ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Διπλωματική εργασία της φοιτήτριας ***Σοφίας Μπαριάμη***

«Μελέτη χαοτικών και ψευδοχαοτικών συστημάτων με στατιστικά εργαλεία και η εξερεύνηση της δυνατότητας πρόγνωσής τους από ένα reservoir computer.»

Τριμελής επιτροπή: Θεόδωρος Αλεξόπουλος (επιβλέπων καθηγητής)
Σταύρος Μαλτέζος
Ευάγγελος Γαζής

Η διπλωματική εργασία έγινε σε συνεργασία με τον κύριο Χάρη Μεσαριτάκη, (MSc, PhD, Associate Researcher) του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών.

Αθήνα, Οκτώβριος 2015

By far the greatest danger of Artificial Intelligence
is that people conclude too early that they understand it.
- Eliezer Yudkowski

Περίληψη

Η συγκεκριμένη διπλωματική εργασία αφορά την πρόβλεψη χαοτικών χρονοσειρών NARMA με τη χρήση νευρωνικών δικτύων και τη συσχέτιση των σφαλμάτων ελέγχου με τη πολυπλοκότητα της χρονοσειράς. Αυτό θα μας επιτρέψει να προβούμε σε συμπεράσματα σχετικά με το πόσο ασφαλής μπορεί να είναι η κρυπτογράφηση κάποιας πληροφορίας, η οποία έχει κωδικοποιηθεί μέσα σε κάποια χρονοσειρά.

Αρχικά γίνεται μια εισαγωγή στις έννοιες που θα χρειαστούμε για τη κατανόηση της διπλωματικής εργασίας, καθώς και μια ιστορική αναδρομή σχετικά με το πώς εξελίχθηκαν τα νευρωνικά δίκτυα στο χρόνο. Στη συνέχεια, αφού δούμε κάποιους τύπους νευρωνικών δικτύων, θα εξειδικευτούμε περισσότερο σε έναν συγκεκριμένο τύπο, το ESN (echo state network) το οποίο θα χρησιμοποιήσουμε και για τους υπολογισμούς στη συνέχεια. Τέλος, θα αναφερθούμε στη θεωρία του χάους και το πώς οι εκθέτες Lyapunov, που μας δίνουν μια εκτίμηση για το μέτρο της χαοτικότητας της χρονοσειράς μου μελετάμε, μπορούν να συσχετιστούν με το μέγεθος του σφάλματος που μας δίνει το ESN.

Για την εκπόνηση της διπλωματικής χρησιμοποιήθηκαν πακέτα του μαθηματικού λογισμικού Matlab.

Abstract

The present thesis deals with the prediction of chaotic NARMA time series using neural network and the connection of the testing error with the complexity of the time series. This will allow us to draw conclusions about the safety of the information encryption that has been encoded in a time series.

First of all, there is an introduction to the terms that are going to be used at the thesis and a throwback concerning how neural networks evolved in time. Then, after citing some types of neural networks, we will specialize in a particular type, ESN (echo state network) which will be used for further calculations. Finally, we will mention the chaos theory and how the Lyapunov exponents that give us an estimation of the chaotic behavior of the time series, can be connected with the magnitude of the testing error that the ESN gives us.

Toolboxes of the mathematical software Matlab were used.

Περιεχόμενα

Σελίδα

1. Εισαγωγή 8

1.1 Εισαγωγικές έννοιες 8
1.2 Ιστορική αναδρομή 11
1.3 Είδη νευρωνικών δικτύων 13
1.4 Κανόνες εκμάθησης 17
1.5 Πλεονεκτήματα τεχνιτών νευρωνικών δικτύων 22
1.6 Το μοντέλο του τεχνητού νευρώνα 24
1.7 Εφαρμογές 27

2. Τα ESN σε χρήση..... 30

2.1: Γενικά 30
2.2: Εκμάθηση από το σύνολο των δεδομένων (batch learning).....31
2.2.1: Συσχέτιση αριθμού νευρώνων- σφαλμάτων 31
2.2.2 Συσχέτιση ποσοστού δεδομένων για εκπαίδευση- σφαλμάτων 40
2.3: Εκμάθηση σε πραγματικό χρόνο (online learning) 43
2.3.1: Συσχέτιση αριθμού νευρώνων- σφαλμάτων 43
2.3.2 Συσχέτιση ποσοστού δεδομένων για εκπαίδευση- σφαλμάτων 54

3. Οι εκθέτες Lyapunov στον προσδιορισμό χαοτικών συστημάτων 56

3.1 Γενικά 56
3.2 Σύγκριση πολυπλοκότητας – σφαλμάτων 58
3.3 Υπολογισμός εκθετών Lyapunov.....62
3.3.1 Γενικά.....62
3.3.2 Μέθοδος Sato για τον υπολογισμό του μέγιστου εκθέτη Lyapunov 64
3.3.2.1 Ανακατασκευή του χώρου των καταστάσεων 65
3.3.2.2 Επιρροή των παραμέτρων της ανακατασκευής 66
3.3.3 Υπολογισμός εκθετών Lyapunov με χρήση Matlab 69

4. **Συμπεράσματα ……..74**

5. **Παράρτημα (Κώδικες Matlab)…….…… 75**

6. **Βιβλιογραφία…….84**

1. Εισαγωγή

1.1 Εισαγωγικές έννοιες

1. Ο νευρώνας

Με τον όρο νευρώνας ονομάζουμε το κύτταρο που αποτελεί δομικό μέρος και λειτουργική μονάδα του νευρικού συστήματος. Κάθε νευρώνας αποτελείται από ένα κυτταρικό σώμα, που περιλαμβάνει τον πυρήνα και από μία ή περισσότερες αποφυάδες, που ονομάζονται "δενδρίτες".

Οι Νευρώνες επικοινωνούν μεταξύ τους και με άλλους νεύρωνες με ηλεκτρικές διεγέρσεις μεγάλης μεταβλητότητας μέσω συνάψεων που βρίσκονται στις άκρες των δενδρίτων.

Επίσης, μια ιδιαίτερα αξιοσημείωτη ιδιότητα είναι η ικανότητα των συνάψεων να εξελίσσονται και να προσαρμόζονται.

2. Τεχνητά νευρωνικά δίκτυα

Τα τεχνητά νευρωνικά δίκτυα είναι μια επιστημονική περιοχή, η οποία έχει αναπτυχθεί τις τελευταίες δεκαετίες και αφορά όλες σχεδόν τις θετικές επιστήμες. Τα νευρωνικά δίκτυα αποτελούνται από ένα σύνολο απλών, διασυνδεδεμένων και προσαρμοστικών μονάδων (νευρώνες), οι οποίες δημιουργούν ένα πολύπλοκο υπολογιστικό μοντέλο. Στην ουσία είναι προγράμματα που υλοποιούνται στους ηλεκτρονικούς υπολογιστές εμπνευσμένα από τους βιολογικούς νευρώνες.

Οι νευρώνες είναι τα δομικά στοιχεία του δικτύου. Κάθε τέτοιος κόμβος δέχεται ένα σύνολο αριθμητικών εισόδων από διαφορετικές πηγές (είτε από άλλους νευρώνες, είτε από το περιβάλλον), επιτελεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μία έξοδο. Η εν λόγω έξοδος είτε κατευθύνεται στο περιβάλλον, είτε τροφοδοτείται ως είσοδος σε άλλους νευρώνες του δικτύου.

Υπάρχουν τρεις τύποι νευρώνων: οι νευρώνες εισόδου, οι νευρώνες εξόδου και οι υπολογιστικοί νευρώνες ή κρυμμένοι νευρώνες. Οι **νευρώνες εισόδου** δεν επιτελούν κανέναν υπολογισμό, μεσολαβούν απλώς ανάμεσα στις περιβαλλοντικές εισόδους του δικτύου και στους υπολογιστικούς νευρώνες. Οι **νευρώνες εξόδου** διοχετεύουν στο περιβάλλον τις τελικές αριθμητικές εξόδους του δικτύου. Οι **υπολογιστικοί νευρώνες** πολλαπλασιάζουν κάθε είσοδό τους με το αντίστοιχο συναπτικό βάρος και υπολογίζουν το ολικό άθροισμα των γινομένων. Το άθροισμα αυτό τροφοδοτείται ως όρισμα στη συνάρτηση ενεργοποίησης, την οποία υλοποιεί εσωτερικά κάθε κόμβος. Η τιμή που λαμβάνει η συνάρτηση για το εν λόγω όρισμα είναι και η έξοδος του νευρώνα για τις τρέχουσες εισόδους και βάρη.

Εάν x_{ki} είναι η i-οστή είσοδος του k νευρώνα, w_{ki} το i-οστό συναπτικό βάρος του k νευρώνα και φ(.) η συνάρτηση ενεργοποίησης του νευρωνικού δικτύου, τότε η έξοδος y_k του k νευρώνα δίνεται από την εξίσωση:

$$y_k = \sum_{i=0}^{N} (x_{ki} w_{ki})$$

Στον k-οστό νευρώνα υπάρχει ένα συναπτικό βάρος w_{ko} με ιδιαίτερη σημασία, το οποίο καλείται πόλωση ή κατώφλι (bias, threshold). Η τιμή της εισόδου του είναι πάντα η μονάδα, $x_{ko}=1$. Εάν το συνολικό άθροισμα από τις υπόλοιπες εισόδους του νευρώνα είναι μεγαλύτερο από την τιμή αυτή, τότε ο νευρώνας ενεργοποιείται. Εάν είναι μικρότερο, τότε ο νευρώνας παραμένει ανενεργός. Η ιδέα προέκυψε από τα βιολογικά νευρικά κύτταρα.

Το κύριο χαρακτηριστικό των νευρωνικών δικτύων είναι η ικανότητα **μάθησης**. Ως μάθηση μπορεί να οριστεί η σταδιακή βελτίωση της ικανότητας του δικτύου να επιλύει κάποιο πρόβλημα (π.χ. η σταδιακή προσέγγιση μίας συνάρτησης). Η μάθηση επιτυγχάνεται μέσω της **εκπαίδευσης**, μίας επαναληπτικής διαδικασίας σταδιακής προσαρμογής των παραμέτρων του δικτύου (συνήθως των βαρών και της πόλωσής του) σε τιμές κατάλληλες ώστε να επιλύεται με επαρκή επιτυχία το προς εξέταση πρόβλημα. Αφού ένα δίκτυο εκπαιδευτεί, οι παράμετροί του συνήθως «παγώνουν» στις κατάλληλες τιμές και από εκεί κι έπειτα είναι σε λειτουργική κατάσταση. Το ζητούμενο είναι το λειτουργικό δίκτυο να χαρακτηρίζεται από μία **ικανότητα γενίκευσης**: αυτό σημαίνει πως δίνει ορθές εξόδους για εισόδους καινοφανείς και διαφορετικές από αυτές με τις οποίες εκπαιδεύτηκε.

