας και των δύο προηγούμενων τιμών του σήματος εισόδου x. Η συνάρτηση **delaysig** δημιουργεί τις αναγκαίες τιμές του σήματος εισόδου.

P=delaysig(x,0,2); (6.17)

Το δίκτυο που απαιτείται για να λύσει αυτό το πρόβλημα έχει μόνο ένα νευρώνα επειδή το σύστημα έχει μόνο μία έξοδο. Αυτός ο νευρώνας πρέπει να έχει τρεις εισόδους για να λαμβάνει την τρέχουσα και τις δύο καθυστερημένες τιμές του σήματος εισόδου.

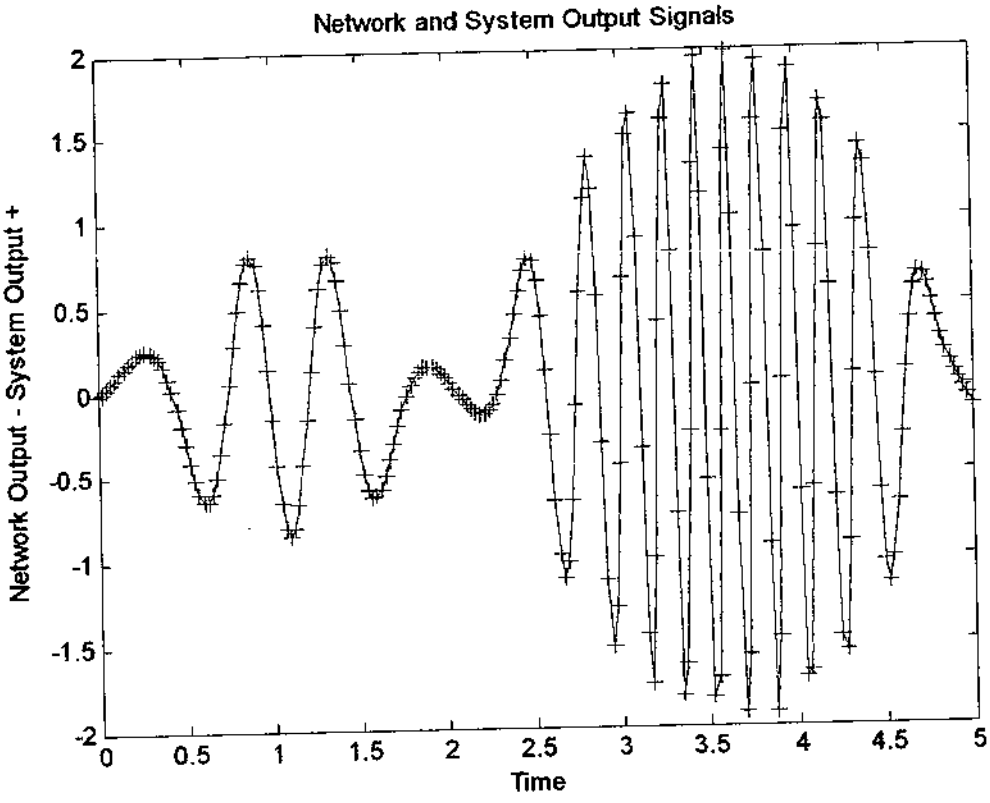
Η συνάρτηση solvelin σχεδιάζει ένα τέτοιο νευρώνα.

[w,b]=solvelin(P,t); (6.18)

Μόλις αποκτηθούν τα βάρη και τα biases, το δίκτυο μπορεί να ελεγχθεί. Εδώ το γραμμικό δίκτυο υπολογίζει κατά προσέγγιση το a της εξόδου t του συστήματος δεδομένων της τρέχουσας και των καθυστερημένων τιμών της εισόδου του συστήματος.

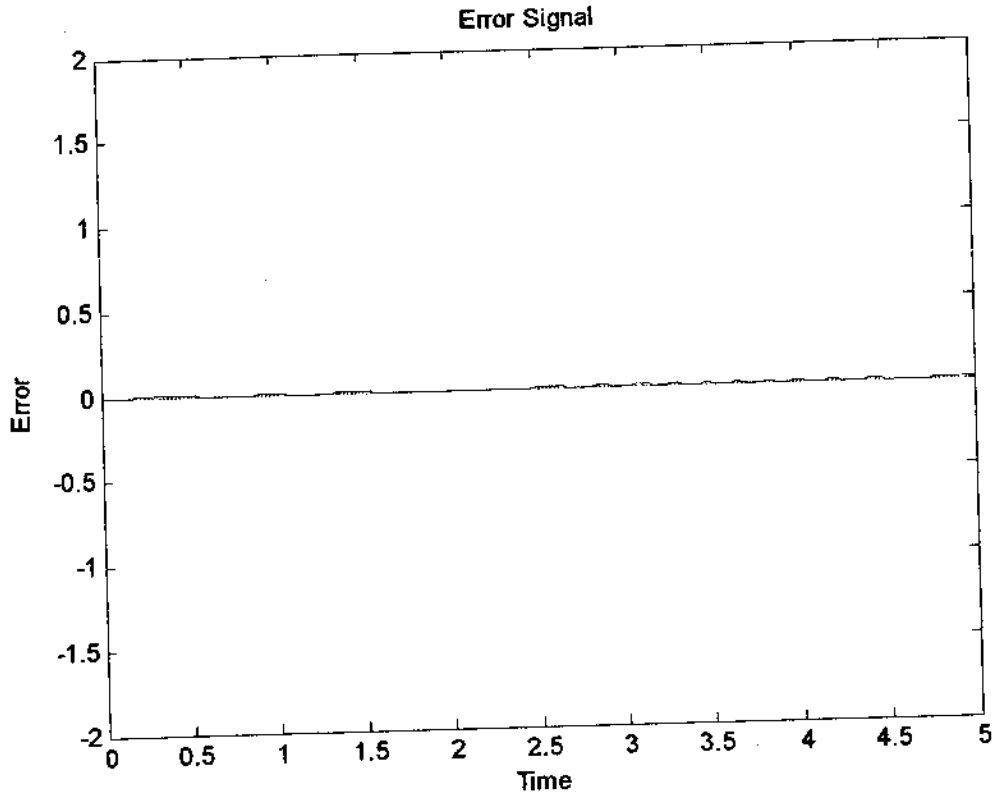
a= simulin (P,w,b); (6.19)

Η έξοδος t του συστήματος και η έξοδος a του δικτύου συγκρίνονται στο παρακάτω διάγραμμα.



Σχήμα 49: *Διάγραμμα σύγκρισης* της εξόδου t του συστήματος και της έξοδος a του δικτύου

Το δίκτυο φαίνεται να κάνει τέλεια δουλειά στη μοντελοποίηση του συστήματος. Παρακάτω φαίνεται ένα διάγραμμα της διαφοράς των εξόδων του δικτύου και του πραγματικού συστήματος.



Σχήμα 50: Διάγραμμα του σήματος λάθους

ΣΥΜΠΕΡΑΣΜΑΤΑ

Όπως συνέβη με την γραμμική πρόβλεψη, τα μοντέλα γραμμικών συστημάτων μπορούν να χρησιμοποιηθούν για να μοντελοποιήσουν γραμμικά συστήματα με μηδενικό σφάλμα, και μη γραμμικά συστήματα με ελάχιστο άθροισμα μέσου τετραγωνικού σφάλματος. Δίνοντας στο γραμμικό δίκτυο πολλές καθυστερημένες τιμές από τα σήματα εισόδου του συστήματος η συνάρτηση solvelin θα μας επιτρέψει να ελαχιστοποιήσουμε το σφάλμα που σχετίζεται με την μοντελοποίηση ενός μη γραμμικού συστήματος.

6.4 AppLin 4: Ταυτοποίηση προσαρμοστικού συστήματος

Στο παράδειγμα αυτό θα παρουσιάσουμε πως μπορούμε να εκπαιδεύσουμε ένα γραμμικό δίκτυο για να μοντελοποιήσει προσαρμοστικά ένα γραμμικό σύστημα. Η προσαρμοστική εκπαίδευση του δικτύου, μπορεί να αλλάξει τη συμπεριφορά του καθώς το σύστημα αλλάζει. Στο σχήμα που ακολουθεί ένα σήμα εισόδου **x** που ορίζεται ως εξής:

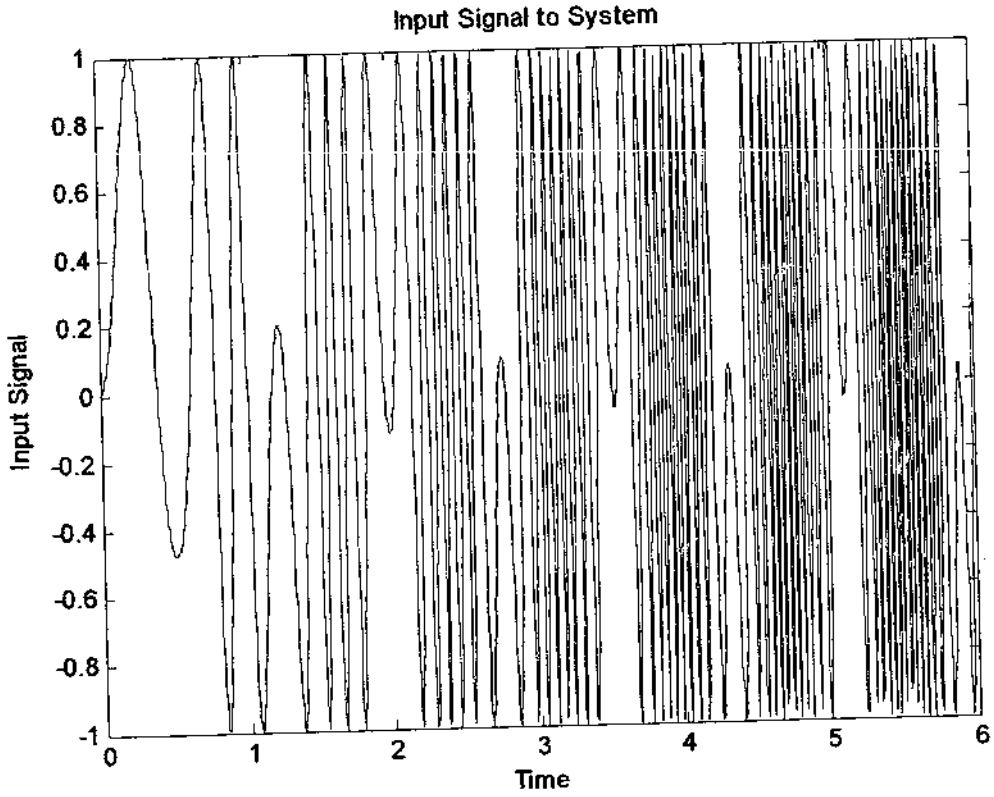
time1 = 0:0.005:4; (6.20)

time2 = 4.005:0.005:6; (6.21)

time = [time1 time2]; (6.22)

x = sin (sin(time*4).*time*8); (6.23)

Στα διαγράμματα 51 και 52 φαίνονται οι γραφικές παραστάσεις των σημάτων εισόδου και εξόδου του συστήματος αντίστοιχα.



Σχήμα 51: Γραφική παράσταση του σήματος εισόδου

Εδώ είναι το σήμα εξόδου t του συστήματος. Σημειώστε ότι το σύστημα ενεργεί διαφορετικά στα διαστήματα πριν και μετά το τέταρτο δευτερόλεπτο.

steps1 = length(time1); (6.24)

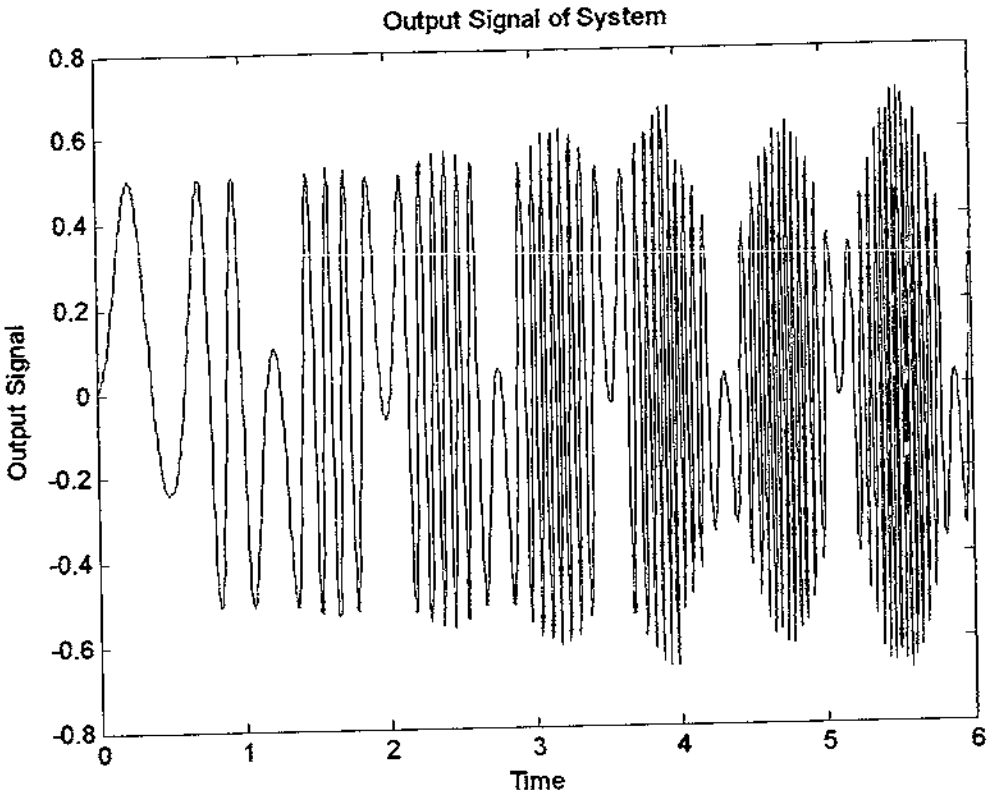
[t1,state] = filter ([1-0.5],1,x(1:steps1)); (6.25)

steps2 = length(time2); (6.26)

t2 = filter ([0.9-0.6],1,x((1:steps2) + steps1), state); (6.27)

t = [t1 t2]; (6.28)

Ακολουθεί το διάγραμμα των εξόδων του συστήματος. Σημειωτέον ότι εξαιτίας της πολυπλοκότητας του συστήματος, δεν είναι φανερό ότι το σύστημα άλλαξε στα 4 δευτερόλεπτα.



Σχήμα 52: *Διάγραμμα του σήματος εξόδου*

Η είσοδος στο δίκτυο **P** είναι η τρέχουσα και οι προηγούμενες τιμές των σημάτων εισόδου **x** του συστήματος. Το δίκτυο πρέπει να εκτιμήσει την έξοδο **t** του συστήματος από αυτές τις δύο τιμές:

P = delaysig (x,0,1); (6.29)

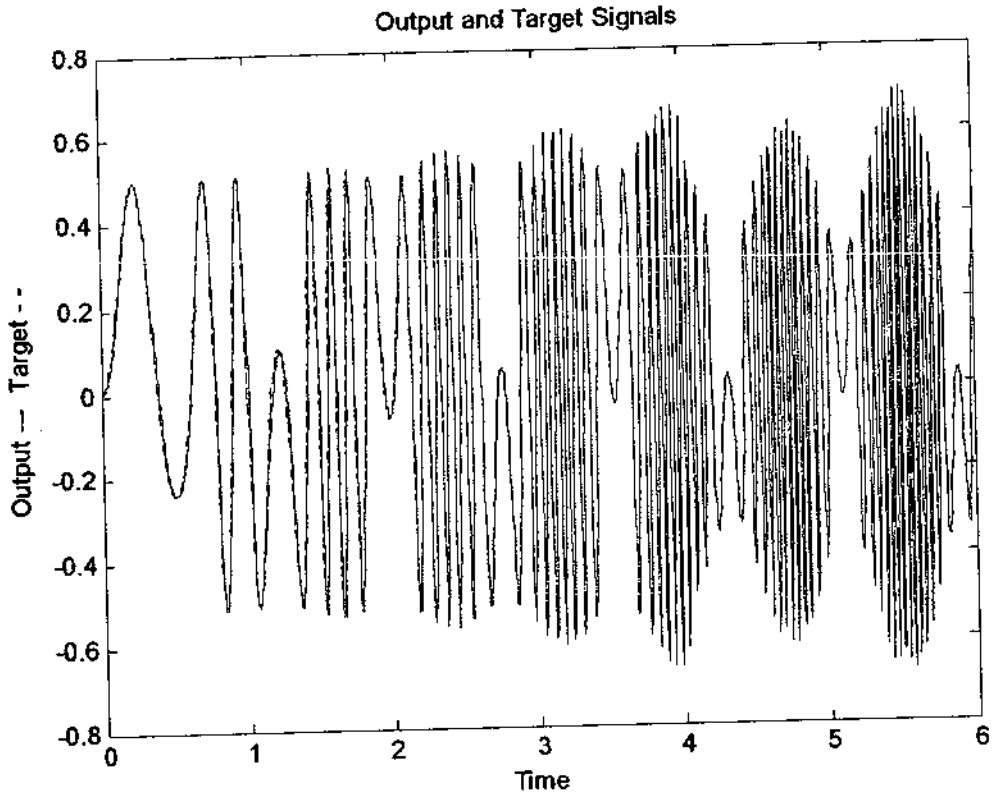
Το δίκτυο καθορίζει τις αρχικές τιμές με την συνάρτηση **initlin**, η οποία δημιουργεί τα βάρη και τα biases για την διπλή γραμμική είσοδο του νευρώνα

[w,b] = initlin(P,t); (6.30)

Αυτά τα βάρη και τα biases μπορούν τώρα να εκπαιδευτούν προσαρμοστικά στο σήμα με την χρήση της συνάρτησης **adaptwh** χρησιμοποιώντας το ρυθμό μάθησης 0.5.

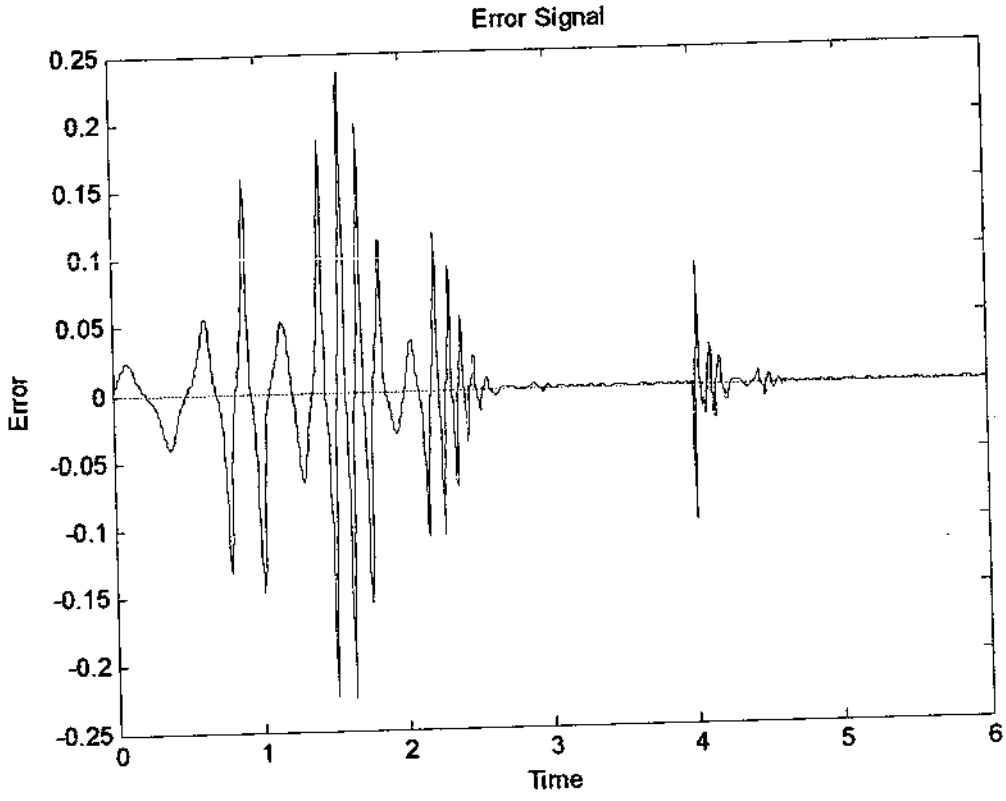
[a,e,w,b] = adaptwh(w,b,P,t,0.5) (6.31)

Για να δούμε πόσο καλά τα πήγε το δίκτυο, μπορούμε να σχεδιάσουμε την εκτίμηση **a** του δικτύου της εξόδου του συστήματος έναντι της πραγματικής εξόδου **t**. Όπως προκύπτει τα σήματα είναι αρκετά πολύπλοκα και αυτό το διάγραμμα δεν βοηθάει αρκετά.



*Σχήμα 53: Διάγραμμα σύγκρισης της εκτίμησης **a** του δικτύου έναντι της πραγματικής εξόδου **t**.*

Αντί αυτού μπορούμε να δούμε το σφάλμα ανάμεσα στην έξοδο του δικτύου και την έξοδο του συστήματος.



Σχήμα 54: Διάγραμμα του σήματος λάθους

Το διάγραμμα σφάλματος δείχνει ότι το δίκτυο χρειάστηκε 2.5 δευτερόλεπτα να εντοπίσει το μοντέλο με πολύ μεγάλη ακρίβεια. Μετά, στο 4° δευτερόλεπτο, όταν το σύστημα αλλάζει απότομα, απαιτούνται άλλα 0.2 δευτερόλεπτα για το δίκτυο για να το μοντελοποιήσει ξανά σωστά.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Ένα προσαρμοστικό γραμμικό μοντέλο ενός συστήματος μπορεί να χρησιμοποιηθεί για να αποκτήσει ένα μεγάλο μέρος πληροφοριών. Για παράδειγμα, το προσαρμοστικό μοντέλο μπορεί να αναλυθεί σε κάθε δεδομένο χρόνο για να καθορίσει χαρακτηριστικά του πραγματικού συστήματος. Για παράδειγμα, το προσαρμοστικό δίκτυο μπορεί να παρακολουθήσει και να δώσει μια προειδοποίηση στις σταθερές του όπου πλησιάζουν οι τιμές σε αστάθεια.

Μια άλλη ικανότητα που έχει το προσαρμοστικό γραμμικό μοντέλο είναι ότι μπορεί να βρίσκει ένα ελάχιστο άθροισμα τετραγωνικού σφάλματος για ένα μη γραμμικό σύστημα. Ένα προσαρμοστικό γραμμικό μοντέλο θα είναι πολύ ακριβές όσο το μη γραμμικό σύστημα παραμένει κοντά σε ένα λειτουργικό σημείο. Εάν το μη γραμμικό σύστημα κινείται σε ένα διαφορετικό λειτουργικό σημείο, το προσαρμοστικό γραμμικό δίκτυο θα αλλάξει για να το μοντελοποιήσει σε ένα νέο σημείο.

7.Εφαρμογές βασισμένες σε πραγματικά δεδομένα

7.1 Μηνιαίες πωλήσεις αυτοκινήτων

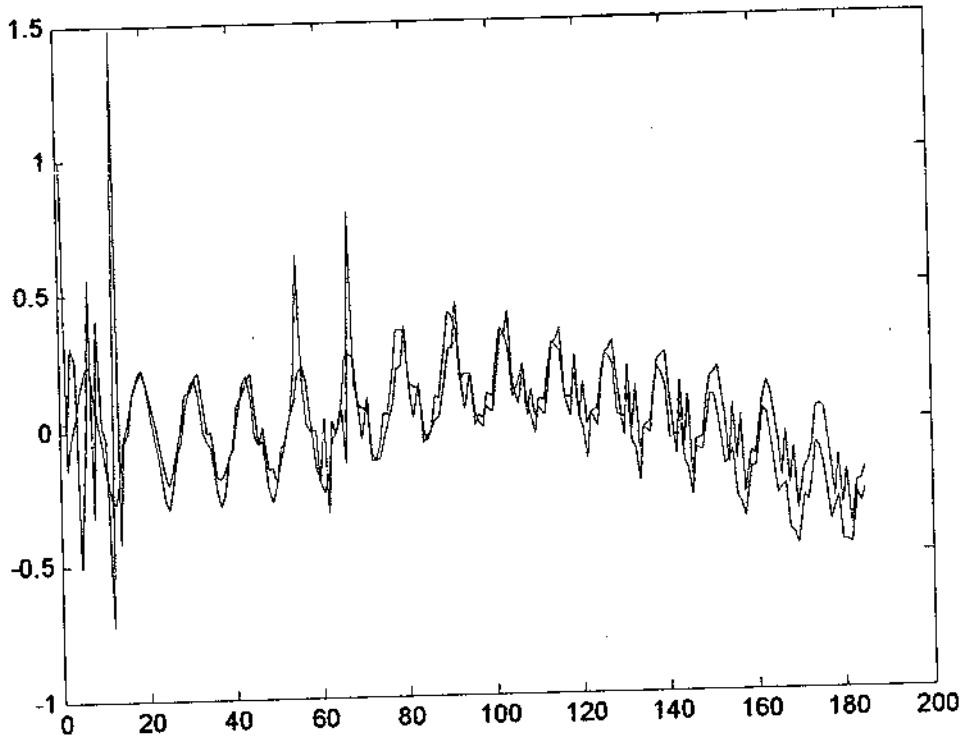
7.1.1 Γραμμικό μοντέλο

Το αρχείο input περιέχει τις Μηνιαίες Πωλήσεις μίας βιομηχανίας Αυτοκινήτων. Οι τιμές που περιέχονται στο αρχείο είναι 185. Θα χρησιμοποιήσουμε ένα γραμμικό νευρωνικό δίκτυο για να μπορέσουμε να προβλέψουμε, με βάση προηγούμενες τιμές, την επόμενη τιμή της χρονοσειράς που έχουμε στη διάθεση μας. Τα δεδομένα του αρχείου θα τα φορτώσουμε στο περιβάλλον του Matlab χρησιμοποιώντας την εντολή load.

Εκτελώντας την εντολή load input φορτώνονται στο Matlab τα περιεχόμενα του αρχείου input.dat και οι τιμές του τοποθετούνται σ' ένα πίνακα διάνυσμα με όνομα input.

Με την εντολή prediction(input,nin,train) μπορούμε να κάνουμε την πρόβλεψη της επόμενης τιμής της χρονοσειράς. Όπου input είναι η προς πρόβλεψη χρονοσειρά input,στο nin βάζουμε το πλήθος των προηγούμενων δειγμάτων που χρησιμοποιούνται για την πρόβλεψη και train ένα διάνυσμα στο οποίο τοποθετούμε το τμήμα της χρονοσειράς που θέλουμε να χρησιμοποιήσουμε για την εκπαίδευση του δικτύου αν train=[m,n], τότε για την εκπαίδευση χρησιμοποιείται το τμήμα της ακολουθίας ΄ input ΄ για παράδειγμα από το m-οστό ως το n-οστό δείγμα. Παρακάτω ακολουθούν κάποια πειράματα που έγιναν χρησιμοποιώντας διαφορετικούς συνδυασμούς των nin και train:

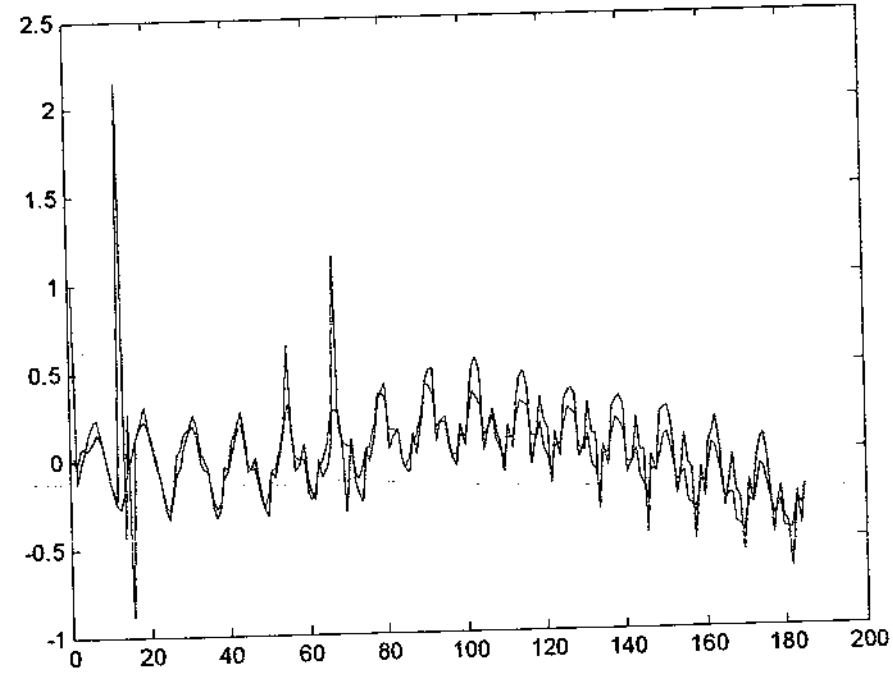
1) prediction(input,15,[31 39 46 53 64 73 86 95 103 117 120 123 126 135])



Σχήμα 55: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της έξοδος του δικτύου

MSE =7.8006

2) prediction(input,15,[2:12 24:36 45:62])



Σχήμα 56: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της έξοδος του δικτύου