3. Εκπαίδευση Νευρωνικών Δικτύων

Μια από τις πιο βασικές ιδιότητες των Νευρωνικών Δικτύων είναι η ικανότητά τους για εκπαίδευση. Η εκπαίδευση αυτή επιτυγχάνεται μέσω της ανταλλαγής τιμών και βαρών, που αποσκοπεί στη βαθμιαία σύλληψη της πληροφορίας η οποία στη συνέχεια θα είναι διαθέσιμη προς ανάκτηση.

Όλες οι μέθοδοι μάθησης μπορούν να καταταχτούν σε δύο κατηγορίες: τη μάθηση με επίβλεψη (supervised learning) και τη μάθηση χωρίς επίβλεψη (unsupervised learning). Κατά κύριο λόγο οι περισσότερες διαδικασίες εκπαίδευσης είναι off line. Όταν χρησιμοποιείται όλο το δείγμα προτύπων για την τροποποίηση των τιμών των βαρών, πριν την τελική χρήση του δικτύου ως εφαρμογή, τότε ονομάζεται off line εκπαίδευση. Οι αλγόριθμοι εκπαίδευσης off line έχουν την απαίτηση να βρίσκονται στην εκπαίδευση του δικτύου παρόντα όλα τα πρότυπα. Το πλεονέκτημα των δικτύων που χρησιμοποιούν off line διαδικασίες εκπαίδευσης επικεντρώνεται κυρίως στη δυνατότητα να δίνουν καλύτερες λύσεις σε δύσκολα προβλήματα.

Τους κανόνες εκπαίδευσης θα τους δούμε αναλυτικότερα παρακάτω

4. Reservoir Computing

Με τον όρο Reservoir computing εννοούμε ένα πλαίσιο υπολογισμού με τη μορφή νευρωνικού δικτύου.

Είναι μια νέα προσέγγιση στην πρόβλεψη χρονοσειρών με χρήση επαναλαμβανόμενων νευρωνικών δικτύων. Σε ένα Reservoir Computer, ένα σήμα εισόδου τροφοδοτείται σε ένα σταθερό (τυχαίο) δυναμικό σύστημα που ονομάζεται reservoir και το αποτυπώνει σε μια υψηλότερη διάσταση. Στη συνέχεια, ένας απλός μηχανισμός ανάγνωσης έχει εκπαιδευτεί να διαβάζει την κατάσταση του reservoir και να απεικονίζει την επιθυμητή έξοδο. Το κύριο πλεονέκτημα είναι ότι η εκπαίδευση γίνεται μόνο στο στάδιο της ανάγνωσης και ότι το reservoir είναι σταθερό.

5. Εκθέτες Lyapunov

Η χρονική εξέλιξη ενός δυναμικού συστήματος περιγράφεται από τη μορφή της τροχιάς του στο χώρο των φάσεων. Οι εκθέτες Lyapunov αποτελούν ένα **μέτρο του βαθμού της χαοτικής κίνησης μια τροχιάς**. Μπορούμε να πούμε ότι χαρακτηρίζουν το μέσο όρο της εκθετικής σύγκλισης ή απόκλισης γειτονικών τροχιών στο χώρο των φάσεων και για αυτό το λόγο η μελέτη τους μας επιτρέπει να καταλήξουμε σε συμπεράσματα σχετικά με το αν ένα δυναμικό σύστημα είναι χαοτικό ή όχι.

Πιο συγκεκριμένα, **θετικός εκθέτης Lyapunov**, συνεπάγεται και εκθετική απόκλιση δύο γειτονικών τροχιών, άρα και απώλεια προβλεπτικότητας μετά από μικρό χρονικό διάστημα (χαοτική τροχιά). Αντιθέτως, **αρνητικός εκθέτης** σημαίνει συστολή και αναδίπλωση του ελκυστή. Τέλος, **μηδενικό εκθέτη** έχουμε όταν συμβαίνουν διαταραχές κατά μήκος του διανυσματικού πεδίου που εφάπτεται της τροχιάς.

Γενικότερα, ένα ντετερμινιστικό δυναμικό σύστημα είναι χαοτικό όταν έχει τουλάχιστον ένα θετικό εκθέτη Lyapunov. Όταν ένα χαοτικό σύστημα έχει δύο ή περισσότερους θετικούς εκθέτες Lyapunov τότε χαρακτηρίζεται από υπερχάος.

Ένα δυναμικό σύστημα ν διαστάσεων έχει ν εκθέτες Lyapunov, που προσδιορίζουν την εξέλιξη των τροχιών στο χώρο των φάσεων. Όσο μεγαλύτερος είναι ο εκθέτης Lyapunov, τόσο μικρότερη είναι η προβλεπτικότητα σε αυτή τη περιοχή.

6. Φράκταλ

«Ο κόσμος δεν αποτελείται από σημεία, σφαίρες και τετράγωνα». Στην Ευκλείδεια Γεωμετρία όλα τα σχήματα ανάγονται σε σημεία που είναι απλά χωρίς όγκο. Ένα σημείο που κινείται στο χώρο δημιουργεί μια γραμμή, μια γραμμή που κινείται στο χώρο δημιουργεί μια επιφάνεια και μια επιφάνεια κινούμενη στον χώρο δημιουργεί ένα στερεό. Έτσι όλα τα φυσικά αντικείμενα μπορούμε να τα θεωρήσουμε πως κατασκευάζονται από απλά συστατικά που είναι τα σημεία, οι

γραμμές, οι επιφάνειες και τα στερεά. Το σημείο έχει διάσταση μηδέν, η γραμμή έχει διάσταση ένα, η επιφάνεια διάσταση δύο, το στερεό διάσταση τρία. Ο Mandlebrot δημιούργησε την φράκταλ γεωμετρία στην οποία περιγράφονται αντικείμενα με κλασματική διάσταση. Έτσι ένα σύνολο σημείων που δεν γεμίζει επαρκώς μια γραμμή έχει διάσταση μικρότερη από ένα ή ένα σύνολο σημείων που δεν γεμίζει επαρκώς μια επιφάνεια έχει διάσταση μικρότερη από δύο ή μικρότερη από τρία εάν δεν γεμίζει επαρκώς ένα στερεό. Επιπλέον τα φράκταλ αντικείμενα συχνά έχουν το χαρακτηριστικό της αυτοομοιότητος σύμφωνα με την οποία μεγενθύνοντας ένα μέρος του φράκταλ αντικειμένου ανακαλύπτουμε μια πολύπλοκη δομή που ομοιάζει με το αρχικό αντικείμενο. Έτσι, καταλαβαίνουμε ότι τα φράκταλ αντικείμενα δεν αποτελούνται από απλά σημεία αλλά από πολύπλοκες δομές μέσα σε δομές.

1.2 Ιστορική αναδρομή

Η μελέτη του ανθρώπινου εγκεφάλου απασχολεί τους ανθρώπους εδώ και χιλιάδες χρόνια. Με την εξέλιξη της τεχνολογίας όμως, ήταν φυσικό ο άνθρωπος να προσπαθήσει να αξιοποιήσει τη διαδικασία της σκέψης και να πειραματιστεί με μοντέλα εγκαφάλου. Το πρώτο βήμα προς την κατεύθυνση των τεχνητών νευρωνικών δικτύων ήρθε το 1943, όταν ο **Warren McCulloch**, ένας νευροφυσιολόγος, και ένας νεαρός μαθηματικός, ο **Walter Pitts**, έγραψε ένα βιβλίο σχετικά με το πώς οι νευρώνες θα μπορούσαν να λειτουργήσουν. Διαμόρφωσαν έτσι ένα **απλό νευρωνικό δίκτυο με ηλεκτρικά κυκλώματα**.

Ενίσχυση σ' αυτήν την έννοια των νευρώνων και πώς λειτουργούν ήταν ένα βιβλίο που γράφτηκε από τον **Donald Hebb**, "The Organization of Behavior", το 1949. Στο βιβλίο αυτό ο συγγραφέας επεσήμανε ότι **οι νευρικές οδοί ενισχύονται κάθε φορά που χρησιμοποιούνται**.

Καθώς οι υπολογιστές βρίσκονταν σε νηπιακό στάδιο τη δεκαετία του 1950, κατέστη δυνατή η αρχή της διαμόρφωσης των βασικών στοιχείων αυτών των θεωριών σχετικά με την ανθρώπινη σκέψη. Ο **Nathanial Rochester** από τα ερευνητικά εργαστήρια της IBM οδήγησε την πρώτη προσπάθεια για την **προσομοίωση ενός νευρωνικού δικτύου**. Η πρώτη προσπάθεια απέτυχε αλλά αργότερα οι προσπάθειες ήταν επιτυχείς. Ήταν κατά τη διάρκεια αυτής της περιόδου που οι παραδοσιακοί υπολογιστές άρχισαν να ανθίζουν και κατά συνέπεια η έμφαση στην πληροφορική την έρευνα των νευρωνικών δικτύων στο παρασκήνιο.

Ωστόσο, όλο αυτό το διάστημα, οι υποστηρικτές των «σκεπτόμενων μηχανών» συνέχισαν να υποστηρίζουν τις υποθέσεις τους. Το 1956 το Dartmouth Summer Research Project on Artificial Intelligences έδωσε ώθηση τόσο στη τεχνητή νοημοσύνη όσο και στα νευρωνικά δίκτυα. Ένα από τα αποτελέσματα αυτής της διαδικασίας ήταν **να ενθαρρυνθεί η έρευνα τόσο στο κομμάτι της τεχνιτής νοημοσύνης**, όσο και στον εγκέφαλο.

Το 1959, οι Bernard Widrow και Marcian Hoff ανέπτυξαν μοντέλα που ονομάστηκαν **ADALINE** και **MADALINE** (Multiple ADAptive LINear Elements). Το MADALINE ήταν το πρώτο νευρωνικό δίκτυο που εφαρμόστηκε σε πραγματικό πρόβλημα. Είναι ένα φίλτρο το οποίο εξαλείφει την ηχώ στις τηλεφωνικές γραμμές και εξακολουθεί να είναι σε εμπορική χρήση.

Δυστυχώς, αυτές οι επιτυχίες προκάλεσαν τους ανθρώπους να υπερβάλλουν σχετικά με τις δυνατότητες των νευρωνικών δικτύων, Εδώ πρέπει να λάβουμε υπ'όψιν και τον περιορισμό στα ηλεκτρονικά που ήταν διαθέσιμα τότε. Αυτή η υπερβολική δημοσιότητα, από τον ακαδημαϊκό κόσμο και τα εργαστήρια, μόλυνε τη γενική βιβλιογραφία εκείνης της εποχής. Η απογοήτευση πήρε τη θέση των υποσχέσεων που έμειναν απλήρωτες. Επίσης, ένας φόβος επικράτησε που οφειλόταν σε συγγραφείς, οι οποίοι αναρωτήθηκαν τι επίδραση θα είχαν οι "σκεπτόμενες μηχανές» για τον άνθρωπο. Η σειρά του **Asimov** για τα ρομπότ αποκάλυψε τις **επιπτώσεις στα ήθη και τις αξίες του ανθρώπου** όταν οι μηχανές θα ήταν ικανές να κάνουν όλες τις εργασίες που έκανε μέχρι τότε ο άνθρωπος.