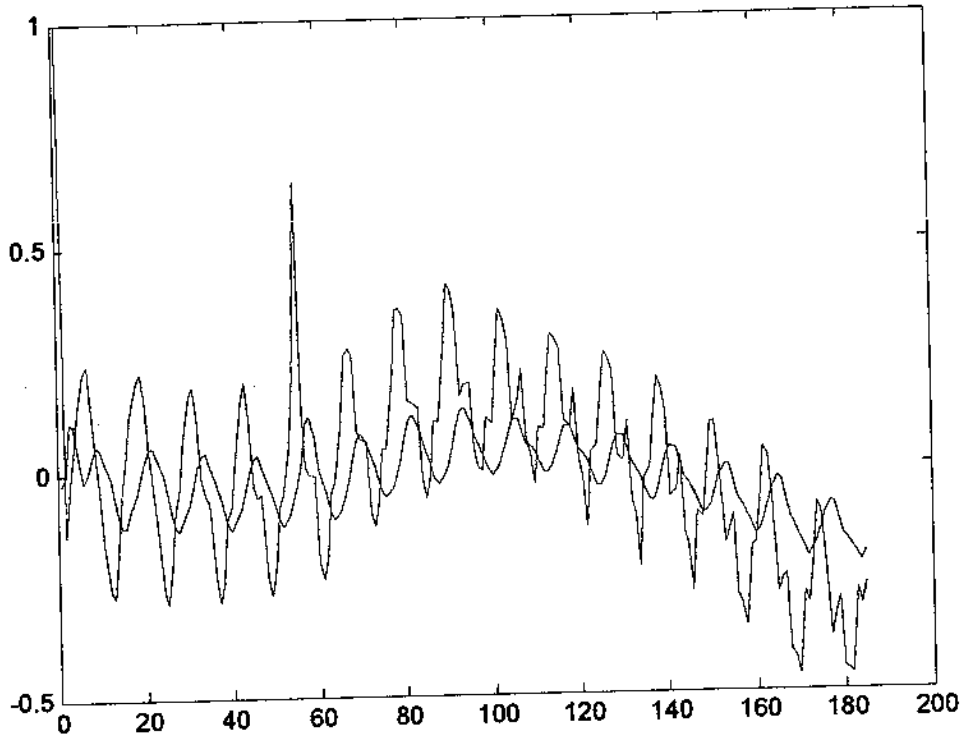
MSE =10.6334

1. Με αφορμή τα παραπάνω παραδείγματα καταλήγουμε στο συμπέρασμα ότι χρησιμοποιώντας για την εκπαίδευση τιμές από όλες τι περιόδους η πρόβλεψη είναι καλύτερη από ότι αν χρησιμοποιούμε τιμές από τις 2 πρώτες περιόδους. Αυτό σημαίνει ότι όσο αντιπροσωπευτικότερο είναι το δείγμα εκπαίδευσης, τόσο καλύτερη προσέγγιση πετυχαίνει. Αυτό γίνεται φανερό και από τις τιμές του μέσου τετραγωνικού σφάλματος.
2. Όταν κρατάμε σταθερή την εκπαίδευση (train) και μεταβάλλουμε μόνο το πλήθος των προηγούμενων δειγμάτων, παρατηρούμε ότι έχουμε μια αύξηση του MSE καθώς αυξάνεται το nin. Ενώ μετά από το nin=8 το MSE αρχίζει να μειώνεται φτάνοντας στη βέλτιστη τιμή του στο nin=15 που έχουμε MSE=6.0186. στη συνέχεια όμως αυξάνοντας το nin έχουμε πάλι αύξηση του MSE.

7.1.2 Προσαρμοστικό μοντέλο

Στην προσπάθεια να έχουμε την καλύτερη δυνατή πρόβλεψη επόμενης τιμής με το ελάχιστο μέσο τετραγωνικό σφάλμα χρησιμοποιήσαμε και το προσαρμοστικό νευρωνικό μοντέλο.
Για το σκοπό αυτό χρησιμοποιούμε την συνάρτηση prediction_ad(input,nin,train) με την μπορούμε να κάνουμε την πρόβλεψη της επόμενης τιμής της χρονοσειρας.
Αξιοσημείωτο είναι ότι στο προσαρμοστικό μοντέλο μπορούμε να μεταβάλλουμε εκτός από τις τιμές των ορισμάτων nin και train το ρυθμό μάθησης(learning rate) όπως και το πλήθος των επαναλήψεων(passes) που κάνει το προσαρμοστικό δίκτυο στη χρονοσειρά για να την μάθει. Επειδή η προσαρμοστική πρόβλεψη ξεκινά από τυχαίες τιμές στα βάρη κάθε φορά ακολουθούμε την στοχαστική πρόβλεψη, δηλαδή χρησιμοποιούμε τη μέση τιμή των μέσων τετραγωνικών σφαλμάτων. Έτσι εξαλείφεται η τυχαιότητα των bias.
Παρακάτω ακολουθούν κάποιες εφαρμογές που έγιναν χρησιμοποιώντας διαφορετικούς συνδυασμούς των nin , train, learning rate και passes που η πρόβλεψη τους είναι αρκετά καλή και το σφάλμα τους μικρό:

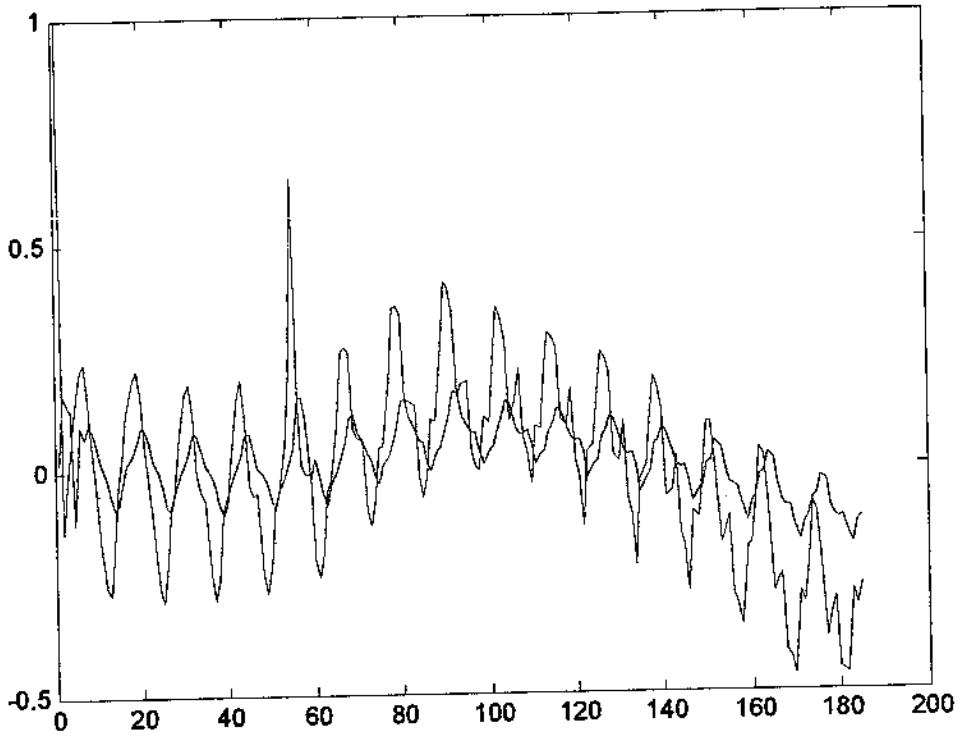
1) Lr:0.037
   Passes:16
   Train: [15 26 31 42 56 68 75 90 112 135 141 146 160 168 175]
   Nin=5



Σχήμα 57: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της έξοδος του δικτύου

MSE =20.4131

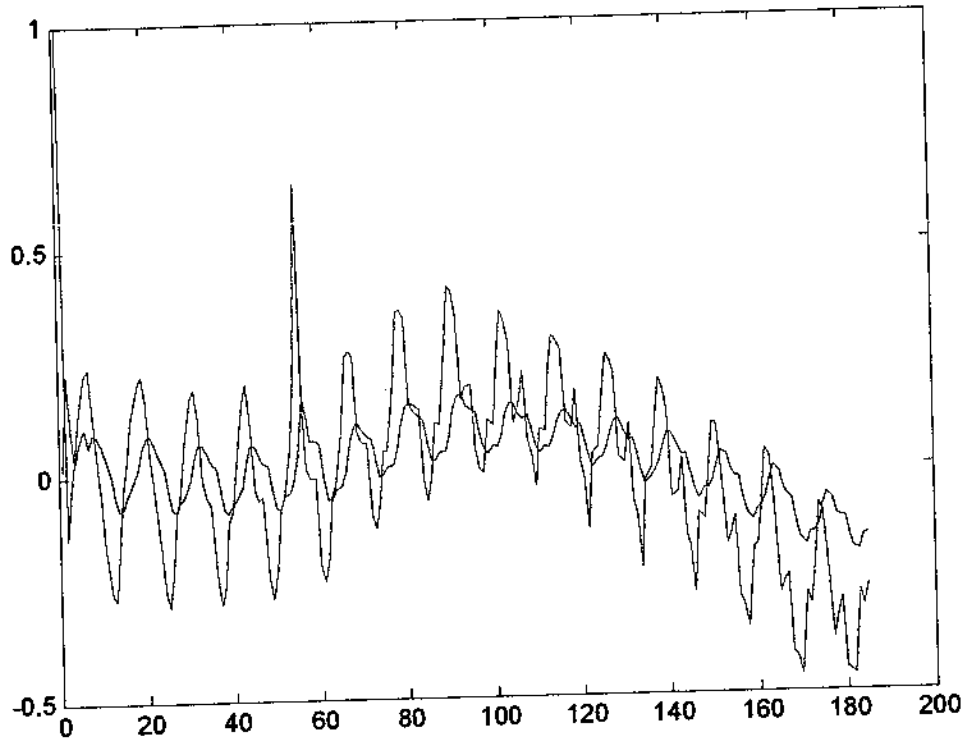
2) Lr:0.037
Passes=10
Train=[15 26 31 42 56 68 75 90 112 135 141 146 160 168 175]
Nin=5



Σχήμα 58: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της έξοδος του δικτύου

MSE =13.1433

3) Lr:0.025
Passes=10
Train=[15 26 31 42 56 68 75 90 112 135 141 146 160 168 175]
Nin=5



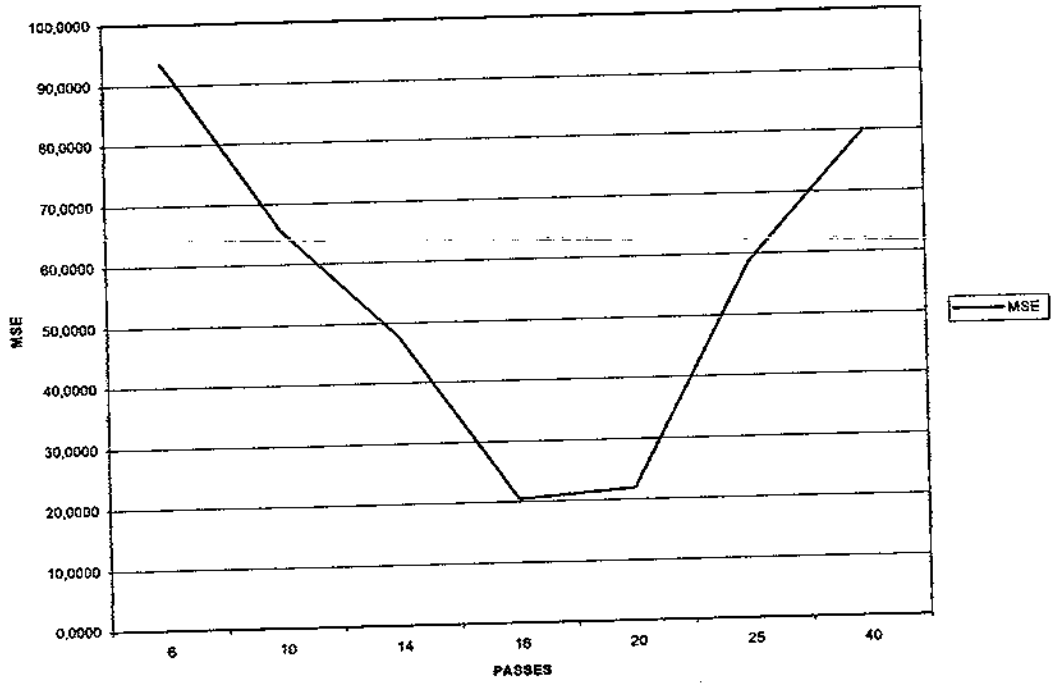
Σχήμα 59: *Διάγραμμα σύγκρισης* της εξόδου πρόβλεψης του συστήματος και της έξοδος του δικτύου

MSE =18.0025

Μετά από αρκετά πειράματα εξάγουμε τα εξής συμπεράσματα:

1. Οι προηγούμενες τιμές που χρησιμοποιήσαμε για πρόβλεψη είναι nin=5, για την εκπαίδευση χρησιμοποιήσαμε τιμές από όλες τις περιόδους train:[15 26 31 42 56 68 75 90 112 135 141 146 160 168 175] και το learning rate που έδωσε τα καλύτερα αποτελέσματα ήταν learning rate:0.035. Κρατώντας όλες τις προηγούμενες μεταβλητές σταθερές και μεταβάλλοντας μόνο τα passes είχαμε τα παρακάτω αποτελέσματα:

PASSES	MSE
6	93,5384
10	65,6324
14	47,5866
16	20,4131
20	21,9049
25	58,6198
40	80,1171

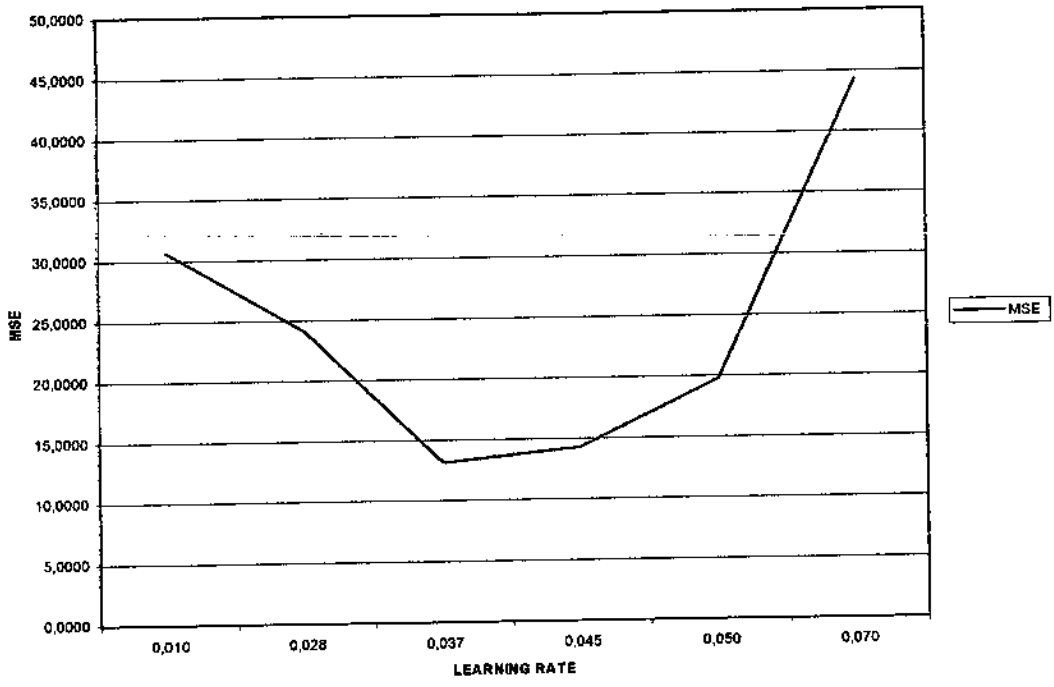


Σχήμα 60: *Διάγραμμα πορείας του MSE για διάφορες τιμές των επαναλήψεων*

Όπως απεικονίζεται και στο παραπάνω διάγραμμα όσο αυξάνονται τα passes το MSE μειώνεται, δηλ. η πρόβλεψη κάνει μια καλύτερη προσέγγιση για την επόμενη τιμή μέχρι ενός σημείου. Στη προκειμένη περίπτωση το βέλτιστο MSE είναι το 20,4131 με passes=16. Από εκεί και πέρα το μέσο τετραγωνικό σφάλμα αυξάνει όσο αυξάνονται τα passes. Αυτό σημαίνει ότι όταν έχουμε αρκετά μεγάλο αριθμό επαναλήψεων η πρόβλεψη του προσαρμοστικού μοντέλου δεν μπορεί να προσεγγίσει την μελλοντική επόμενη τιμή.

2. Ανάλογη πορεία με αυτή του μέσου τετραγωνικού σφάλματος μεταβάλλοντας τα passes έχουμε και με την αύξηση του learning rate, κρατώντας όλες τις υπόλοιπες μεταβλητές για κάθε περίπτωση σταθερές και συγκεκριμένα nin=5, passes:16 ενώ για την εκπαίδευση χρησιμοποιήσαμε τις τιμές train:[15 26 31 42 56 68 75 90 112 135 141 146 160 168 175]. Τα αποτελέσματα που πήραμε ήταν τα εξής:

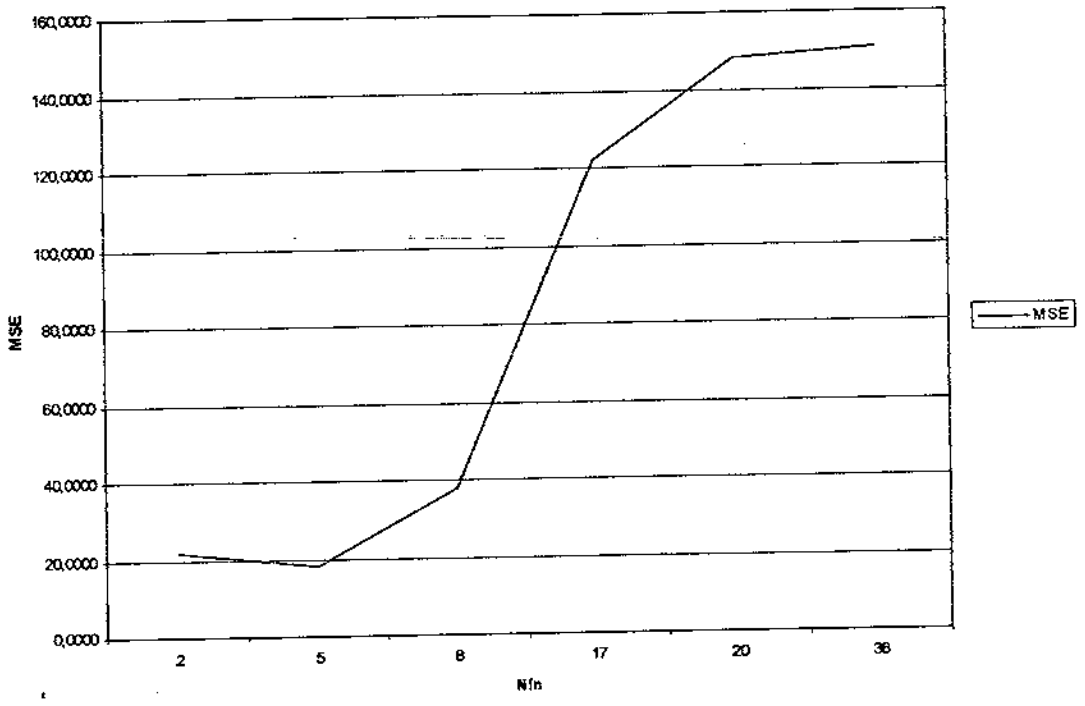
LR	MSE
0,010	30,6574
0,028	24,1047
0,037	13,1433
0,045	14,2935
0,050	19,7675
0,070	44,2563



Σχήμα 61: Διάγραμμα πορείας του MSE για διάφορες τιμές του ρυθμού μάθησης

3. Την ίδια πορεία με αυτή του μέσου τετραγωνικού σφάλματος μεταβάλλοντας τα passes ή και το learning rate έχουμε και με την αύξηση του πλήθους των προηγούμενων τιμών του δείγματος nin, κρατώντας όλες τις υπόλοιπες μεταβλητές σταθερές, με learning rate:0.025, passes=10 και για την εκπαίδευση χρησιμοποιήσαμε τις τιμές train:[15 26 31 42 56 68 75 90 112 135 141 146 160 168 175].

Nin	MSE
2	21,9363
5	18,0025
8	37,7811
17	122,2542
20	148,0553
36	150,6688



Σχήμα 62: Διάγραμμα πορείας του MSE για διάφορες τιμές του πλήθους των προηγούμενων δειγμάτων

4. Όσον αφορά την εκπαίδευση που είναι ένας πολύ σημαντικός παράγοντας της πρόβλεψης παρατηρούμε ότι: όταν παίρνουμε τιμές από όλες τις περιόδους το MSE είναι το βέλτιστο από αυτό που θα πάρουμε αν χρησιμοποιήσουμε τιμές που αφορούν μόνο τις πρώτες ή μόνο τις τελευταίες περιόδους. Επιπροσθέτως μια ολόκληρη περίοδος μας δίνει καλύτερη πρόβλεψη από τη μισή περίοδο. Οι παρατηρήσεις μας αυτές απεικονίζονται και στον παρακάτω πίνακα.

TRAIN	MSE
[12:32 42:64 73:96 124:145 154:178]	25.5670
[10:26 42:64 70:86]	34.2476
[80:95 120:142 164:180]	32.2476
[124:142]	68.4130
[20:80]	123.8961

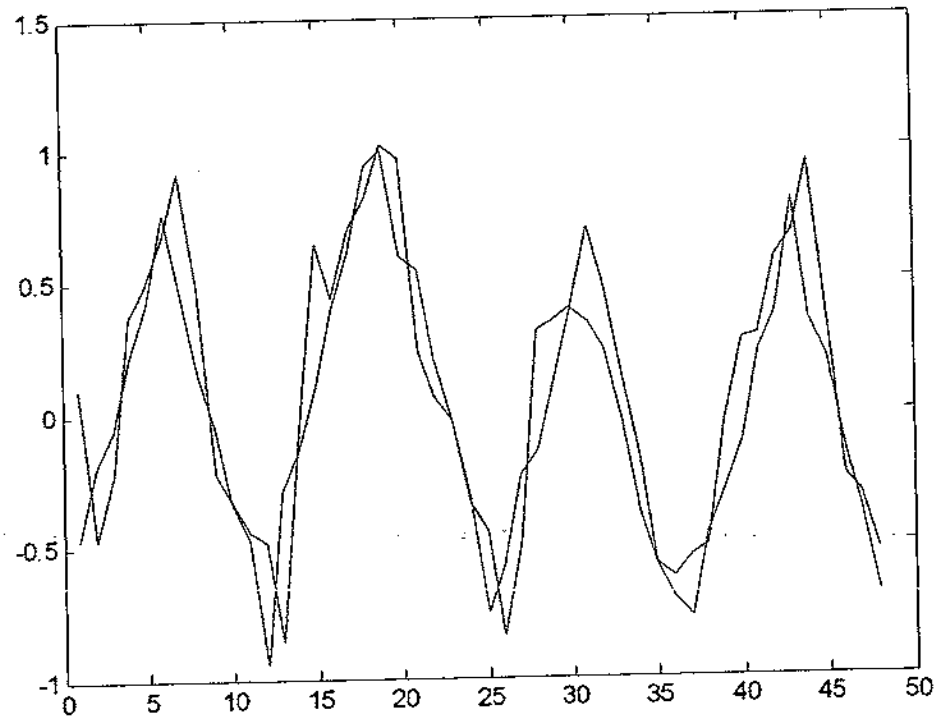
7.2 Μηνιαίες απαιτήσεις για αναψυκτικά με χαμηλές θερμίδες

7.2.1 Γραμμικό μοντέλο

Το αρχείο coca περιέχει τις Μηνιαίες Απαιτήσεις μιας επιχείρησης σε Αναψυκτικά με Χαμηλές θερμίδες . Οι τιμές που περιέχονται στο αρχείο είναι 48. Θα χρησιμοποιήσουμε ένα γραμμικό νευρωνικό δίκτυο για να μπορέσουμε να προβλέψουμε, με βάση προηγούμενες τιμές, την επόμενη τιμή της χρονοσειράς που έχουμε στη διάθεση μας. Τα δεδομένα του αρχείου θα τα φορτώσουμε στο περιβάλλον του Matlab χρησιμοποιώντας την εντολή load. Εκτελώντας την εντολή load coca φορτώνονται στο Matlab τα περιεχόμενα του αρχείου coca.dat και οι τιμές του τοποθετούνται σ' ένα πίνακα διάνυσμα με όνομα z.

Με την εντολή prediction(z,nin,train) μπορούμε να κάνουμε την πρόβλεψη της επόμενης τιμής της χρονοσειράς. Όπου z είναι η προς πρόβλεψη χρονοσειρά coca,στο nin βάζουμε το πλήθος των προηγούμενων δειγμάτων που χρησιμοποιούνται για την πρόβλεψη και train ένα διάνυσμα στο οποίο τοποθετούμε το τμήμα της χρονοσειράς που θέλουμε να χρησιμοποιήσουμε για την εκπαίδευση του δικτύου αν train=[m,n], τότε για την εκπαίδευση χρησιμοποιείται το τμήμα της ακολουθίας ΄ z΄ για παράδειγμα από το m-οστό ως το n-οστό δείγμα. Παρακάτω ακολουθούν κάποια πειράματα που έγιναν χρησιμοποιώντας διαφορετικούς συνδυασμούς των nin και train:

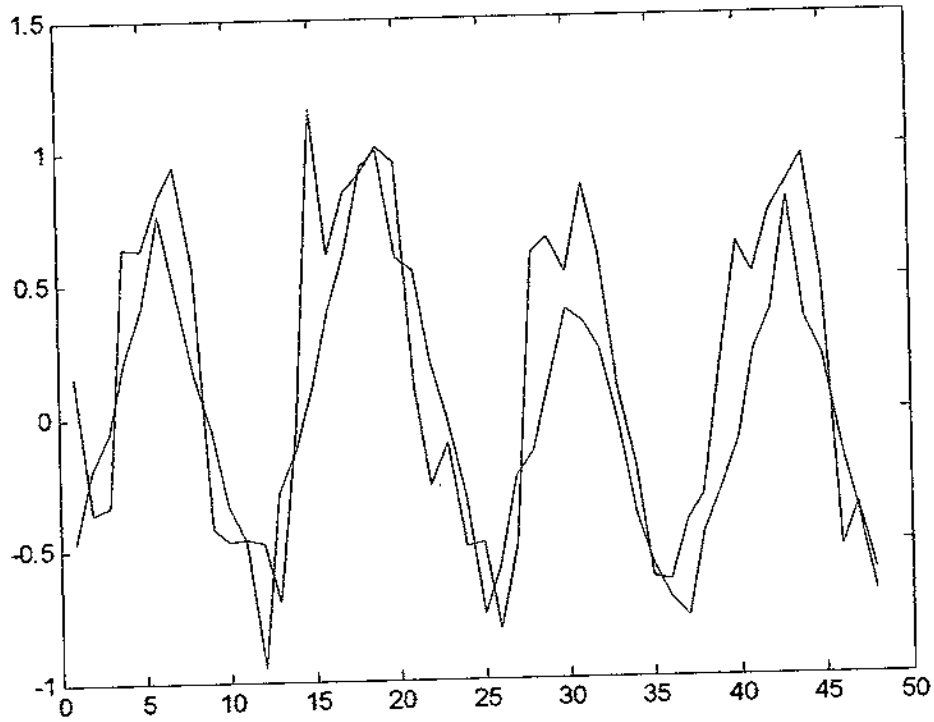
1) prediction(z,3,[2 6 12 18 23 27 34 38 42 46])



Σχήμα 63: *Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου*

MSE =3.4015

2) prediction(z,3,[2 6 12 18 23 27])



Σχήμα 64: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου

MSE =6.7150

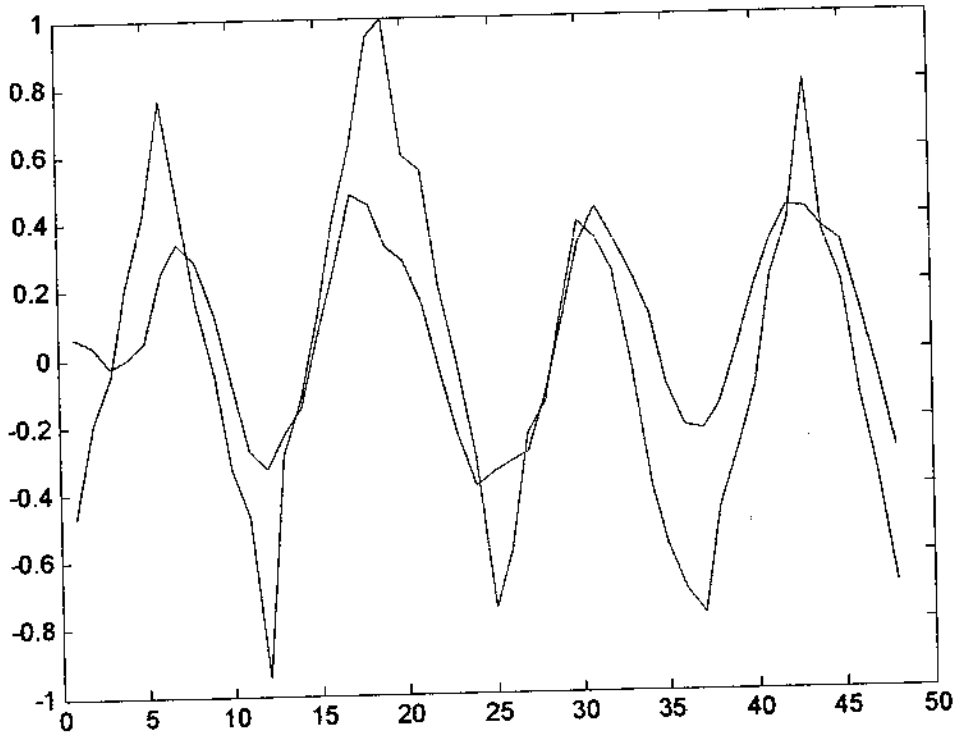
1. Με αφορμή τα παραπάνω παραδείγματα καταλήγουμε στο συμπέρασμα ότι χρησιμοποιώντας για την εκπαίδευση τιμές από όλες τι περιόδους η πρόβλεψη είναι καλύτερη από ότι αν χρησιμοποιούμε τιμές από τις 2 πρώτες περιόδους. Αυτό σημαίνει ότι όσο αντιπροσωπευτικότερο είναι το δείγμα εκπαίδευσης, τόσο καλύτερη προσέγγιση πετυχαίνει. Αυτό γίνεται φανερό και από τις τιμές του μέσου τετραγωνικού σφάλματος.
2. Το nin αυτής της χρονοσειράς έχει την ίδια συμπεριφορά με το nin της χρονοσειράς των μηνιαίων πωλήσεων αυτοκινήτων.