Οι φόβοι αυτοί, σε συνδυασμό με τις ανεκπλήρωτες εξωφρενικές απαιτήσεις, προκάλεσαν σεβαστούς επιστήμονες να ασκήσουν κριτική κατά της έρευνας των νευρωνικών δικτύων. Το αποτέλεσμα ήταν να **σταματήσει ένα μεγάλο μέρος της χρηματοδότησης**. Αυτή η περίοδος της καχεκτικής ανάπτυξης διήρκεσε μέχρι το 1981.

Το 1982 πολλά γεγονότα προκάλεσαν ανανεωμένο ενδιαφέρον. **Ο John Hopfield** του Caltech παρουσίασε ένα έγγραφο στην Εθνική Ακαδημία Επιστημών. Η προσέγγιση του Hopfield δεν ήταν να μοντελοποιήσει απλά εγκεφάλους, αλλά **να δημιουργήσει χρήσιμες συσκευές**. Με σαφήνεια και μαθηματική ανάλυση, έδειξε πώς τα εν λόγω δίκτυα θα μπορούσαν να λειτουργήσουν και τι θα μπορούσαν να κάνουν. Ωστόσο, το μεγαλύτερο προσόν του Hopfield ήταν ο χαρακτήρας του. Μπορούσε να μιλήσει σε ακροατήριο, ήταν συμπαθής και κατάφερε να γίνει πρωταθλητής μιας τεχνολογίας, η οποία λίγο έλειψε να πεθάνει.

Την ίδια στιγμή, ένα άλλο γεγονός συνέβη. Ένα συνέδριο πραγματοποιήθηκε στο Κιότο της Ιαπωνίας. Η διάσκεψη αυτή ήταν η κοινή διάσκεψη ΗΠΑ-Ιαπωνίας για τα Συνεταιριστικά/Ανταγωνιστικά Νευρωνικά Δίκτυα. Η **Ιαπωνία** έτσι ανακοίνωσε την προσπάθεια για την κατασκευή **νευρωνικών δικτύων πέμπτης γενιάς**. Στις ΗΠΑ δημιουργήθηκε έτσι μια ανησυχία ότι θα μπορούσαν να μείνουν πίσω στις εξελίξεις. Σύντομα χρηματοδότηση ξεκίνησε και πάλι.

Μέχρι το 1985 το **Αμερικανικό Ινστιτούτο Φυσικής** ξεκίνησε να διοργανώνει μια ετήσια συνάντηση - " Τα Νευρωνικά Δίκτυα για την Πληροφορική". Μέχρι το 1987, το Ινστιτούτο Ηλεκτρολόγων και το πρώτο Διεθνές Συνέδριο Ηλεκτρολόγων Μηχανικών που αφορούσε τα Νευρωνικά Δίκτυα προσέλκυσε πάνω από 1.800 συμμετέχοντες.

Σήμερα, συζητήσεις και συνέδρια σχετικά με τα νευρωνικά δίκτυα συμβαίνουν παντού. Το μέλλον τους φαίνεται πολύ φωτεινό αφού και η ίδια η φύση είναι η απόδειξη ότι αυτή η ιδέα λειτουργεί. Ωστόσο, το μέλλον της, και μάλιστα το κλειδί για το σύνολο της τεχνολογίας, έγκειται στην **ανάπτυξη hardware**. Επί του παρόντος, η περισσότερη ανάπτυξη των νευρικών δικτύων απλώς μας αποδεικνύει ότι το πρότυπο αυτό λειτουργεί. Αυτή η έρευνα δημιουργεί νευρωνικά δίκτυα τα οποία, λόγω των περιορισμών κατά την επεξεργασία, χρειάζονται εβδομάδες για να μάθουν και να λειτουργήσουν αποδοτικά. Για να τα πάρουμε από το εργαστήριο και να τα χρησιμοποιήσουμε χρειαζόμαστε εξειδικευμένα **chips.** Οι εταιρείες εργάζονται σε τριών τύπων chips: ψηφιακά, αναλογικά, και οπτικά. Ορισμένες εταιρείες εργάζονται για τη δημιουργία ενός "compiler πυριτίου" για να δημιουργήσει ένα νευρωνικό δίκτυο: Application Specific Integrated Circuit (ASIC). Αυτά τα κυκλώματα ASIC και τα ψηφιακά chips που μοιάζουν με νευρνώες φαίνεται να είναι το μέλλον της τεχνολογίας. Επίσης τα οπτικά chips φαίνονται πολύ ελπιδοφόρα, νστόσο, μπορεί να περάσουν χρόνια πριν τα οπτικά chips δουν το φως της ημέρας σε εμπορικές εφαρμογές.

1.3 Είδη νευρωνικών δικτύων

Τα τεχνητά νευρωνικά δίκτυα χωρίζονται σε δύο κατηγορίες: τα στατικά και τα δυναμικά.

Τα στατικά νευρωνικά δίκτυα (static) δεν περιέχουν στοιχεία με μνήμη, αλλά μπορούν αν έχουν σαν εισόδους προηγούμενες τιμές των εισόδων. Αυτά τα δίκτυα εκπαιδεύουν κάποιες νευρωνικές δομές ανεξάρτητα από το χρόνο, αποτελούν δηλαδή ένα είδος πάγιας μνήμης.

Τα δυναμικά νευρωνικά δίκτυα (dynamic) περιέχουν στοιχεία με μνήμη, δηλαδή χρονική πληροφορία. Αυτά είναι κατάλληλα για τη μοντελοποίηση μη-γραμμικών δυναμικών συστημάτων και τη σύνθεση δυναμικών ελεγκτών. Στη περίπτωση αυτή έχουμε:

$$u(k) = u(k-1) + \sum_{i=1}^{n+1} w_i x_i(k)$$

Όπου k είναι ο διακριτός χρόνος και απαιτείται η αποθήκευση του u(k-1) που είναι η προηγούμενη τιμή της εισόδου. Τα δυναμικά νευρωνικά δίκτυα διαθέτουν ως συνάρτηση μεταφοράς των τεχνιτών νευρώνων τους μια εξίσωση διαφορών ή ακόμα και μια διαφορική εξίσωση, και διακρίνονται σε:

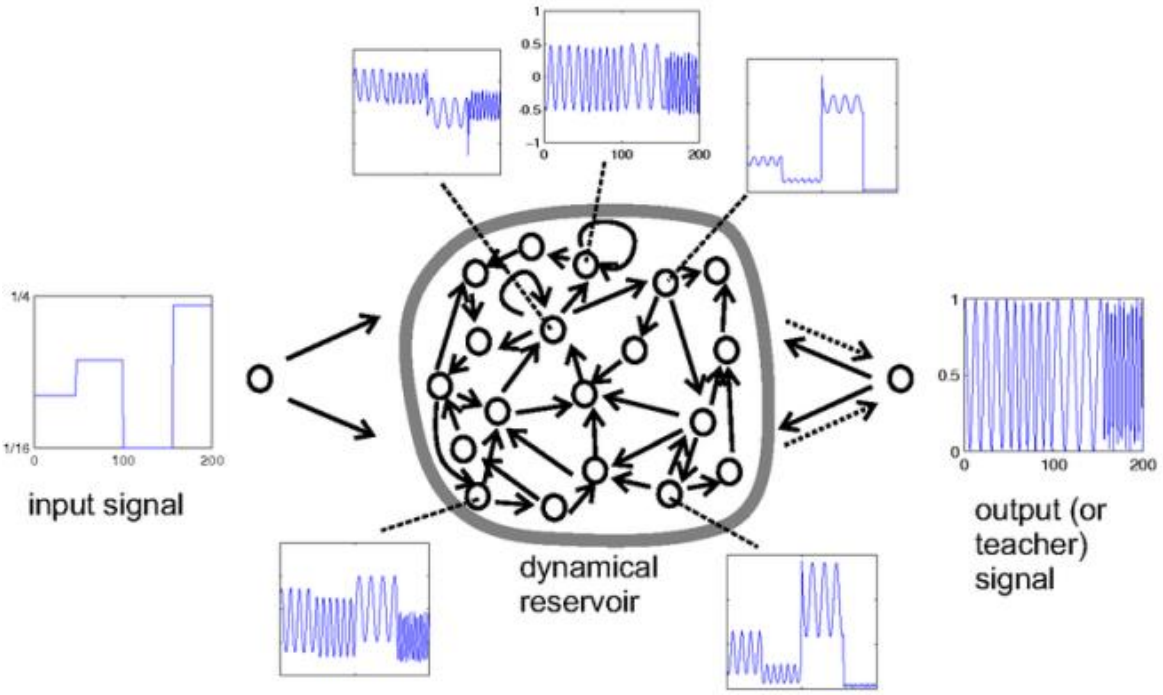
- Forward dynamics, που δεν διαθέτουν ανάδραση δυναμικών
- Output feedback, που διαθέτουν ανατροφοδότηση εξόδου
- State feedback, που διαθέτουν ανατροφοδότηση καταστάσεων

Ας δούμε πιο αναλυτικά κάποια είδη:

1. **ESN (Echo State Network)**

Είναι η απλούστερη μορφή ενός reservoir computer. Σε γενικές γραμμές, το ESN είναι ένα επαναλαμβανόμενο νευρωνικό δίκτυο με ένα αραιά συνεδεδεμένο κρυφό στρώμα (reservoir) το οποίο δεν μπορεί να εκπαιδευτεί και μια απλή γραμμική έξοδο (readout). Τα βάρη σύνδεσης στο reservoir του ESN, καθώς και τα βάρη εισόδου, είναι τυχαία.

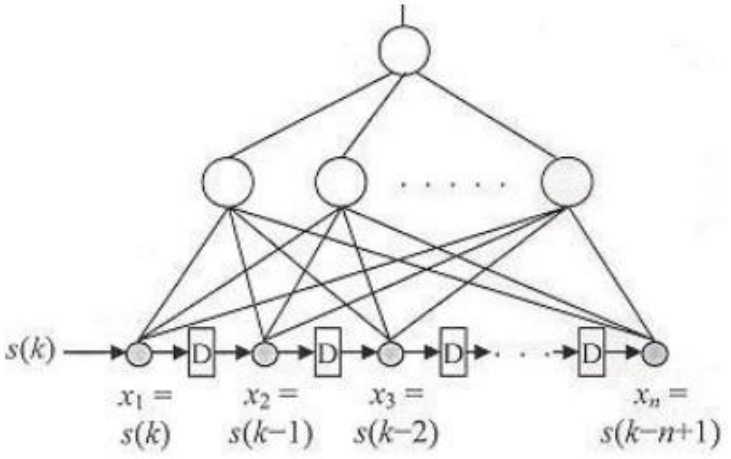
Τα βάρη του reservoir έχουν κλιμακωθεί με τέτοιο τρόπο ώστε να εξασφαλίζεται η ιδιότητα της "ηχού" (Echo state property- ESP): η κατάσταση του reservoir είναι μια "ηχώ" ολόκληρου του ιστορικού της εισόδου. Συνήθως, η φασματική ακτίνα του πίνακα βαρών W του reservoir τίθεται μικρότερη του 1. Τα ESN έχουν εφαρμοστεί επιτυχώς για πρόβλεψης χρονοσειρών, αναγνώριση φωνής, μοντελοποίηση θορύβων, αναγνώριση προτύπων και μοντελοποίηση ξένων γλωσσών .