7.2.2 Προσαρμοστικό μοντέλο

Στην προσπάθεια να έχουμε την καλύτερη δυνατή πρόβλεψη επόμενης τιμής με το ελάχιστο μέσο τετραγωνικό σφάλμα χρησιμοποιήσαμε και το προσαρμοστικό νευρωνικό μοντέλο.
Για το σκοπό αυτό χρησιμοποιούμε την συνάρτηση prediction_ad(z,nin,train) με την μπορούμε να κάνουμε την πρόβλεψη της επόμενης τιμής της χρονοσειρας.

Αξιοσημείωτο είναι ότι στο προσαρμοστικό μοντέλο μπορούμε να μεταβάλλουμε εκτός από τις τιμές των ορισμάτων nin και train το ρυθμό μάθησης(learning rate) όπως και το πλήθος των επαναλήψεων(passes) που κάνει το προσαρμοστικό δίκτυο στη χρονοσειρά για να την μάθει. Επειδή η προσαρμοστική πρόβλεψη ξεκινά από τυχαίες τιμές στα βάρη κάθε φορά ακολουθούμε την στοχαστική πρόβλεψη, δηλαδή χρησιμοποιούμε τη μέση τιμή των μέσων τετραγωνικών σφαλμάτων. Έτσι εξαλείφεται η τυχαιότητα των bias.
Παρακάτω ακολουθούν κάποιες εφαρμογές που έγιναν χρησιμοποιώντας διαφορετικούς συνδυασμούς των nin , train, learning rate και passes που η πρόβλεψη τους είναι αρκετά καλή και το σφάλμα τους μικρό:

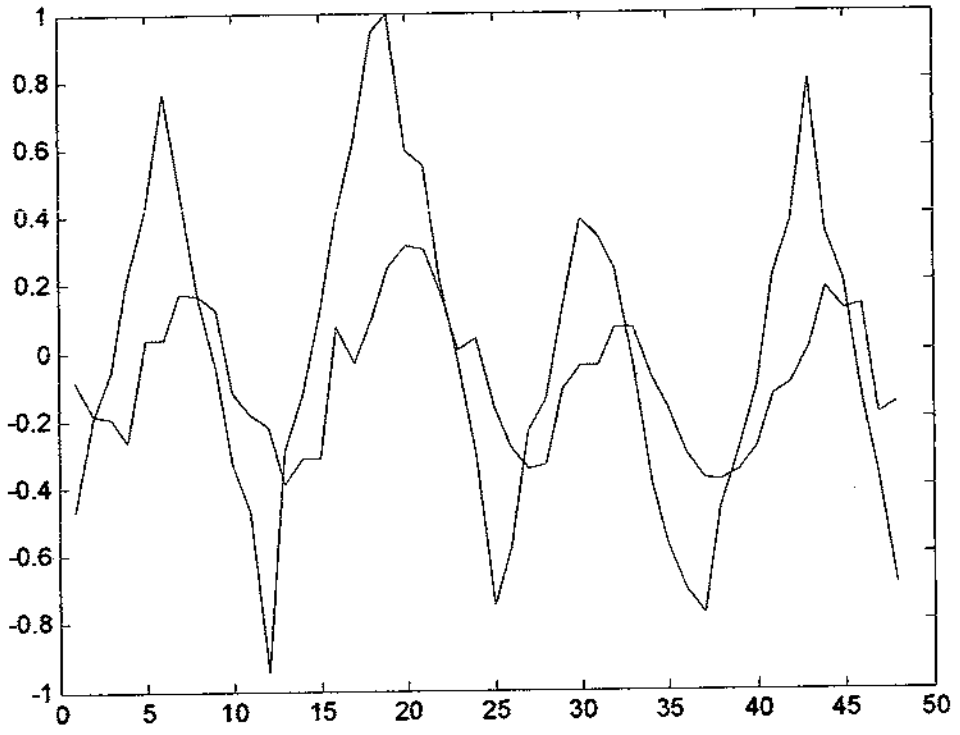
1) Lr :0.012
Passes:2
Train:[2 6 12 18 22 29 30 34 42 46]
Nin=6



Σχήμα 65: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου

MSE =19.5624

2) Lr:0.020
Passes=2
Train=[2 6 12 18 22 29 34 30 42 46]
Nin=6



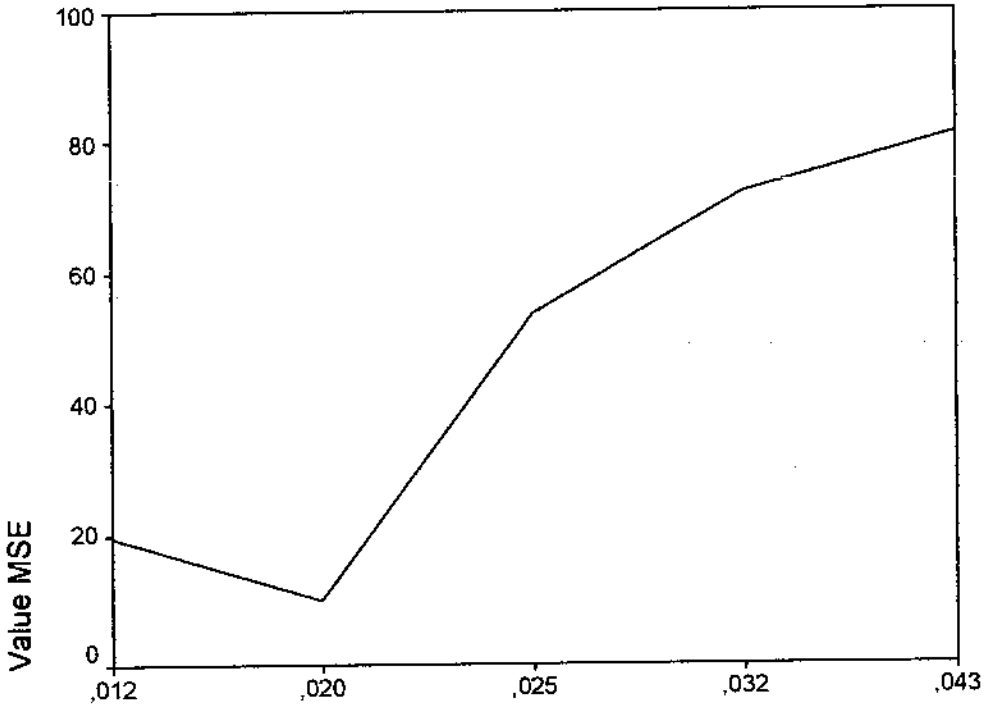
Σχήμα 66: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου

MSE =9.9050

Μετά από αρκετά πειράματα εξάγουμε τα εξής συμπεράσματα:

1. Οι προηγούμενες τιμές που χρησιμοποιήσαμε για πρόβλεψη είναι nin=6, για την εκπαίδευση χρησιμοποιήσαμε τιμές από όλες τις περιόδους train:[2 6 12 18 22 29 30 34 42 46] και οι επαναλήψεις που μας έδιναν τα καλύτερα αποτελέσματα ήταν passes=2. Κρατώντας όλες τις προηγούμενες μεταβλητές σταθερές και μεταβάλλοντας μόνο το learning rate είχαμε τα παρακάτω αποτελέσματα:

LR	MSE
0,012	19,5624
0,020	9,9050
0,025	53,5254
0,032	72,2580
0,043	81,0706



Σχήμα 67:Διάγραμμα πορείας του MSE για διάφορες τιμές του ρυθμού μάθησης

Όπως απεικονίζεται και στο παραπάνω διάγραμμα όσο αυξάνεται το Learning rate το MSE μειώνεται, δηλ. η πρόβλεψη κάνει μια καλύτερη προσέγγιση για την επόμενη τιμή μέχρι ενός σημείου. Στη προκειμένη περίπτωση το βέλτιστο MSE είναι το 9,9050 με learning rate:0.020. Από εκεί και πέρα το μέσο τετραγωνικό σφάλμα αυξάνει όσο αυξάνει το learning rate.Αυτό σημαίνει ότι το πολύ μεγάλο learning rate κάνει τη μάθηση του νευρώνα ασταθή. Άλλωστε σύμφωνα με τη θεωρία το learning rate πρέπει να κυμαίνεται μεταξύ των ορίων 0.01 και 0.05 για να έχουμε τα βέλτιστα αποτελέσματα.

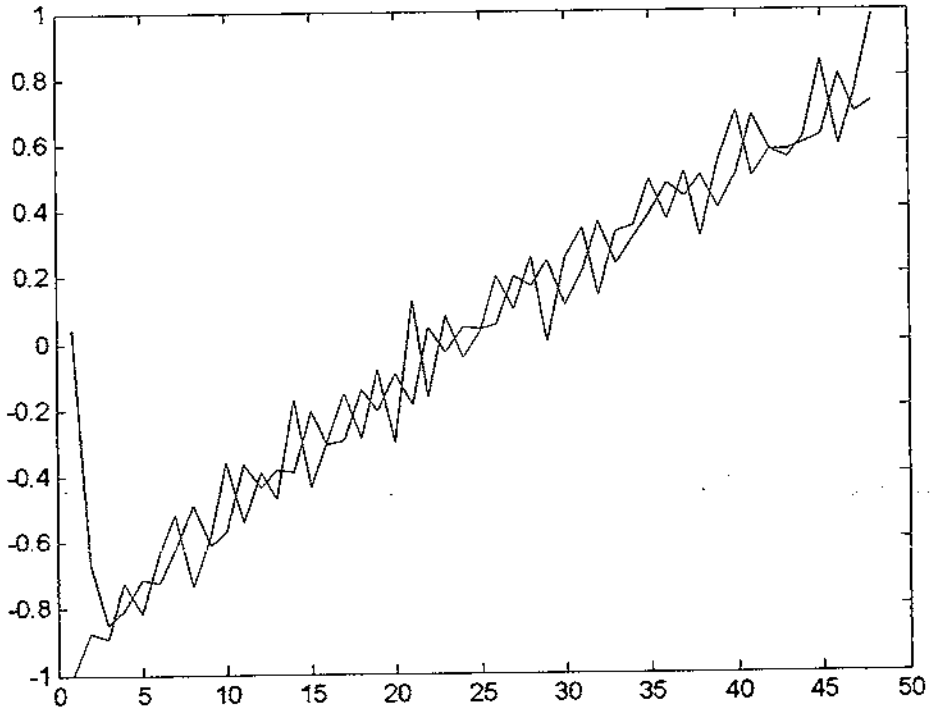
2. Ανάλογη πορεία με αυτή του μέσου τετραγωνικού σφάλματος μεταβάλλοντας το learning rate έχουμε και με την αύξηση των passes ή το πλήθος των προηγούμενων τιμών του δείγματος nin, κρατώντας όλες τις υπόλοιπες μεταβλητές για κάθε περίπτωση σταθερές.
3. Όσον αφορά την εκπαίδευση που είναι ένας πολύ σημαντικός παράγοντας της πρόβλεψης παρατηρούμε ότι: όταν παίρνουμε τιμές από όλες τις περιόδους το MSE είναι το βέλτιστο από αυτό που θα πάρουμε αν χρησιμοποιήσουμε τιμές που αφορούν μόνο τις πρώτες ή μόνο τις τελευταίες περιόδους. Επιπροσθέτως μια ολόκληρη περίοδος μας δίνει καλύτερη πρόβλεψη από τη μισή περίοδο.

7.3 Μηνιαίες απαιτήσεις για προηγμένους μικροϋπολογιστές

7.3.1 Γραμμικό μοντέλο

Το αρχείο comp περιέχει τις Μηνιαίες Απαιτήσεις μίας εταιρείας σε Προηγμένους Μικροϋπολογιστές. Οι τιμές που περιέχονται στο αρχείο είναι 48. Θα χρησιμοποιήσουμε ένα γραμμικό νευρωνικό δίκτυο για να μπορέσουμε να προβλέψουμε, με βάση προηγούμενες τιμές, την επόμενη τιμή της χρονοσειράς που έχουμε στη διάθεση μας. Τα δεδομένα του αρχείου θα τα φορτώσουμε στο περιβάλλον του Matlab χρησιμοποιώντας την εντολή load.
Εκτελώντας την εντολή load comp φορτώνονται στο Matlab τα περιεχόμενα του αρχείου comp.dat και οι τιμές του τοποθετούνται σ' ένα πίνακα διάνυσμα με όνομα y.
Με την εντολή prediction(y,nin,train) μπορούμε να κάνουμε την πρόβλεψη της επόμενης τιμής της χρονοσειράς. Όπου y είναι η προς πρόβλεψη χρονοσειρά comp, στο nin βάζουμε το πλήθος των προηγούμενων δειγμάτων που χρησιμοποιούνται για την πρόβλεψη και train ένα διάνυσμα στο οποίο τοποθετούμε το τμήμα της χρονοσειράς που θέλουμε να χρησιμοποιήσουμε για την εκπαίδευση του δικτύου αν train=[m,n], τότε για την εκπαίδευση χρησιμοποιείται το τμήμα της ακολουθίας ΄ y΄ για παράδειγμα από το m-οστό ως το n-οστό δείγμα. Παρακάτω ακολουθούν κάποια πειράματα που έγιναν χρησιμοποιώντας διαφορετικούς συνδυασμούς των nin και train:

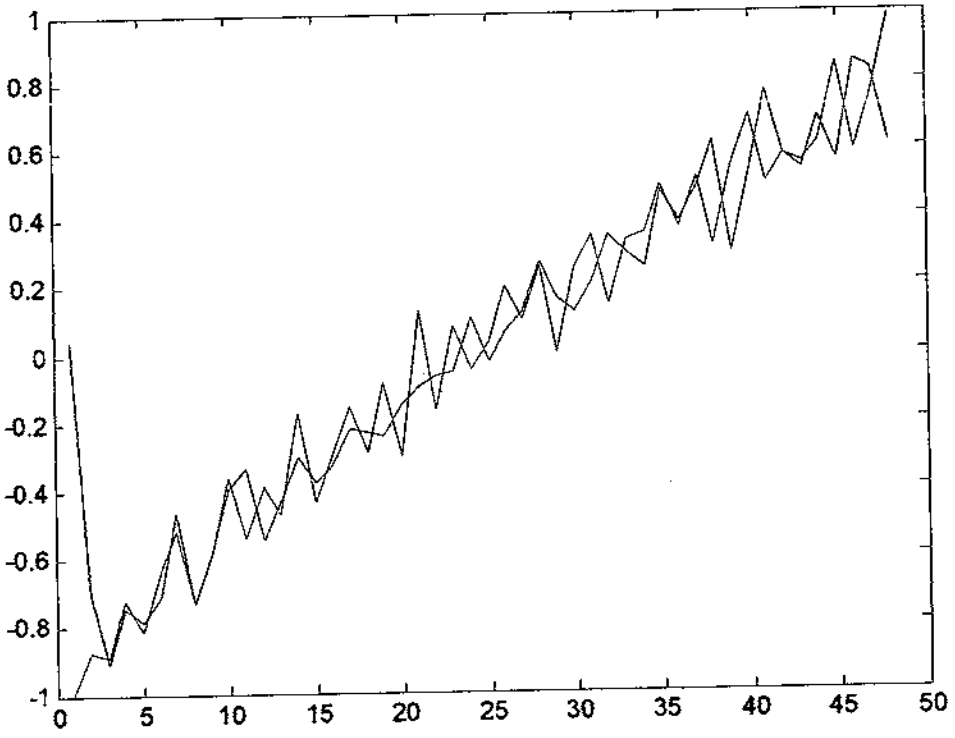
1) prediction(y,5,[2:14 24:35])



Σχήμα 68: *Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου*

MSE =2.2053

2) prediction(y,10,[2:14 24:35])



Σχήμα 69: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου

MSE =2.0940

1. Με αφορμή τα παραπάνω παραδείγματα καταλήγουμε στο συμπέρασμα ότι Όταν κρατάμε σταθερή την εκπαίδευση (train) και μεταβάλλουμε μόνο το πλήθος των προηγούμενων δειγμάτων, παρατηρούμε ότι έχουμε μια αύξηση του MSE καθώς αυξάνεται το nin. Ενώ μετά από το nin=5 το MSE αρχίζει να μειώνεται φτάνοντας στη βέλτιστη τιμή του στο nin=10 που έχουμε MSE=2,0940. στη συνέχεια όμως αυξάνοντας το nin έχουμε πάλι αύξηση του MSE.
2. Χρησιμοποιώντας για την εκπαίδευση τιμές από όλες τι περιόδους η πρόβλεψη είναι καλύτερη από ότι αν χρησιμοποιούμε τιμές από τις 2 πρώτες περιόδους ή τις τελευταίες. Αυτό σημαίνει ότι όσο αντιπροσωπευτικότερο το δείγμα εκπαίδευσης, τόσο καλύτερη προσέγγιση πετυχαίνει.

7.3.2 Προσαρμοστικό μοντέλο

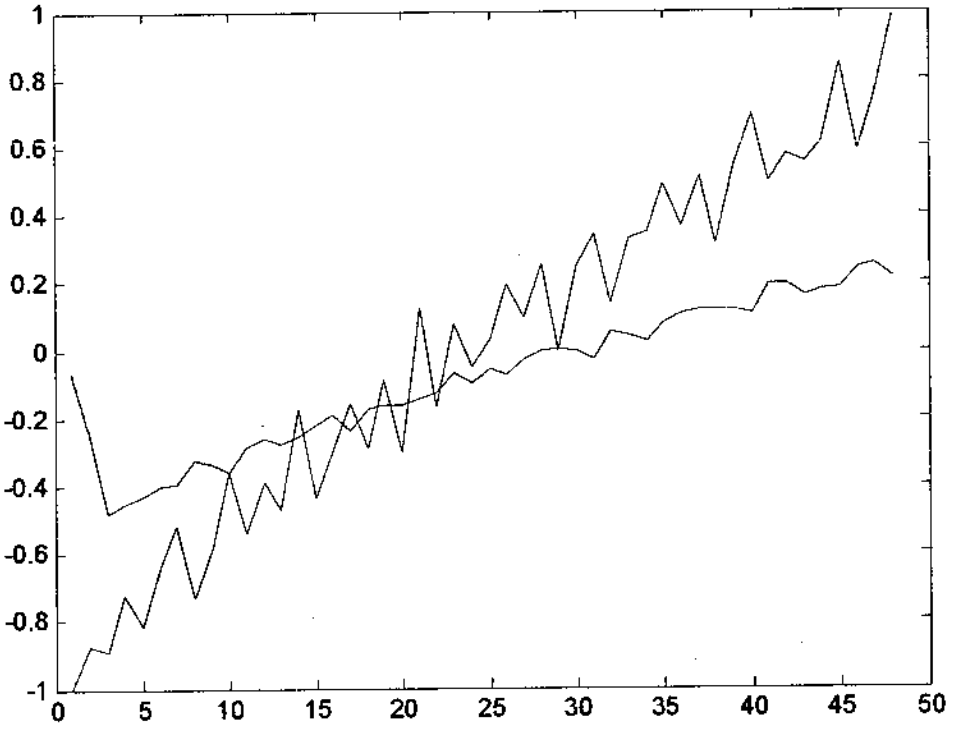
Στην προσπάθεια να έχουμε την καλύτερη δυνατή πρόβλεψη επόμενης τιμής με το ελάχιστο μέσο τετραγωνικό σφάλμα χρησιμοποιήσαμε και το προσαρμοστικό νευρωνικό μοντέλο.

Για το σκοπό αυτό χρησιμοποιούμε την συνάρτηση prediction_ad(y,nin,train) με την μπορούμε να κάνουμε την πρόβλεψη της επόμενης τιμής της χρονοσειράς.

Αξιοσημείωτο είναι ότι στο προσαρμοστικό μοντέλο μπορούμε να μεταβάλλουμε εκτός από τις τιμές των ορισμάτων nin και train το ρυθμό μάθησης(learning rate) όπως και το πλήθος των επαναλήψεων(passes) που κάνει το προσαρμοστικό δίκτυο στη χρονοσειρά για να την μάθει. Επειδή η προσαρμοστική πρόβλεψη ξεκινά από τυχαίες τιμές στα βάρη κάθε φορά ακολουθούμε την στοχαστική πρόβλεψη, δηλαδή χρησιμοποιούμε τη μέση τιμή των μέσων τετραγωνικών σφαλμάτων. Έτσι εξαλείφεται η τυχαιότητα των bias.

Παρακάτω ακολουθούν κάποιες εφαρμογές που έγιναν χρησιμοποιώντας διαφορετικούς συνδυασμούς των nin , train, learning rate και passes που η πρόβλεψη τους είναι αρκετά καλή και το σφάλμα τους μικρό:

1) Lr:0.028
   Passes=5
   Train=[2:6 12:14 22:28 34:44]
   Nin=2

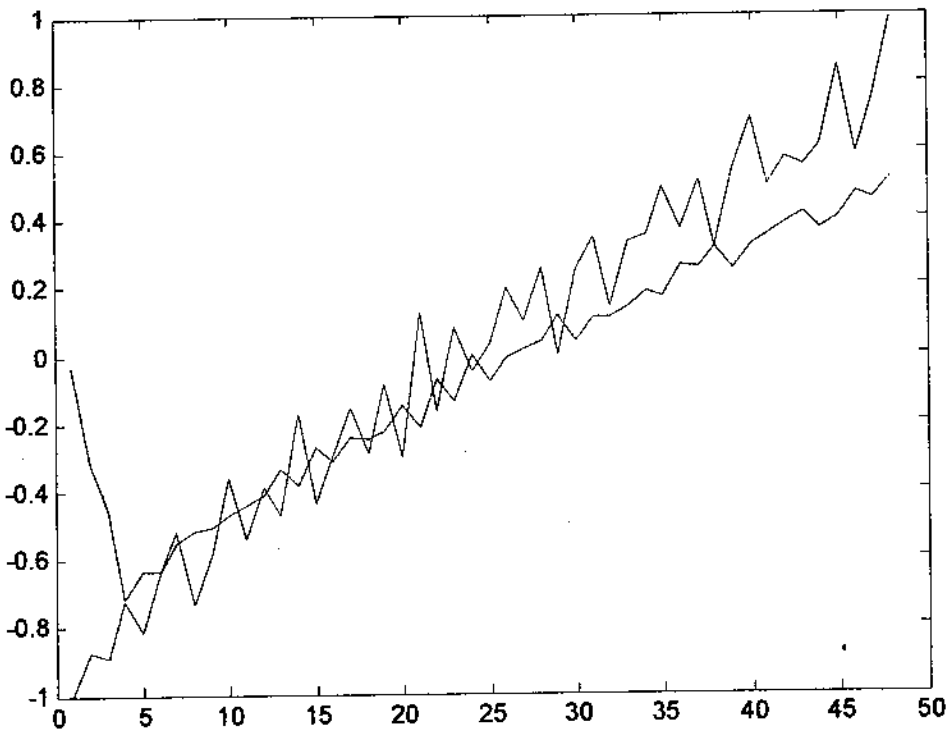


Σχήμα 70: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου

MSE =9.4203

2) Lr:0.028
   Passes=5
   Train=[2 6 12 18 22 29 34 30 42 46]

Nin=3

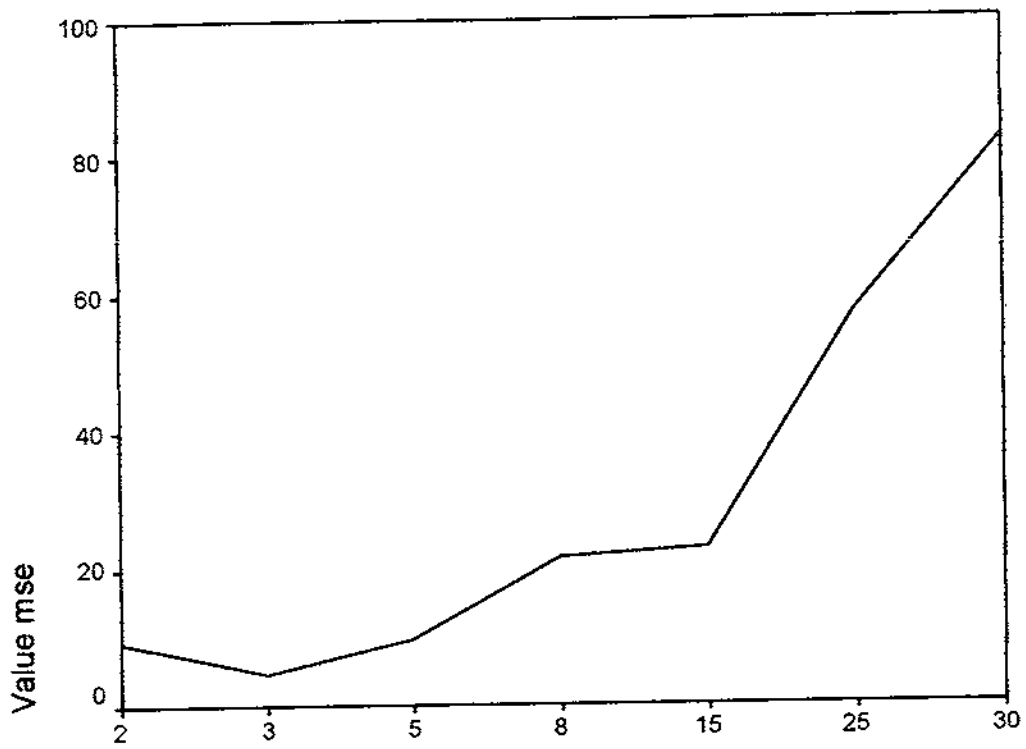


Σχήμα 71: *Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου*
MSE =4.6714

Μετά από αρκετά πειράματα εξάγουμε τα εξής συμπεράσματα:

1. Για την εκπαίδευση χρησιμοποιήσαμε τιμές από όλες τις περιόδους train:[2 6 12 18 22 29 30 34 42 46], το learning rate που μας έδινε τα καλύτερα αποτελέσματα ήταν learning rate=0.028 και τα passes είναι 5. Κρατώντας όλες τις προηγούμενες μεταβλητές σταθερές και μεταβάλλοντας μόνο τα nin είχαμε τα παρακάτω αποτελέσματα:

NIN	MSE
2	9,4203
3	4,6714
5	9,6714
8	21,4168
15	22,9758
25	57,3976
30	82,3051



Σχήμα 72:Διάγραμμα πορείας του MSE για διάφορες τιμές του πλήθους των προηγούμενων δειγμάτων

Όπως απεικονίζεται και στο παραπάνω διάγραμμα όσο αυξάνονται τα nin το MSE μειώνεται, δηλ. η πρόβλεψη κάνει μια καλύτερη προσέγγιση για την επόμενη τιμή μέχρι ενός σημείου. Στη προκειμένη περίπτωση το βέλτιστο MSE είναι το 4,6714 με nin:3. Από εκεί και πέρα το μέσο τετραγωνικό σφάλμα αυξάνει όσο αυξάνονται τα nin. Αυτό σημαίνει ότι όταν έχουμε αρκετά μεγάλο πλήθος προηγούμενων δειγμάτων η πρόβλεψη του προσαρμοστικού μοντέλου δεν μπορεί να προσεγγίσει ικανοποιητικά την μελλοντική επόμενη τιμή.

2. Ανάλογη πορεία με αυτή του μέσου τετραγωνικού σφάλματος μεταβάλλοντας το πλήθος των προηγούμενων τιμών του δείγματος nin έχουμε και με την αύξηση του learning rate ή τα passes, κρατώντας όλες τις υπόλοιπες μεταβλητές για κάθε περίπτωση σταθερές.
3. Όσον αφορά την εκπαίδευση που είναι ένας πολύ σημαντικός παράγοντας της πρόβλεψης παρατηρούμε ότι: όταν παίρνουμε τιμές από όλες τις περιόδους το MSE είναι το βέλτιστο από αυτό που θα πάρουμε αν χρησιμοποιήσουμε τιμές που αφορούν μόνο τις πρώτες ή μόνο τις τελευταίες περιόδους. Επιπροσθέτως μια ολόκληρη περίοδος μας δίνει καλύτερη πρόβλεψη από τη μισή περίοδο.