Σχήμα 1.1: Echo State Network

2. Δίκτυα χρονικής καθυστέρησης (Time Delay neural networks)

Σε αυτά τα δίκτυα υπάρχει μια χρονική καθυστέρηση μεταξύ των εισόδων, δηλαδή η είσοδος x_i κατά τη χρονική στιγμή k ισούται με την είσοδο x_{i+1} κατά τη χρονική στιγμή k+1. Ένα τέτοιο δίκτυο φαίνεται παρακάτω:



Σχήμα 1.2: Δίκτυο χρονικής καθυστέρησης

Το σήμα s(k) εισάγεται από τα αριστερά και προωθείται προς τα δεξιά με διαδοχικές χρονικές καθυστερήσεις (delays) που συμβολίζεται με το γράμμα D. Κάθε είσοδος δηλαδή αναπαράγει το σήμα s με ένα χρόνο καθυστέρησης παραπάνω απ ότι ο αριστερός της γείτονας. Η συνάρτηση που υλοποιείται από το δίκτυο είναι

$$y(k) = F\,[s(k), s(k-1), \ldots, s(k-n+1)]$$

για κάποια μη γραμμική συνάρτηση F. Αυτή η συνάρτηση λέγεται μη γραμμικό φίλτρο, σε αντίθεση με ένα γραμμικό φίλτρο που περιγράφεται από τη συνάρτηση

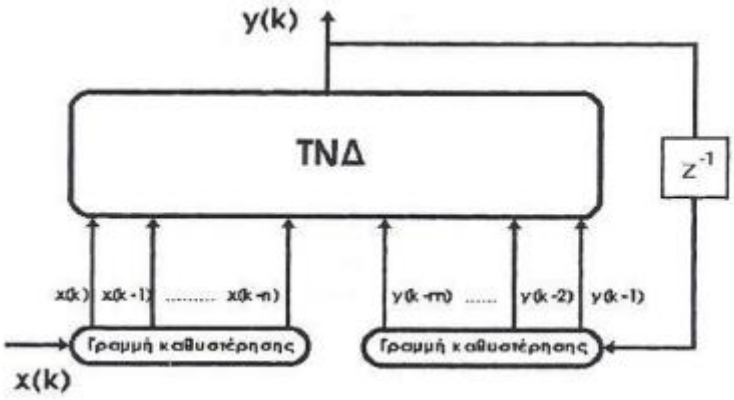
$$y=w_0 s(k) + w_1 s(k-1) + \ldots + w_{n-1} s(k-n+1)$$

Όλη η χρονική πληροφορία βρίσκεται στην είσοδο του δικτύου και επίσης δεν υπάρχει ανατροφοδότηση. Τα δίκτυα χρονικής καθυστέρησης έχουν εφαρμοστεί με επιτυχία σε πολλούς τομείς όπως στον αυτόματο έλεγχο, στη ρομποτική, στην αναγνώριση ανθρώπινης κίνησης καθώς και στην αναγνώριση φωνής.

3. Αναδρομικά δίκτυα (Recurrent networks)

Αναδρομικό είναι ένα δίκτυο στο οποίο υπάρχουν επιπλέον συνδέσεις προς τα πίσω, δηλαδή νευρώνες μεγαλύτερων στρωμάτων τροφοδοτούν είτε νευρώνες προηγουμένων στρωμάτων, είτε νευρώνες του ίδιου στρώματος. Υπάρχει δηλαδή ανατροφοδότηση μεταξύ νευρώνων και αυτό είναι ένας τρόπος εισαγωγής της δυναμικής συμπεριφοράς στο δίκτυο. Τα αναδρομικά δίκτυα είναι ιδιαίτερα κατάλληλα για μοντελοποίηση και έλεγχο δυναμικών συστημάτων.

Ένα γενικευμένο παράδειγμα ανατροφοδοτούμενου τεχνητού νευρωνικού δικτύου φαίνεται παρακάτω:



Σχήμα 1.3: γενικευμένο παράδειγμα ανατροφοδοτούμενου δικτύου

Η έξοδος του ανατροφοδοτούμενου δικτύου σε κάθε χρονική στιγμή είναι συνάρτηση των προηγουμένων χρονικά εξόδων και εισόδων και αυτό το δίκτυο περιγράφεται από τη σχέση:

$$y(k) = F\,[x(k), x(k-1), \ldots, x(k-n), y(k-1), y(k-2), \ldots, y(k-m)]$$

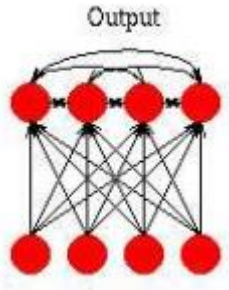
Η τιμή ενός νευρώνα δεν εξαρτάται μόνο από τις τιμές των εισόδων, αλλά και από τις τιμές των άλλων νευρώνων ή και του εαυτού του. Το σύστημα έχει μνήμη γιατί η απόκρισή του εξαρτάται όχι μόνο από την είσοδο αλλά και από τις αρχικές τιμές των νευρώνων. Για τον υπολογισμό της τιμής εξόδου y_i ενός νευρώνα i τη χρονική στιγμή k χρησιμοποιείται ο παρακάτω αναδρομικός τύπος που περιέχει τις τιμές των νευρώνων κατά την προηγούμενη χρονική στιγμή:

$$y_i(k) = f(\sum_j w_{ij} y_j(k-1))$$

όπου έχουν δοθεί κάποιες αρχικές τιμές $y_i = y_i(0)$ σε όλους τους νευρώνες του δικτύου. Το δίκτυο μαθαίνει να ακολουθεί κάποια τροχιά στο χρόνο, η οποία εξαρτάται από τις τιμές των συναπτικών βαρών και τις αρχικές τιμές $y_i(0)$

4. Ανταγωνιστικά νευρωνικά δίκτυα

Ο τύπος αυτός τεχνητού νευρωνικού δικτύου χρησιμοποιείται στην περίπτωση μη επιβλεπόμενης εκμάθησης. Η αναγκαιότητά του εμφανίζεται στις περιπτώσεις που απαιτείται να αναλύσουμε ακατέργαστα δεδομένα για τα οποία δε διαθέτουμε προηγούμενη γνώση για τυχούσες κατηγοριοποιήσεις τους. Ο μόνος δυνατός τρόπος είναι ο προσδιορισμός ειδικών χαρακτηριστικών και η οργάνωση των δειγμάτων σε ομάδες με βάση την ομοιότητά τους ως προς αυτά. Στο παρακάτω σχήμα παρατίθεται το διάγραμμα ενός τέτοιου δικτύου.



Σχήμα 1.4: Δομή ενός απλού ανταγωνιστικού δικτύου

Όπως φαίνεται, αποτελείται από δύο υπομονάδες:

a)Ένα δίκτυο Hemming που υπολογίζει το βαθμό ομοιότητας του διανύσματος εισόδου με αυτό των βαρών που αντιστοιχούν σε κάποιο από τα νευρώνια εξόδου.

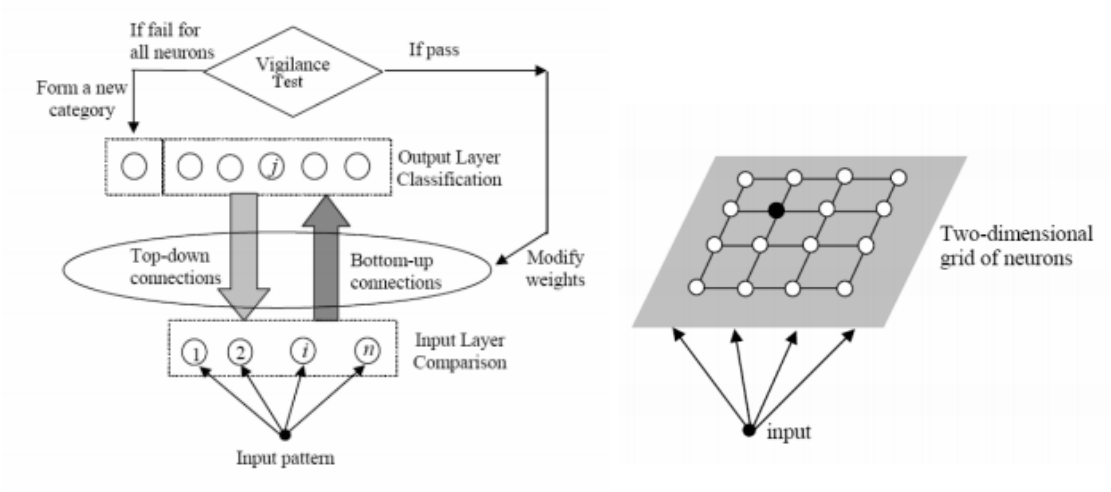
b)Ένα δίκτυο Maxnet που προσδιορίζει τον κόμβο εξόδου που αντιστοιχεί στη μεγαλύτερη τιμή.

Σχήμα 1.5: Δίκτυο Hemming και Maxnet

Όποτε παρουσιάζεται ένα δείγμα στην είσοδο του δικτύου, η υπομονάδα τύπου Hemming υπολογίζει την απόσταση για καθένα από τους k κόμβους εξόδου της υπομονάδας. Κατόπιν, το δίκτυο Maxnet αναλαμβάνει τον προσδιορισμό του επικρατώντα νευρώνα. Για τον κόμβο-νικητή, τα βάρη διασύνδεσης του δικτύου Hemming μεταβάλλονται σύμφωνα με τη σχέση:

$$\vec{w}_{new} = \vec{w}_{old} + \eta\left(\vec{x} - \vec{w}_{old}\right)$$

με τα υπόλοιπα να παραμένουν ίδια. Με αυτό τον τρόπο, τα διανύσματα w προσεγγίζουν το κέντρο βάρους της άγνωστης ομάδας. Η διαδικασία επαναλαμβάνεται για όλα τα δείγματα εκπαίδευσης μέχρι να σταθεροποιηθούν οι τιμές των συναπτικών συνδέσεων. Όταν η διαδικασία ολοκληρωθεί με επιτυχία, κάθε νέα είσοδος θα κατατάσσεται στην κοντινότερη ομάδα. Στην κατηγορία της μη επιβλεπόμενης εκμάθησης ανήκουν επίσης τα δίκτυα Kohonen (ή δίκτυα αυτοοργανούμενου χάρτη γνωρισμάτων-Self Organizing Feature Map,SOFM) και τα δίκτυα ART (θεωρία προσαρμοστικού συντονισμού- Adaptive Resonance Theory).



Σχήμα 1.6: Δίκτυο Art και Kohonen

1.4 Κανόνες εκμάθησης

Με τον όρο εκμάθηση, αναφερόμαστε σε μια διαδικασία με την οποία οι ελεύθερες παράμετροι του δικτύου προσαρμόζονται με τη βοήθεια διεγέρσεων από το περιβάλλον. Ο τύπος της εκμάθησης εξαρτάται από τον τρόπο με τον οποίο γίνονται οι μεταβολές στις παραμέτρους.