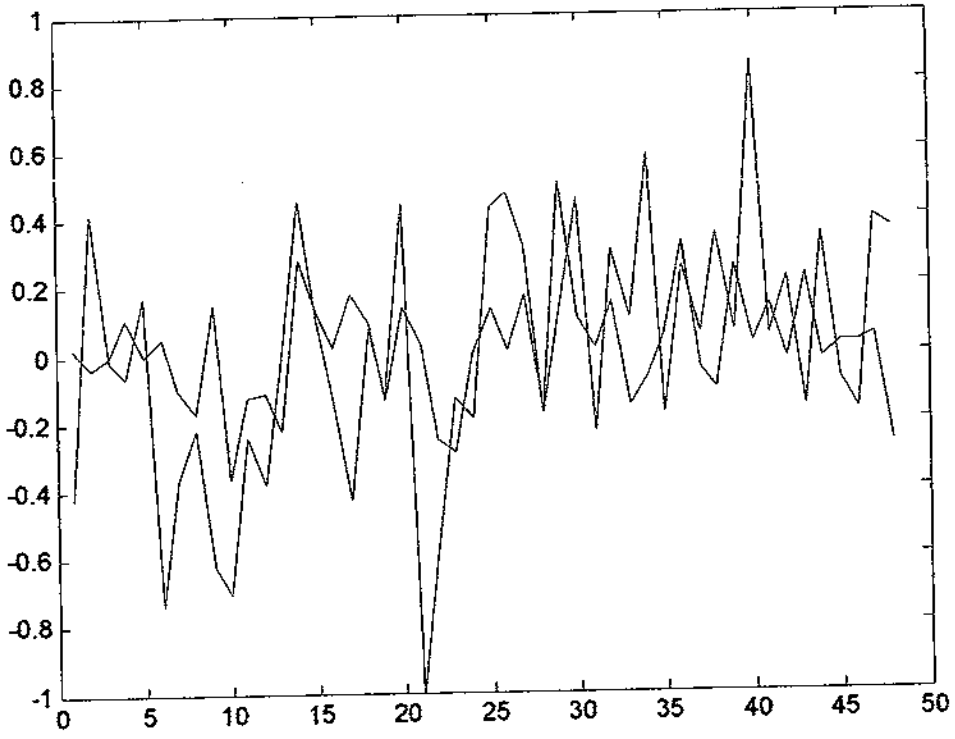
7.4 Μηνιαίες απαιτήσεις για χαρτί εκτυπωτή

7.4.1 Γραμμικό μοντέλο

Το αρχείο print περιέχει τις Μηνιαίες Απαιτήσεις μίας εταιρείας σε Χαρτί Εκτυπωτή. Οι τιμές που περιέχονται στο αρχείο είναι 48. Θα χρησιμοποιήσουμε ένα γραμμικό νευρωνικό δίκτυο για να μπορέσουμε να προβλέψουμε, με βάση προηγούμενες τιμές, την επόμενη τιμή της χρονοσειράς που έχουμε στη διάθεση μας. Τα δεδομένα του αρχείου θα τα φορτώσουμε στο περιβάλλον του Matlab χρησιμοποιώντας την εντολή load.
Εκτελώντας την εντολή load print φορτώνονται στο Matlab τα περιεχόμενα του αρχείου print.dat και οι τιμές του τοποθετούνται σ' ένα πίνακα διάνυσμα με όνομα x.
Με την εντολή prediction(x,nin,train) μπορούμε να κάνουμε την πρόβλεψη της επόμενης τιμής της χρονοσειράς. Όπου x είναι η προς πρόβλεψη χρονοσειρά print,στο nin βάζουμε το πλήθος των προηγούμενων δειγμάτων που χρησιμοποιούνται για την πρόβλεψη και train ένα διάνυσμα στο οποίο τοποθετούμε το τμήμα της χρονοσειράς που θέλουμε να χρησιμοποιήσουμε για την εκπαίδευση του δικτύου αν train=[m,n], τότε για την εκπαίδευση χρησιμοποιείται το τμήμα της ακολουθίας ΄ x΄ για παράδειγμα από το m-οστό ως το n-οστό δείγμα.

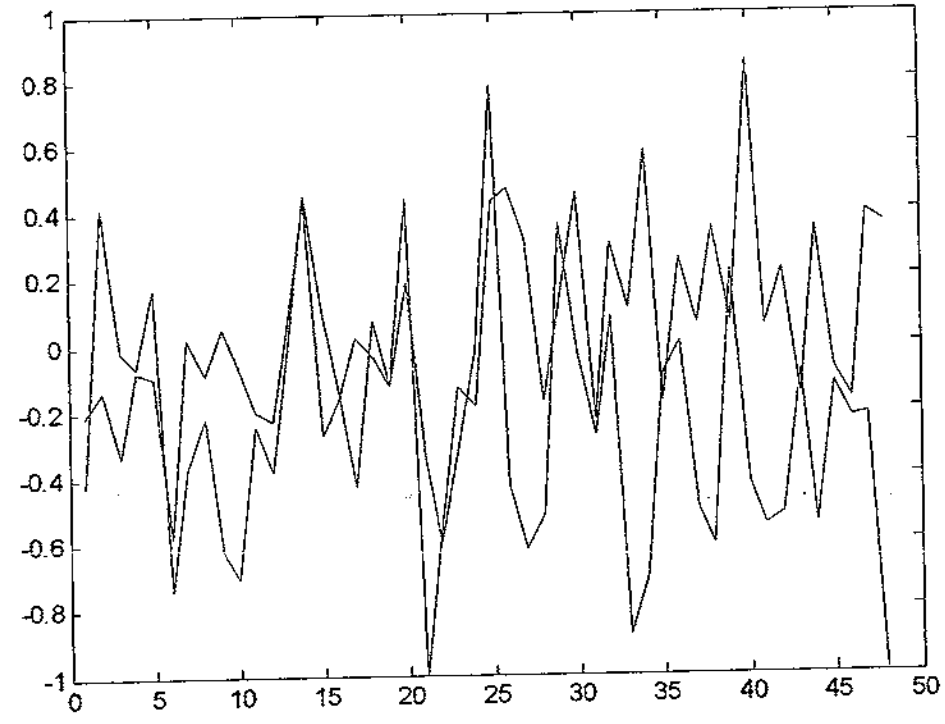
Παρακάτω ακολουθούν κάποια πειράματα που έγιναν χρησιμοποιώντας διαφορετικούς συνδυασμούς των nin και train:

1) prediction(x,10,[1:5 10:15 20:35])



Σχήμα 73: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου
MSE =6.8211

2) prediction(x,10,[2:8 12:24])



Σχήμα 74: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου

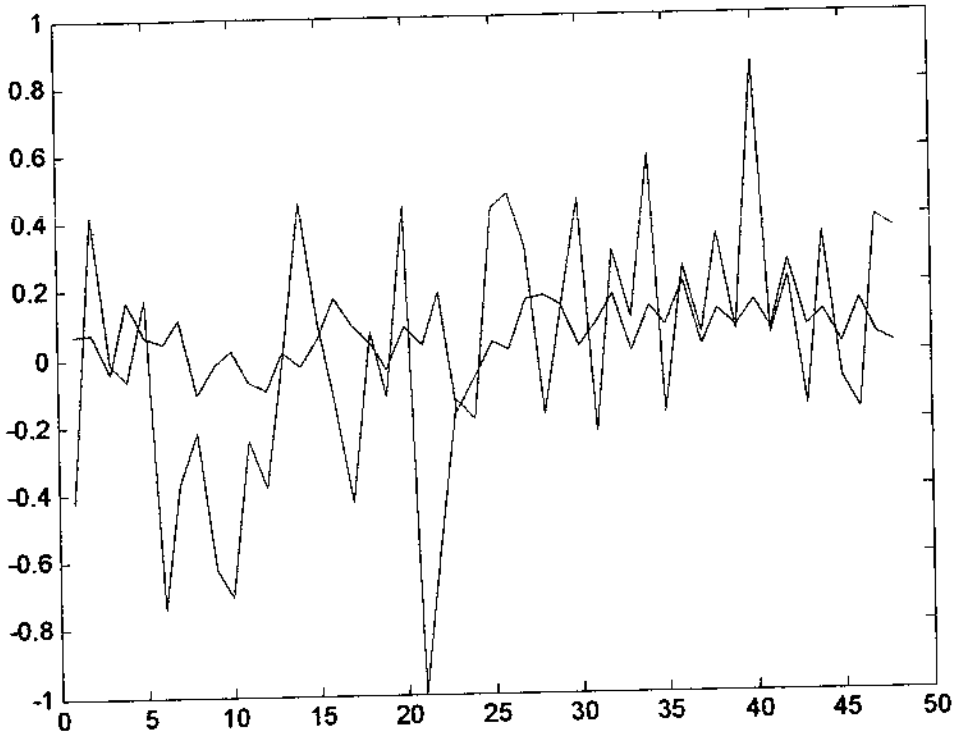
MSE =14.3141

3. Με αφορμή τα παραπάνω παραδείγματα καταλήγουμε στο συμπέρασμα ότι χρησιμοποιώντας για την εκπαίδευση τιμές από όλες τι περιόδους η πρόβλεψη είναι καλύτερη από ότι αν χρησιμοποιούμε τιμές από τις 2 πρώτες περιόδους. Αυτό σημαίνει ότι όσο αντιπροσωπευτικότερο είναι το δείγμα εκπαίδευσης, τόσο καλύτερη προσέγγιση πετυχαίνει. Αυτό γίνεται φανερό και από τις τιμές του μέσου τετραγωνικού σφάλματος.
4. Όταν κρατάμε σταθερή την εκπαίδευση (train) και μεταβάλλουμε μόνο το πλήθος των προηγούμενων δειγμάτων, παρατηρούμε ότι έχουμε μια αύξηση του MSE καθώς αυξάνεται το nin. Ενώ μετά από το nin=8 το MSE αρχίζει να μειώνεται φτάνοντας στη βέλτιστη τιμή του στο nin=15 που έχουμε MSE=6.0186. στη συνέχεια όμως αυξάνοντας το nin έχουμε πάλι αύξηση του MSE.

7.4.2 Προσαρμοστικό μοντέλο

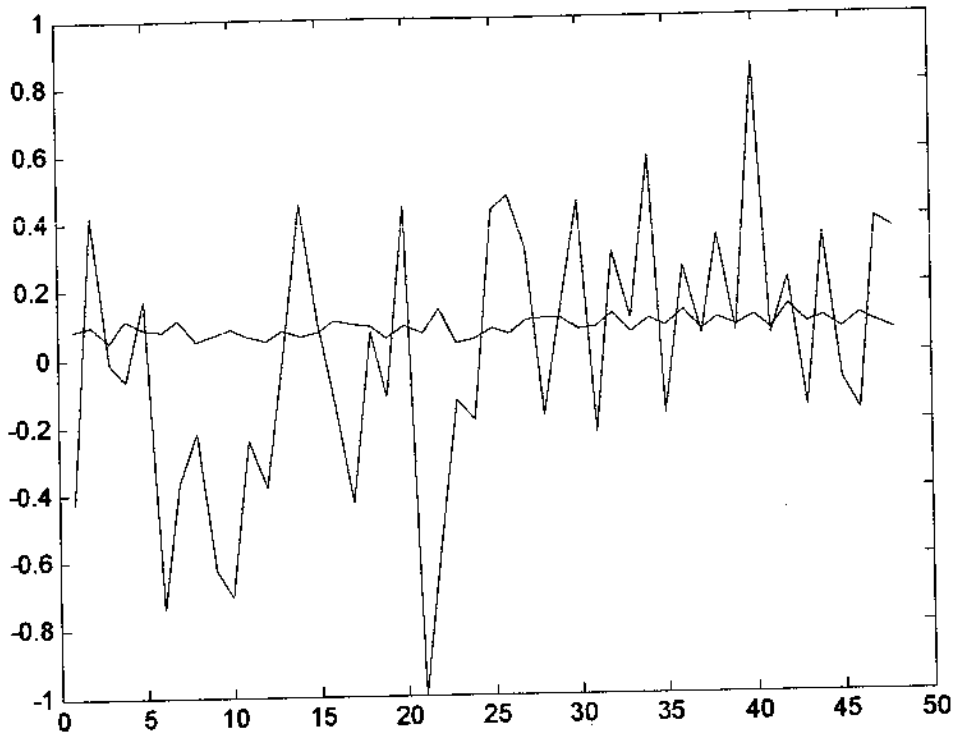
Στην προσπάθεια να έχουμε την καλύτερη δυνατή πρόβλεψη επόμενης τιμής με το ελάχιστο μέσο τετραγωνικό σφάλμα χρησιμοποιήσαμε και το προσαρμοστικό νευρωνικό μοντέλο.
Για το σκοπό αυτό χρησιμοποιούμε την συνάρτηση prediction_ad(x,nin,train) με την μπορούμε να κάνουμε την πρόβλεψη της επόμενης τιμής της χρονοσειρας.
Αξιοσημείωτο είναι ότι στο προσαρμοστικό μοντέλο μπορούμε να μεταβάλλουμε εκτός από τις τιμές των ορισμάτων nin και train το ρυθμό μάθησης(learning rate) όπως και το πλήθος των επαναλήψεων(passes) που κάνει το προσαρμοστικό δίκτυο στη χρονοσειρά για να την μάθει. Επειδή η προσαρμοστική πρόβλεψη ξεκινά από τυχαίες τιμές στα βάρη κάθε φορά ακολουθούμε την στοχαστική πρόβλεψη, δηλαδή χρησιμοποιούμε τη μέση τιμή των μέσων τετραγωνικών σφαλμάτων. Έτσι εξαλείφεται η τυχαιότητα των bias.
Παρακάτω ακολουθούν δύο εφαρμογές που έγιναν χρησιμοποιώντας διαφορετικούς συνδυασμούς των nin , train, learning rate και passes που η πρόβλεψη τους είναι αρκετά καλή και το σφάλμα τους μικρό:

1)Learning rate:0.025
Passes=2
Nin:2
Train[2 6 8 12 16 20 23 26 30 34 38 42]



Σχήμα 75: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου
MSE =7.0856

2) Learning rate:0.025
Passes=4
Nin=2
Train[2 6 8 12 16 20 23 26 30 34 38 42]