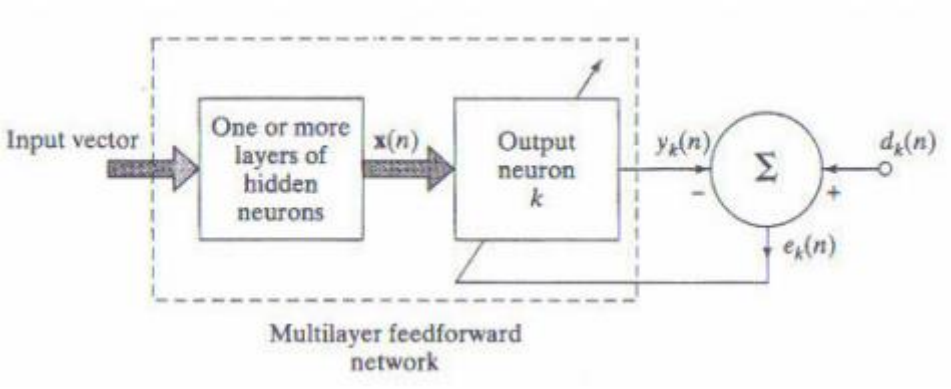
Η διαδικασία που ακολουθείται συνοψίζεται ως εξής:

- το δίκτυο διεγείρεται από το περιβάλλον
- υποβάλλεται σε μεταβολές των ελεύθερων παραμέτρων ως αποτέλεσμα της διέγερσης
- αποκρίνεται με νέο τρόπο στο περιβάλλον εξαιτίας των αλλαγών που έχουν συμβεί στην εσωτερική τους δομή.

Τα βήματα για την επίλυση ενός προβλήματος εκμάθησης ονομάζεται αλγόριθμος εκμάθησης. Όπως είναι αναμενόμενο, δεν υπάρχει μοναδικός τρόπος για την επιλογή του κατά τη σχεδίαση ενός νευρωνικού δικτύου, αφού καθένας διαθέτει τα δικά του πλεονεκτήματα. Το σημείο στο οποίο κατά κύριο λόγο διαφέρουν είναι η διαδικασία που τροποποιούνται τα συναπτικά βάρη και τα επίπεδα πόλωσης. Με βάση την τελευταία παρατήρηση, μπορούμε να τους κατατάξουμε σε πέντε βασικές κατηγορίες:

i. **Διόρθωσης λάθους (error-correction learning)**

Η μέθοδος ονομάζεται κανόνας δέλτα ή Widrow-Hoff , προς τιμή των θεμελιωτών του, και έχει τις ρίζες της στην τεχνική του βέλτιστου φιλτραρίσματος (optimum filtering). Για ευκολία στην παρουσίαση, υιοθετούμε τη δομή του παρακάτω σχήματος, με ένα νευρώνα στο στρώμα εξόδου και τυχαίο αριθμό κρυφών στρωμάτων.



Σχήμα 1.7: Γραφική αναπαράσταση του κανόνα διόρθωσης λάθους

Έστω $x_r(n)$ η διέγερση στην είσοδο του νευρωνίου εξόδου τη χρονική στιγμή r, που προκαλεί την απόκριση y_k (n). Η τιμή αυτή συγκρίνεται με την προσδοκόμενη απόκριση d_k (n) και παράγει το σήμα σφάλματος

$$e_k(n) = y_k(n) - d_k(n)$$

Το σύστημα αναλαμβάνει την ρύθμιση των συναπτικών βαρών w_{kj} (n) , ώστε να ελαχιστοποιηθεί το σφάλμα

$$E(n) = \frac{1}{2} e_k^2(n)$$

Κατά τη σχέση:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \text{ με } \Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

όπου η ο ρυθμός εκμάθησης και x_j (n) η j-οστή συνιστώσα του x(n). Ιδιαίτερη σημασία έχει η προσεκτική επιλογή της παραμέτρου η ώστε να εξασφαλίζεται η σύγκλιση της επαναληπτικής διαδικασίας. Γενικά, πρέπει να σημειωθεί ο σπουδαίος ρόλος που διαδραματίζει στην απόδοση του αλγορίθμου.

ii. Μνημονική εκμάθηση (memory-based learning)

Ο αλγόριθμος βασίζεται στην αποθήκευση όλων (ή ενός σημαντικού μέρους) των εκπαιδευτικών δειγμάτων σε μια μεγάλη μνήμη μαζί με την πραγματική τους απόκριση, στη μορφή

$$\{\vec{x}_i, d_i\}_{i=1}^{N}$$

και περιλαμβάνει δύο απαραίτητα συστατικά:

a) ένα κριτήριο που καθορίζει την τοπική γειτονιά (local neighbourhood) του διανύσματος δοκιμής x_{test}

b) Ένα κανόνα εκμάθησης που εφαρμόζεται στα δείγματα εκπαίδευσης της τοπικής γειτονιάς του x_{test}

Διαφορετικές εκδόσεις αυτής της μεθόδου οφείλονται σε διαφορετικούς ορισμούς των δύο παραπάνω τμημάτων.

Ο τρόπος καθορισμού της γειτονιάς του x_{test} βασίζεται στον προσδιορισμό των k κοντινότερων προς αυτό προτύπων, μέσω του κριτηρίου ελάχιστης (ευκλείδειας) απόστασης. Με τον τρόπο αυτό αποφεύγεται η επίδραση των outliers στην απόφασή μας, όπως φαίνεται και στο παρακάτω σχήμα.

Σχήμα 1.8. Επίδειξη της ανεξαρτησίας του αλγόριθμου στην ύπαρξη outlier

iii. Hebbian learning

Η διατύπωσή του στην περίπτωση των τεχνητών νευρωνικών δικτύων είναι η παρακάτω :

a) Αν δύο νευρώνες εκατέρωθεν μιας σύναψης ενεργοποιούνται ταυτόχρονα, η ισχύς της επιλεκτικά ενισχύεται.

b) Αν αυτό γίνεται ασύγχρονα, η σύναψη αποδυναμώνεται.

Η σύναψη αυτή καλείται Χεβιανή (Hebbian). Πρόκειται για ένα:

- χρονικά εξαρτώμενο(time-dependent) –οι μεταβολές στη σύναψη εξαρτώνται από το χρόνο εκδήλωσης προσυναπτικών και μετασυναπτικών σημάτων.
- υψηλά τοπικό (highly local) με τη χωροχρονική έννοια- τοπική εγγύτητα νευρώνων και χρονική εγγύτητα σημάτων.
- ισχυρά διαδραστικό (interactive) –οι μεταβολές εξαρτώνται από τα σήματα εκατέρωθεν της σύναψης.
- συζυγικό ή συσχετιστικό (conjunctional or correlational) –η συνθήκη για τη μεταβολή της συναπτικής δραστικότητας είναι η συνύπαρξη προσυναπτικών και μετασυναπτικών σημάτων

μηχανισμό που στοχεύει στην ενίσχυση της συναπτικής αποδοτικότητας ως συνάρτησης της συσχέτισης των προσυναπτικών και μετασυναπτικών δράσεων.

Μπορούμε να γενικεύσουμε τη ιδέα της τροποποίησης κατά τον κανόνα του Hebb, θεωρώντας πως θετικά συσχετιζόμενα σήματα προκαλούν ενίσχυση της σύναψης και ότι καθόλου ή αρνητικά συσχετιζόμενα σήματα την οδηγούν σε αποδυνάμωση.

Η μαθηματική διατύπωση του κανόνα στη γενική του μορφή είναι :

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n))$$

με x_j (n) το προσυναπτικό και y_k (n) το μετασυναπτικό σήμα στο χρονικό βήμα n.

Μια συνήθης μορφή της F είναι η

$$\Delta w_{kj}(n) = \eta(x_j - x)(y_k - y)$$

όπου η είναι ο ρυθμός εκπαίδευσης , x και y οι χρονικές μέσες τιμές (εντός ενός μικρού χρονικού διαστήματος) των x_j (n) και y_k (n) αντίστοιχα.

Οι τελευταίες ποσότητες επηρεάζουν το πρόσημο της τροποποίησης Δ. Η πρόταση αυτή ονομάζεται υπόθεση συμμεταβλητότητας και επιτρέπει τα παρακάτω:

a) σύγκλιση στη μηδενική κατάσταση, όταν $x_j = x$ ή $y_k = y$

b) Πρόβλεψη τόσο της συναπτικής ενίσχυσης όσο και συναπτικής αποδυνάμωσης.

Για την συνάρτηση που χρησιμοποιήθηκε, μπορούμε να κάνουμε τις παρακάτω παρατηρήσεις:

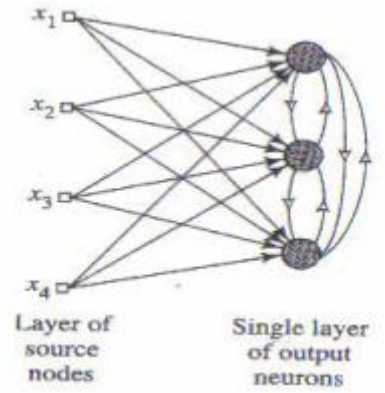
1. τα βάρη w_{kj} ισχυροποιούνται αν $x_j > x$ και $y_k > y$
2. αντίθετα υποβαθμίζονται αν ισχύουν ταυτόχρονα $x_j > x$ και $y_k < y$, ή αντίστροφα.

iv. Ανταγωνιστική εκμάθηση (competitive learning)

Στην περίπτωση αυτή, τα νευρώνια εξόδου ανταγωνίζονται μεταξύ τους ώστε να καθοριστεί πιο θα ενεργοποιηθεί. Κάθε χρονική στιγμή μόνο ένα μπορεί βρίσκεται σ'αυτή την κατάσταση. Αυτό το γνώρισμα καθιστά την ανταγωνιστική εκμάθηση εξαιρετικά κατάλληλη για την ανακάλυψη στατιστικά σημαντικών γνωρισμάτων που μπορεί να χρησιμοποιηθούν για την ταξινόμηση ενός συνόλου δειγμάτων.