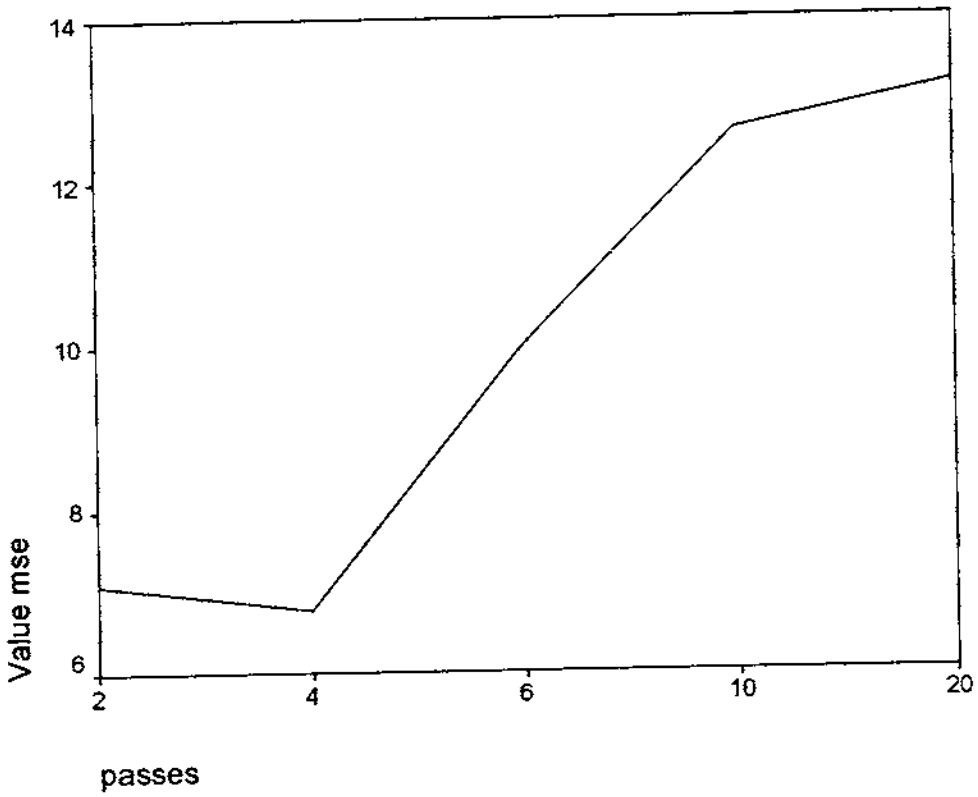


Σχήμα 76: Διάγραμμα σύγκρισης της εξόδου πρόβλεψης του συστήματος και της εξόδου του δικτύου
MSE =6.7788

Μετά από αρκετά πειράματα εξάγουμε τα εξής συμπεράσματα:

1. Οι προηγούμενες τιμές που χρησιμοποιήσαμε για πρόβλεψη είναι nin=2, για την εκπαίδευση χρησιμοποιήσαμε τιμές από όλες τις περιόδους train:[2 6 8 12 16 20 23 26 30 34 38 42] και ο ρυθμός μάθησης που μας έδινε τα καλύτερα αποτελέσματα ήταν learning rate:0.025. Κρατώντας όλες τις προηγούμενες μεταβλητές σταθερές και μεταβάλλοντας μόνο τις επαναλήψεις - passes είχαμε τα παρακάτω αποτελέσματα:

PASSES	MSE
2	7,0856
4	6,7788
6	9,9838
10	12,6260
20	13,1696



Σχήμα 77: Διάγραμμα πορείας του MSE για διάφορες τιμές των επαναλήψεων

Όπως απεικονίζεται και στο παραπάνω διάγραμμα όσο αυξάνονται τα passes το MSE μειώνεται, δηλ. η πρόβλεψη κάνει μια καλύτερη προσέγγιση για την επόμενη τιμή μέχρι ενός σημείου. Στη προκειμένη περίπτωση το βέλτιστο MSE είναι το 6,7788 με passes=4. Από εκεί και πέρα το μέσο τετραγωνικό σφάλμα αυξάνει όσο αυξάνει το learning rate.Αυτό σημαίνει ότι οι πολλές επαναλήψεις κάνουν τη μάθηση του νευρώνα ασταθή.

2. Ανάλογη πορεία με αυτή του μέσου τετραγωνικού σφάλματος μεταβάλλοντας τα passes έχουμε και με την αύξηση του learning rate ή το πλήθος των προηγούμενων τιμών του δείγματος nin, κρατώντας όλες τις υπόλοιπες μεταβλητές για κάθε περίπτωση σταθερές.
3. Όσον αφορά την εκπαίδευση που είναι ένας πολύ σημαντικός παράγοντας της πρόβλεψης παρατηρούμε ότι: όταν παίρνουμε τιμές από όλες τις περιόδους το MSE είναι το βέλτιστο από αυτό που θα πάρουμε αν χρησιμοποιήσουμε τιμές που αφορούν μόνο τις πρώτες ή μόνο τις τελευταίες περιόδους. Επιπροσθέτως μια ολόκληρη περίοδος μας δίνει καλύτερη πρόβλεψη από τη μισή περίοδο.

Βιβλιογραφία

ΕΛΛΗΝΙΚΗ

- Μαλαβάζος Κ., (2002), "*Γενετικοί Αλγόριθμοι και το Πρόβλημα Ταυτοποίησης Συστημάτων ARMA*", Master Thesis, Τμήμα Η/Υ και Πληροφορικής Πανεπιστημίου Πατρών, Πάτρα.
- Ταμπάκης Π., (2004, Ιούνιος) "*Τεχνητά Νευρωνικά Δίκτυα και Πρόβλεψη Οικονομικών Χρονοσειρών*", Πτυχιακή Εργασία, Α.Τ.Ε.Ι. Πατρών, ΣΔΟ, Τμήμα Επιχειρηματικού Σχεδιασμού & Πληροφοριακών Συστημάτων, Πάτρα.
- Ψωμάς Π., (1996), "*Πολλαπλά Νευρωνικά Δίκτυα και Εφαρμογές σε Πρόβλεψη Χρονοσειρών*", Διδακτορική Διατριβή, Εθνικό Μετσόβιο Πολυτεχνείο, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών.

ΞΕΝΗ

- Anderson, J. A. (1977). Neural models with cognitive implications. In D. LaBerge & S. J.Samuels (Eds.), "*Basic Processes in Reading Perception and Comprehension Models* ",(p.27-90). Hillsdale, NJ: Erlbaum.
- Azoff, E.M., (1994), "*Neural Network Time Series Forecasting of Financial Markets*" , Chichester, John Wiley & Sons – (Wiley finance editions).
- Braspenning P.J., Thuijman, F., Weijter, Ed., (1991), "*Artificial Neural Networks: An Introduction to ANN Theory and Practice*", Springer, Hiedelberg, Germany.
- Cybenko, G. (1989), "*Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems*", 2(4), 303-314.
- Funahashi, K.-I. (1989), "*On the approximate realization of continuous mappings by neural networks. Neural Networks*", 2(3), 193-192.
- Hagan, M.T. and H.B. Demuth, (1999), "*Neural Networks for Control*", Proceedings of the 1999 American Control Conference, San Diego, CA, pp.1642-1656.
- Hagan, M.T., O. De Jesus, and R. Schultz, (1999), "*Training Recurrent Networks for Filtering and Control*", Chapter 12 in Recurrent Neural Networks: Design and Applications, L. Medsker and L.C. Jain, Eds, CRC Press, pp. 311-340.
- Hartman, E. J., Keeler, J. D., & Kowalski, J. M. (1990), "*Layered neural networks with Gaussian hidden units as universal approximations. Neural Computation*" 2(2), 210-215.
- Haupt R.L. and S. Haupt, (1997), "*Practical Genetic Algorithms*", John Wiley & Sons, inc,.

- Haykin S., (2002), "*Adaptive Filter Theory*", Prentice Hall, U.S.A,.
- Hebb, D. O. (1949), "*The Organization of Behaviour*" New York: Wiley.
- Holland H., (1975) "*Adaptation on Natural and Artificial Systems*", Ann Arbor: The University of Michigan Press.
- Hopfield, J. J. (1982), "*Neural networks and physical systems with emergent collective computationalabilities. Proceedings of the National Academy of Sciences*" 79, 2554-2558.
- Hornik, K., Stinchcombe, M., &White, H. (1989), "*Multilayer feedforward networks are universal approximators. Neural Networks*", 2(5), 359-366.
- Hunt, K.J., D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop, (1992), "*Neural Networks for Control System – A Survey*", "Automatica", Vol. 28, pp. 1083-1112.
- Kohonen, T. (1977), "*Associative Memory: A System-Theoretical Approach*", Springer-Verlag.
- Krose Ben and Patrick van der Smagt, (1996), "*Artificial Neural Networks: An Introduction to Neural Networks*", University of Amsterdam.
- McCulloch, W. S., & Pitts, W. (1943), "*A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics*", 5, 115-133.
- Minsky, M., & Papert, S. (1969), "*Perceptrons: An Introduction to ComputationalGeometry*", The MIT Press.
- Murray, R., D. Neumerkel, and D. Sbarbaro, (1992), "*Neural Networks for Modeling and Control of a Non-linear Dynamic System*", "*Proceedings of the 1992 IEEE International Symposium on Intelligent Control*", pp. 404-409.
- Narendra, K.S. and S. Mukhopadhyay, (1997), "*Adaptive Control Using Neural Networks and Approximate Models*", "*IEEE Transactions on Neural Networks*", Vol. 8, pp. 475-485.
- Rosenblatt, F. (1959), "*Principles of Neurodynamics*", New York: Spartan Books.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986), "*Learning representations by backpropagating errors*", Nature, 323, 533-536.
- Soloway, D. and P.J. Haley, (1996), "*Neural Generalized Predictive Control*" ,Proceedings of the 1996 IEEE International Symposium on Intelligent Control, pp. 277-281.
- Tang K.S., K.F. Man, S. Kwong and Q. He, (November 1996), "*Genetic Algorithms and their Applications*", IEEE Signal Processing Magazine, pp 22-37.
- Widrow, B., & Hoff, M. E. (1960), "*Adaptive switching circuits*", In 1960 IRE WESCON Convention Record (pp. 96-104). DUNNO.

Widrow Bernard and Samuel Stearns, (1985), "*Adaptive Signal Processing,*" Prentice Hall inc. Signal Processing Series, U.S.A.

ΗΛΕΚΤΡΟΝΙΚΗ

http://www.doc.ic.ac.uk/~ nd/surprise_96/journal/vol4/cs11/report.html

http://www.ee.ucr.edu/~yhua/ee211/Note_2.pdf

http://www.egnatia.ee.auth.gr/~imatz/ann_begin.html

http://www.gc.ssr.upm.es/inves/neural/annl/supmodel/linanet.htm

http://www.iau.dtu.dk/~jj/pubs/nnet.pdf

http://library.thinkquest.org/18242/ga_math.shtml

http://www.neuromod.org/courses/connectionism1999/backpropagation/sld003.htm

http://scholar.lib.vt.edu/theses/available/etd-122099-153321/unrestricted/chapter01.pdf

http://www.telecomlab.oulu.fi/home/coursematerial/adapt_LMS.pdf

Παραρτήματα

Λεξικό μερικών βασικών ορών

ATM	**Αερογραμμή Μάρκετινγκ Tactician**
ART	**Adaptive Resonance Theory**
LMS	**Least Mean Square**
M	**Misadjustment**
MSE	**Mean Square Error**
ΤΝΔ	**Τεχνητό Νευρωνικό Δίκτυο**

Δεδομένα Μηνιαίων Πωλήσεων Αυτοκινήτων

Για n=185 τιμές

3975 1233 1488 1665 1889 2104 2139 1905 1674 1466 1191 943 907 1183 1491
1634 1865 2048 2105 1961 1680 1501 1313 953 865 1060 1360 1453 1739 1973
2023 1874 1540 1450 1402 1015 875 982 1309 1387 1630 1942 2047 1862 1487
1428 1451 1018 907 1064 1382 1449 1700 2066 3110 1968 1597 1547 1538 1084
990 1119 1452 1508 1793 2191 2222 2187 1788 1735 1720 1339 1265 1405 1683
1677 2034 2420 2427 2388 1938 1922 1905 1526 1421 1536 1825 1812 2211 2558
2528 2400 1952 2016 2027 1674 1575 1553 1841 1807 2133 2417 2361 2239 1816
1934 2093 1730 1641 1480 1786 1777 2077 2284 2247 2196 1813 1787 1987 1609
1512 1250 1645 1679 1964 2176 2140 2069 1627 1594 1802 1395 1311 1035 1502
1565 1839 2037 1986 1855 1404 1441 1612 1213 1126 899 1322 1291 1542 1792
1799 1586 1153 1212 1303 881 825 704 1134 1150 1369 1658 1617 1371 889 947
982 576 541 446 886 826 1029 1359 1305 1083 644 759 860 481 470 446 894 812
931

Δεδομένα Μηνιαίων Απαιτήσεων για Χαρτί Εκτυπωτή

Για n=48 τιμές

779 920 848 839 879 726 788 813 745 732 809 786 849 926 867 828 778 862 830
924 682 754 828 818 922 929 903 820 871 926 811 901 867 948 820 893 859 909
860 993 858 887 823 909 837 821 916 911

Δεδομένα Μηνιαίων Απαιτήσεων για Προηγμένους Μικροϋπολογιστές

Για n=48 τιμες

493 536 530 588 556 617 656 584 634 709 648 700 672 773 684 730 779 733 803
729 874 776 858 815 843 899 864 917 830 916 947 879 944 950 997 956 1006
937 1016 1066 999 1026 1020 1038 1119 1030 1084 1166

Δεδομένα Μηνιαίων Απαιτήσεων για Αναψυκτικά με Χαμηλές Θερμίδες

Για n=48 τιμές

518 572 599 652 692 759 705 643 600 546 518 426 554 585 633 688 731 794 805
726 717 651 605 550 463 497 564 582 633 686 676 657 602 534 499 472 459 519
552 589 653 684 766 678 649 582 537 475

Κώδικας γραμμικής πρόβλεψης:

```
function [output,e,MSE]=prediction(data,Nin,train);

data=(data-mean(data))/max(abs(data-mean(data)));

A=zeros(Nin,length(data));
for i=1:Nin
   A(i,i+1:end)=data(1:end-i);
end


net=newlind(A(:,train),data(train));

output=sim(net,A);

figure(1);plot(1:length(data),data,1:length(data),output);

e=output-data;

MSE=sum(e.^2)

figure(2);plot(data(Nin:end)-output(1:end-Nin+1))
```

Κώδικας προσαρμοστικής πρόβλεψης:

```
function [output,e,MSE,A]=prediction_ad(data,Nin,train);

data=(data-mean(data))/max(abs(data-mean(data)));

A=zeros(Nin,length(data));
for i=1:Nin
   A(i,i+1:end)=data(1:end-i);
   Range(i,1:2)=[min(A(i,:)) max(A(i,:))];

end
net = nnt2lin(Range,randn(1,Nin),randn(1,1),0.020);
%net = nnt2lin(Range,[-1 1 1 -1 1],.0*randn(1,1),0.03);
net.adaptParam.passes=2;
[net,output,e,Pf]=adapt(net,A(:,train),data(train));
%pause
output=sim(net,A);


e=output-data;
MSE=sum(e.^2)
%plot(output)
%e=data(Nin:end)-output(1:end-Nin+1);
%pause
%size(e)
%figure(2);plot(data(Nin:end)-output(1:end-Nin+1))
```

Η εντολή που τρέξαμε σε σχέση με τον παραπάνω κώδικα είναι η παρακάτω:

```
K=0; O=zeros(1,48);E=O;mbifla=[3 5 8 10 13]
for i=1:100 [output,e,MSE,A]=prediction_ad(data,6,mbifla);
if MSE<100 E=E+e;
O=O+output;else K=K+1;end
end
O=O/(100-K);
E=E/(100-K);
```