Οι αναδράσεις δρουν ως πλευρικοί αναστολείς, με κάθε νευρώνιο να επιχειρεί την καταστολή της ενεργοποίησης των άλλων με τα οποία συνδέεται πλευρικά. Σε αντίθεση, οι συνδέσεις εμπροσθοδιάδοσης είναι όλες διεγερτικές (excitatory). Επίσης, για τις τελευταίες, μπορούμε να εισάγουμε κάποιο περιορισμό στην τιμή τους, π.χ.

$$\sum_j w_{kj}^2 = 1 \quad \text{για κάθε k.}$$



Σχήμα 1.9: Δίκτυο ανταγωνιστικής εκμάθησης

Ο καθορισμός του νικητή-νευρωνίου γίνεται κατά τη σχέση:

$$y_k = \begin{cases} 1 & \text{αν } v_k > v_j \text{ για καθε } j, j \neq k \\ 0 & \text{διαφορετικα} \end{cases}$$

όπου $υ_κ$ αναπαριστά τη συνδυασμένη δράση των εμπρόσθιων και ανατροφοδοτούμενων εισόδων του νευρωνίου k. Η μεταβολή των συναπτικών βαρών εμπροσθοδιάδοσης wkj γίνεται κατά τη σχέση:

$$\Delta w_{kj} = \begin{cases} \eta\left(x_j - w_{kj}\right) & \text{αν το νευρωνιο k ειναι ο νικητης} \\ 0 & \text{διαφορετικα} \end{cases}$$

με η ο ρυθμός εκμάθησης. Με τον τρόπο αυτό το διάνυσμα βαρών $w_κ$ του νικητήριου νευρωνίου μετακινείται κάθε φορά προς το πρότυπο εισόδου x.

v. Boltzmann learning

Ο κανόνας βασίζεται σε συμπεράσματα της στατιστικής μηχανικής. Το νευρωνικό δίκτυο που στηρίζεται σ'αυτού του είδους την εκμάθησης λέγεται μηχανή Boltzmann. Δομικά, πρόκειται για ένα επαναληπτικό δίκτυο του οποίου τα νευρώνια συμπεριφέρονται δυαδικά, δηλαδή μπορούν να βρίσκονται σε κατάσταση "on"(x_j = 1) ή "off"(x_j = -1). Περιλαμβάνει τα φανερά νευρώνια (αυτά που συνιστούν τη διασύνδεση με το περιβάλλον) και τα κρυφά που λειτουργούν ελεύθερα. Χαρακτηρίζεται από μια συνάρτηση ενέργειας της μορφής

$$E = -\frac{1}{2}\sum_j \sum_{k \atop j \neq k} w_{kj} x_k x_j$$

με w_{kj} το βάρος σύνδεσης μεταξύ των νευρωνίων x_j και x_k . Η εκπαίδευσή του στηρίζεται στην τυχαία επιλογή σε κάποιο βήμα ενός νευρωνίου και την αλλαγή της κατάστασής του ($x_j \rightarrow - x_j$). Αυτή η διαδικασία επαναλαμβάνεται μέχρι να φτάσουμε σε «θερμική ισορροπία».

Διακρίνουμε 2 καταστάσεις λειτουργίας:

- ✓ Επιβαλλόμενη (clamped) συνθήκη στην οποία τα φανερά νευρώνια τίθενται σε συγκεκριμένες καταστάσεις που καθορίζονται από το περιβάλλον.
- ✓ Ελεύθερη εκτέλεση (free-running) , κατά την οποία όλα ανεξαιρέτως τα νευρώνια επιτρέπεται να λειτουργούν ελεύθερα. 25

Θεωρώντας με $ρ_{kj}$ την κατά μέσο όρο συσχέτιση για όλες τις δυνατές καταστάσεις των νευρωνίων k και j στη θερμική ισορροπία και συμβολίζοντας τη με $ρ^+_{kj}$, $ρ^-_{kj}$ για τις δύο προαναφερθείσες καταστάσεις λειτουργίας, η μεταβολή των συναπτικών συνδέσεων στη μηχανή Boltzmann καθορίζεται από τη σχέση

$$\Delta w_{kj} = \eta\left(\rho^+_{kj} - \rho^-_{kj}\right), \quad j \neq k$$

με η το ρυθμό εκμάθησης.

1.5 Πλεονεκτήματα τεχνιτών νευρωνικών δικτύων

Τα νευρωνικά δίκτυα οφείλουν την υπολογιστική τους ισχύ πρώτον στην **παράλληλη, κατανεμημένη δομή** τους και δεύτερον στην **ικανότητά τους να μαθαίνουν και να γενικεύουν**. Αυτές οι δύο δυνατότητες επιτρέπουν στα νευρωνικά δίκτυα να βρίσκουν καλές προσεγγιστικές λύσεις σε πολύπλοκα προβλήματα. Ωστόσο δεν είναι μόνο αυτά τα πλεονεκτήματά τους. Έχουν και τις παρακάτω χρήσιμες ιδιότητες:

- **Μη γραμμικότητα**

Η ιδιότητα του νευρώνα να έχει τη δυνατότητα να μην είναι γραμμικός είναι πολύ σημαντική, κυρίως αν ο μηχανισμός που παράγει το σήμα εισόδου (πχ ομιλία) είναι εκ φύσεως μη γραμμικός.

- **Παράλληλος τρόπος λειτουργίας**

Τα νευρωνικά δίκτυα λειτουργούν με παράλληλο τρόπο, γιατί μια εργασία τους μοιράζεται στα διάφορα τμήματα του δικτύου, δηλαδή σε όλους τους επιμέρους νευρώνες, κάτι που παρέχει μεγάλες ταχύτητες, γιατί είναι σαν να έχουμε ταυτόχρονα πολλούς επεξεργαστές.

- **Προσαρμοστικότητα**

Η ικανότητά τους να προσαρμόζουν τα συναπτικά τους βάρη ανάλογα με τις μεταβολές του περιβάλλοντος κάνει τα νευρωνικά δίκτυα ευπροσάρμοστα σε νέα περιβάλλοντα.

- **Ευχρηστία**

Τα νευρωνικά δίκτυα εκπαιδεύονται με παραδείγματα. Ο χρήστης συγκεντρώνει αντιπροσωπευτικά δεδομένα και στη συνέχεια, καθώς τα τροφοδοτεί συστηματικά στο δίκτυο μέσω των κατάλληλων αλγορίθμων εκπαίδευσης, το δίκτυο «αντιλαμβάνεται» αυτομάτως τη δομή των δεδομένων και η «γνώση» αυτή εκφράζεται ως κατάλληλες επιλογές συναπτικών βαρών. Επομένως το τελικό αποτέλεσμα της εκπαίδευσης με ένα συγκεκριμένο σύνολο παραδειγμάτων είναι ο προσδιορισμός των κατάλληλων βαρών του δικτύου. Ο χρήστης χρειάζεται να έχει κάποιες ουσιώδεις γνώσεις σχετικά με τον τρόπο επιλογής και προετοιμασίας των δεδομένων, τον τρόπο εκλογής του κατάλληλου νευρωνικού δικτύου και στο πως θα ερμηνευτούν τα αποτελέσματα.

- **Αναλογία με τη φυσιολογία του εγκεφάλου**

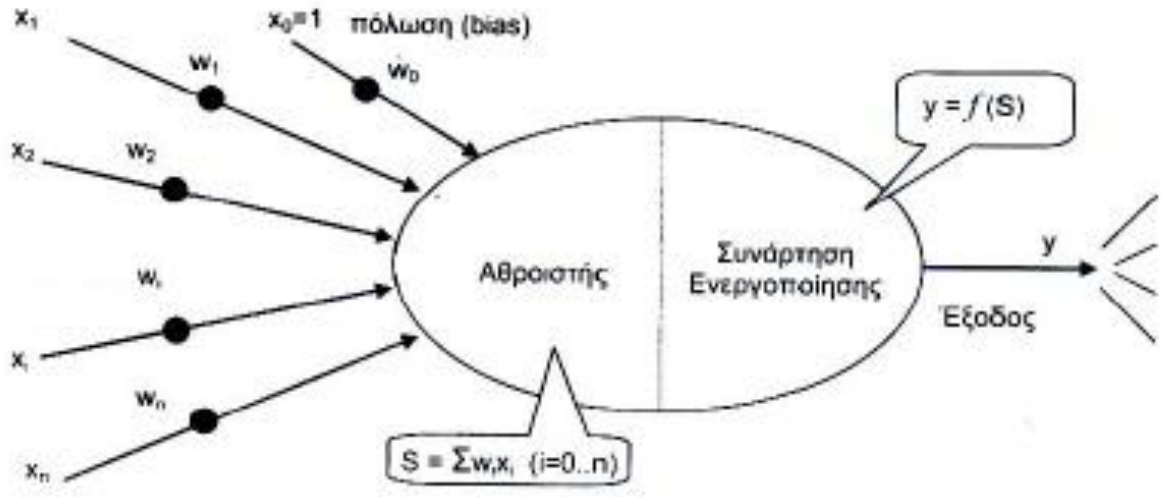
Η σχεδίαση ενός νευρωνικού δικτύου δανείζεται στοιχεία από τη λειτουργία του ανθρώπινου εγκεφάλου. Τα τεχνητά, όπως και τα βιολογικά νευρωνικά δίκτυα, έχουν μεγάλη ανοχή σε δομικά σφάλματα. Αυτό σημαίνει ότι η κακή λειτουργία ή η καταστροφή ενός νευρώνα ή κάποιων συνδέσεων δεν είναι ικανή να διαταράξει σημαντικά τη λειτουργία τους, καθώς η πληροφορία που εσωκλείουν δεν είναι εντοπισμένη σε συγκεκριμένο σημείο αλλά διάχυτη σε όλο το δίκτυο.

- **Ικανότητα για αναγνώριση προτύπων**

Τα τεχνητά νευρωνικά δίκτυα έχουν εξαιρετική ικανότητα αναγνώρισης προτύπων καθώς δεν επηρεάζονται από ελλειπή ή και με θόρυβο δεδομένα. Από τη στιγμή που ένα τεχνητό νευρωνικό δίκτυο εκπαιδεύεται στο να αναγνωρίζει συνθήκες και καταστάσεις, απαιτείται μόνο ένας κύκλος λειτουργίας τους για να προσδιορίσουν μια συγκεκριμένη κατάσταση. Η τελευταία ιδιότητα είναι αυτή που τα κάνει ιδανικά για χρήση σε αυτοματισμούς που θα λειτουργήσουν σε αντίξοες συνθήκες, όπως για παράδειγμα σε διαστημικές αποστολές, σε χώρους με ραδιενέργεια και σε πεδία μάχης.

1.6 Το μοντέλο του τεχνητού νευρώνα

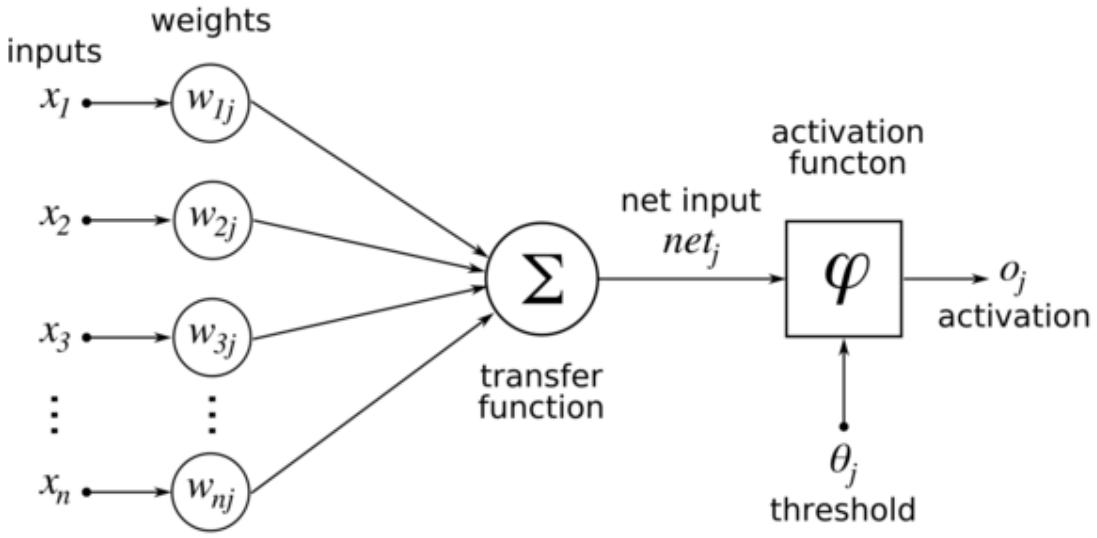
Ο τεχνητός νευρώνας είναι ένα υπολογιστικό μοντέλο, τα μέρη του οποίου μπορούν να αντιστοιχιστούν άμεσα με αυτά του βιολογικού νευρώνα. Ένας τεχνητός νευρώνας δέχεται κάποια σήματα εισόδου $x_1, x_2, \ldots, x_n$ και κάθε τέτοιο σήμα μεταβάλλεται από μια τιμή βάρους w_i (weight), ο ρόλος της οποίας είναι αντίστοιχος του ρόλου της σύναψης στο βιολογικό νευρώνα. Η τιμή του βάρους μπορεί να είναι θετική ή αρνητική σε αντιστοιχία με την επιταχυντική ή επιβραδυντική λειτουργία της σύναψης. Το παρακάτω σχήμα παρουσιάζει το μοντέλο του τεχνητού νευρώνα:



Σχήμα 1.10: Το μοντέλο του τεχνητού νευρώνα

Το σώμα του νευρώνα χωρίζεται σε δύο μέρη, τον αθροιστή (sum), ο οποίος προσθέτει τα επηρεασμένα από τα βάρη σήματα εισόδου παράγοντας τη ποσότητα S και τη συνάρτηση ενεργοποίησης (activation function), ένα είδος φίλτρου, το οποίο διαμορφώνει τη τελική τιμή του σήματος εξόδου y, σε συνάρτηση με τη ποσότητα S και τη τιμή κατωφλίου της συνάρτησης ενεργοποίησης.

Η εναλλακτική μορφή ενός μοντέλου τεχνητού νευρώνα φαίνεται στο παρακάτω σχήμα και αποτελείται από τα εξής στοιχεία:



Σχήμα 1.11: Εναλλακτική μορφή τεχνητού νευρώνα

- Ένα σύνολο συνάψεων, κάθε μία από τις οποίες χαρακτηρίζεται από το δικό της βάρος w_{ki}. Ο δείκτης k αναφέρεται στο νευρώνα, ενώ ο δείκτης i στο άκρο εισόδου σύναψης. Τα συναπτικά βάρη είναι θετικοί πραγματικοί αριθμοί για τις ενισχυτικές συνάψεις και αρνητικοί για τις ανασταλτικές.

- Έναν αθροιστή για την άθροιση των σημάτων εισόδου, πολλαπλασιασμένων με τα αντίστοιχα συναπτικά βάρη του νευρώνα.

- Μια συνάρτηση ενεργοποίησης, η οποία είναι μη γραμμικός μετασχηματιστής για τον περιορισμό του πλάτους της εξόδου του νευρώνα σε κάποια πεπερασμένη τιμή. Συνήθως το κανονικοποιημένο εύρος τιμών πλάτους της εξόδου ενός νευρώνα είναι το μοναδιαίο κλειστό διάστημα [0,1] ή [-1,1].

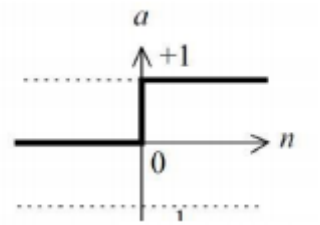
Το μοντέλο του νευρώνα περιλαμβάνει επίσης μια εξωτερικά εφαρμοζόμενη **πόλωση**, την b_i. Η πόλωση έχει ως αποτέλεσμα την **αύξηση ή μείωση της δικτυακής διέγερσης της συνάρτησης ενεργοποίησης**, ανάλογα με το αν είναι θετική ή αρνητική αντίστοιχα. Η πόλωση είναι μια εξωτερική παράμετρος του νευρώνα και είναι ίση με το συναπτικό βάρος w_{k0} της σταθερής εισόδου x_0 =+1. Οι μαθηματικές εξισώσεις που περιγράφουν τον νευρώνα είναι οι εξής:

$$u_k = \sum_{j=0}^{m} wkjx_j \qquad y_k = \varphi(u_k)$$

Η συνάρτηση ενεργοποίησης η οποία συμβολίζεται με $\phi(u_k)$ ορίζει την έξοδο ενός νευρώνα βάσει του τοπικού πεδίου u. Υπάρχουν πολλά είδη συναρτήσεων και οι πιο βασικές είναι οι εξής:

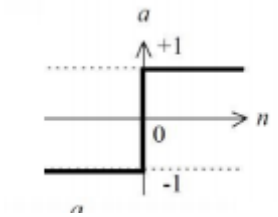
- **Βηματική συνάρτηση 0/1** (step function)

$$\alpha(n)=\begin{cases} 0 & \text{αν } n \geq 0 \\ 1 & \text{αν } n < 0 \end{cases}$$



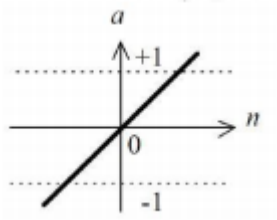
- **Βηματική συνάρτηση -1/1** (συμβολίζεται και σαν sgn)

$$\alpha(n)=\begin{cases} 1 & \text{αν } n \geq 0 \\ -1 & \text{αν } n < 0 \end{cases}$$



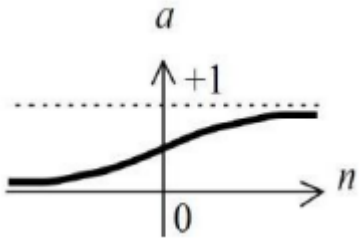
- **Γραμμική συνάρτηση α(n)= b_n για κάθε n,**

όπου b η κλίση της ευθείας.



- **Σιγμοειδής συνάρτηση** (sigmoid):

Αποτελεί την πιο κοινή μορφή συνάρτησης ενεργοποίησης που χρησιμοποιείται στην κατασκευή νευρωνικών δικτύων. ορίζεται ως αυστηρά αύξουσα συνάρτηση και λαμβάνει τιμές από ένα συνεχές πεδίο τιμών από 0 ως 1 και είναι διαφορίσιμη.



1.7 Εφαρμογές

Ενδεικτικά αντιπροσωπευτικά παραδείγματα προβλημάτων στα οποία η ανάλυση των νευρωνικών δικτύων έχει εφαρμοστεί με επιτυχία είναι τα εξής:

- **Ιατρική διάγνωση:** Τα Νευρωνικά Δίκτυα χρησιμοποιούνται πειραματικά για τη **μοντελοποίηση του ανθρώπινου καρδιαγγειακού συστήματος**. Η διάγνωση μπορεί να επιτευχθεί με την κατασκευή ενός μοντέλου του καρδιαγγειακού συστήματος ενός ατόμου το οποίο θα συγκρίνεται με τις φυσιολογικές μετρήσεις που λαμβάνονται σε πραγματικό χρόνο από τον ασθενή. Εάν αυτή η ρουτίνα πραγματοποιείται τακτικά, πιθανές βλαβερές ιατρικές παθήσεις μπορεί να ανιχνευθούν σε πρώιμο στάδιο και έτσι η διαδικασία της καταπολέμησης της ασθένειας θα γίνει πολύ ευκολότερη.

  Ένα μοντέλο του καρδιαγγειακού συστήματος ενός ατόμου πρέπει να μιμείται τη σχέση μεταξύ των φυσιολογικών μεταβλητών (δηλαδή, του καρδιακού ρυθμού, τη συστολική και διαστολική αρτηριακή πίεση, και τον ρυθμό αναπνοής) σε διαφορετικά επίπεδα σωματικής δραστηριότητας. Αν ένα μοντέλο είναι προσαρμοσμένο σε ένα άτομο, τότε **δημιουργείται ένα πρότυπο για τη φυσική κατάσταση** του εν λόγω ατόμου. Ο προσομοιωτής θα πρέπει να είναι σε θέση να προσαρμόζεται στα χαρακτηριστικά οποιουδήποτε ατόμου χωρίς την επίβλεψη ενός εμπειρογνώμονα. Γι αυτό το λόγο είναι απαραίτητη η χρήση ενός νευρωνικού δικτύου.

  Ένας άλλος λόγος που δικαιολογεί τη χρήση της τεχνολογίας τεχνιτών νευρωνικών δικτύων, είναι η ικανότητα τους για συγχώνευση όλων των αισθητήρων, δηλαδή η ικανότητα να **συνδυάζουν τιμές που δέχεται από διαφορετικούς αισθητήρες**. Αυτό το γεγονός επιτρέπει στα τεχνητά νευρωνικά δίκτυα να μαθαίνουν πολύπλοκες σχέσεις μεταξύ των επιμέρους τιμών των αισθητήρων, οι οποίες διαφορετικά θα χανόταν εάν οι τιμές αναλύονταν μεμονωμένα. Στην ιατρική διάγνωση και μοντελοποίηση, αυτό σημαίνει ότι ακόμα και αν κάθε αισθητήρας σε ένα σύνολο μπορεί να είναι ευαίσθητος μόνο σε μια συγκεκριμένη φυσιολογική μεταβλητή, τα νευρωνικά δίκτυα είναι ικανά να ανιχνεύουν πολύπλοκες ιατρικές καταστάσεις με συγχώνευση των δεδομένων από τα μεμονωμένους βιοϊατρικούς αισθητήρες.

  Το μοντέλο αυτό θα μπορούσε να χρησιμοποιηθεί για την **ιατρική παρακολούθηση των εργαζομένων σε επικίνδυνα περιβάλλοντα**, όπως για παράδειγμα των πυροσβεστών. Θα μπορούσε να καθοριστεί αν οι πυροσβέστες έχουν αναρρώσει επαρκώς από τις τελευταίες εισπνοές καπνού, για να τους επιτραπεί να εισέλθουν και πάλι σε περιβάλλον γεμάτο καπνό.

  Τα πλεονεκτήματα που μπορεί να προσφέρει ένα τέτοιο σύστημα είναι προφανή. Οι άνθρωποι μπορούν να ελεγχθούν για καρδιακές παθήσεις

γρήγορα και ανώδυνα και συνεπώς μπορεί να γίνει ανίχνευση οποιασδήποτε ασθένειας σε πρώιμο στάδιο.

Μία άλλη πολύ ενδιαφέρουσα εφαρμογή τους, είναι στη **πρόβλεψη επιληπτικών κρίσεων**. Ας τη δούμε πιο αναλυτικά:

Περίπου το 1% του παγκόσμιου πληθυσμού έχει διαγνωστεί με επιληψία. Η επιληψία είναι μια διαταραχή του νευρικού συστήματος που χαρακτηρίζεται από επεισοδιακή απώλεια της προσοχής ή υπνηλία και στις περισσότερες περιπτώσεις απώλεια συνείδησης. Η επιληψία προκαλείται από μη φυσιολογική ηλεκτρική δραστηριότητα στον εγκέφαλο, πιο συγκεκριμένα από μία συγχρονισμένη εκκένωση των νευρώνων της φαιάς ουσίας του εγκεφάλου. Επιληπτικά επεισόδια παθαίνουν εκατομμύρια άτομα κάθε μέρα, με επιπτώσεις τόσο σωματικές όσο και ψυχολογικές. Άγχος, διαταραχή της παραγωγικότητας, απώλεια της θέσης εργασίας του ατόμου, κοινωνική απομόνωση είναι μερικές μόνο από τις οδυνηρές επιπτώσεις της επιληψίας.

Η πρόβλεψη των επιληπτικών κρίσεων θα βελτιώσει την ποιότητα της ζωής εκατομμυρίων ανθρώπων που πάσχουν από ξαφνικές κρίσεις. Διαφορετικά χαρακτηριστικά του ηλεκτροεγκεφαλογραφήματος (EEG) χρησιμοποιούνται για την πρόβλεψη επιληπτικών κρίσεων. Νευρολόγοι και τεχνικοί που διαβάζουν τα σήματα των εγκεφαλογραφημάτων πρέπει να δαπανήσουν ένα σημαντικό χρονικό διάστημα για να διαβάσουν και να ερμηνεύουν τα σήματα για κάθε ασθενή.

Ένα άλλο μειονέκτημα αυτής της οπτικής ερμηνείας των σημάτων του ηλεκτροεγκεφαλογραφήματος είναι ότι η διάγνωση μπορεί να διαφέρει μεταξύ των γιατρών, για να μην αναφέρουμε τη πιθανότητα αποτυχίας του εντοπισμού των σημάτων κάποιας επικείμενης επιληπτικής κρίσης. Ως εκ τούτου, μια αυτοματοποιημένη πρόβλεψη των επιληπτικών κρίσεων θα βοηθήσει όχι μόνο τους γιατρούς στη διάγνωση τους, αλλά θα βελτιώσει επίσης την ποιότητα της φροντίδας του ασθενούς και τελικά θα οδηγήσει στην ανάπτυξη εμφυτεύσιμων έξυπνων συσκευών που προειδοποιούν τους ασθενείς πριν την εκδήλωση της επιληπτικής κρίσης.

- **Χρηματιστηριακές προβλέψεις**: Οι διακυμάνσεις των τιμών των **μετοχών** και των χρηματιστηριακών δεικτών είναι ακόμα ένα παράδειγμα ενός πολύπλοκου, πολυδιάστατου, αλλά και σε ορισμένες περιπτώσεις εν μέρει ντετερμινιστικού φαινομένου. Τα νευρωνικά δίκτυα χρησιμοποιούνται από πολλούς τεχνικούς αναλυτές, ώστε να κάνουν προβλέψεις σχετικά με τις τιμές των μετοχών, βασιζόμενοι σε ένα μεγάλο αριθμό παραγόντων, όπως τις προηγούμενες επιδόσεις άλλων αποθεμάτων και διαφόρων οικονομικών δεικτών.

- **Παρακολούθηση της κατάστασης των μηχανημάτων**: Τα νευρωνικά δίκτυα μπορούν να συμβάλλουν στη μείωση του κόστους με την εξασφάλιση της

πρόσθετης εμπειρογνωμοσύνης για τον προγραμματισμό **προληπτικής συντήρησης των μηχανημάτων**. Ένα νευρωνικό δίκτυο, λοιπόν, μπορεί να εκπαιδευτεί με τέτοιο τρόπο, ώστε να διακρίνει από τους ήχους τους οποίους παράγει μια μηχανή είτε αν εκτελεί κανονικά τις λειτουργίες της, είτε βρίσκεται στα πρόθυρα εμφάνισης οποιασδήποτε δυσλειτουργίας. Μετά από αυτήν την περίοδο εκπαιδευτικής κατάρτισης, η εμπειρία του ίδιου δικτύου είναι δυνατό να χρησιμοποιηθεί με σκοπό την προειδοποίηση ενός τεχνικού για κάποια επικείμενη βλάβη προτού συμβεί και ενδεχομένως προκαλέσει πολυδάπανες και απρόβλεπτες χρονικές καθυστερήσεις.

- **Αναγνώριση εικόνων, κειμένων και γενικά προτύπων (pattern recognition)**: Η εφαρμογή αυτή περιλαμβάνει πάρα πολλές δραστηριότητες, από τις πλέον επιτυχείς των νευρωνικών δικτύων. Μία από αυτές είναι το πρόγραμμα «Omnipage» που το ανέπτυξε η εταιρία Caere (που τώρα λέγεται ScanSoft) το 1994 και υλοποιείται σε ένα απλό PC. Το πρόγραμμα διαβάζει τυπωμένα κείμενα με σαρωτή (scanner) και τα μετατρέπει σε χαρακτήρες ASCII. Μάλιστα το πρόγραμμα αυτό δουλεύει ικανοποιητικά, έστω και αν τα γράμματα είναι μερικώς καταστραμμένα, όπως λ.χ. συμβαίνει συχνά σε σελίδες fax.

  Η εταιρία Nestor έχει επίσης αναπτύξει ένα πρόγραμμα που αναγνωρίζει την γραφή Κάντζι (ιαπωνική γραφή) και έτσι μεταφράζει αυτόματα διάφορα κείμενα στα Αγγλικά. Η αρχική έκδοση μπορούσε να αναγνωρίσει 2500 χαρακτήρες με επιτυχία 92%. Ο μέσος Ιάπωνας αναγνωρίζει περίπου 2000–3000 τέτοιους χαρακτήρες. Το δίκτυο αυτό χρησιμοποιεί μία γενικευμένη λογική που θα μπορούσε εύκολα να εφαρμοσθεί και σε άλλες γλώσσες, όπως Κυριλλικά, Εβραικά κτλ.

  Ένα άλλο γνωστό πρόβλημα είναι η μετατροπή κειμένου σε φωνή, και βέβαια το αντίστροφο. Ενα γνωστό πρόγραμμα, το NETtalk, κάνει ακριβώς αυτό, δηλ. ένα δίκτυο εκπαιδεύεται στο να διαβάζει γραπτά κείμενα και να τα απαγγέλλει [8]. Το δίκτυο έχει 309 νευρώνες με 18629 συνάψεις σε 3 διαφορετικά επίπεδα. Η είσοδος του δικτύου αποτελείται από 7 ομάδες νευρώνων και κάθε ομάδα από 29 νευρώνες (ένα για τα 26 γράμματα, ένα για το κενό, την τελεία, και το κόμα). Η έξοδος απο- τελείται από 26 νευρώνες, ενώ το μεσαίο επίπεδο έχει 80 νευρώνες. Το πρόγραμμα εξετάζει ένα παράθυρο με 7 χαρακτήρες, το οποίο συνεχώς μετακινείται κατά ένα χαρακτήρα, διορθώνει τα σφάλματα του και μετά την εκπαίδευση του το δίκτυο μπο- ρεί να βρεί τους κανόνες για τα φωνήεντα, τα κενά κτλ. και μεταβάλλει τα βάρη του ανάλογα. Στην αρχή η απαγγελία ήταν ακατανόητη, μετά ήταν νηπιακής μορφής και τελικά έφθασε σε 95% αναγνωρίσιμης και κατανοητής ομιλίας.

2. Τα ESN σε χρήση

2.1: Γενικά

Όπως αναφέρθηκε προηγουμένως, ένα Reservoir Computer απεικονίζει την έξοδο σε μία υψηλότερη διάσταση. Συγκεκριμένα, όσο πιο **πολύπλοκο είναι το reservoir** του νευρωνικού μας δικτύου, **τόσο μεγαλύτερη είναι και η διάσταση** στην οποία απεικονίζεται το αποτέλεσμα, συνεπώς τόσο **δυσκολότερη και η μελέτη του**. Επιπλέον, ο χρόνος που απαιτείται για τους υπολογισμούς είναι επίσης ανάλογος της πολυπλοκότητας αυτής. Ωστόσο, **όσες περισσότερες συνάψεις υπάρχουν, τόσο μικρότερο σφάλμα** έχει το τελικό μας αποτέλεσμα, άρα η πρόβλεψη είναι καλύτερη. Κατανοούμε λοιπόν την ανάγκη για βελτιστοποίηση της πολυπλοκότητας του reservoir σε σχέση με το αποτέλεσμα που θέλουμε να πάρουμε.

Πως θα γίνει όμως αυτό; Θα πρέπει να γίνει μία μοντελοποίηση του reservoir έτσι ώστε να μπορεί να μελετηθεί.

Ένα νευρωνικό δίκτυο είναι γενικά ένα **σταθμισμένο γράφημα** όπου οι κόμβοι είναι οι νευρώνες και οι ακμές είναι οι συνάψεις. (Σταθμισμένο λέγεται το γράφημα του οποίου οι ακμές έχουν κάποιο βάρος). Κάθε κόμβος χαρακτηρίζεται από μια **εξίσωση "εξέλιξης"** (evolution equation), όπου η κατάσταση των νευρώνων εξαρτάται από τους **γείτονές** του (προσυναπτικοί νευρώνες). Τα συναπτικά βάρη μπορεί να είναι σταθερά ή να **εξελίσονται διαχρονικά** (συναπτική πλαστικότητα) σύμφωνα με τη κατάσταση και την ιστορία των δύο κόμβων με τους οποίους συνδέονται (προ- και μετα-συναπτικοί νευρώνες).

Στη προκειμένη όμως περίπτωση **θα μοντελοποιήσουμε το νευρωνικό μας δίκτυο και συγκεκριμένα το reservoir σαν δυναμικό σύστημα**. Αυτό είναι δυνατό, καθώς το reservoir δεν είναι τίποτα άλλο από ένα δυναμικό σύστημα το οποίο, κατευθυνόμενο από τα δεδομένα εισόδου, και δια μέσω των νευρώνων κάνει μια **μη γραμμική απεικόνιση των δεδομένων** αυτών και μας δίνει ένα αποτέλεσμα σε n-διαστάσεις. Η μοντελοποίηση αυτή μας διευκολύνει καθώς διαθέτουμε πολλά μαθηματικά εργαλεία για τη μελέτη δυναμικών συστημάτων. Ένα από αυτά είναι οι **εκθέτες Lyapunov**, οι οποίοι προσδιορίζουν τη χαοτική ή όχι συμπεριφορά ενός δυναμικού συστήματος και για τους οποίους θα δούμε περισσότερες λεπτομέρειες στη συνέχεια.

Για την ώρα θα μελετήσουμε τη λειτουργία ενός ESN με τη βοήθεια του μαθηματικού πακέτου Matlab. Συγκεκριμένα θα **προβλέψουμε και θα ανακατασκευάσουμε το χώρο φάσεων μιας χρονοσειράς**. Θα χρησιμοποιήσουμε ένα σύστημα διακριτού χρόνου, που ονομάζεται NARMA (Nonlinear AutoRegressive Moving Average), το οποίο είναι δύσκολο να μοντελοποιηθεί λόγω της **μη γραμμικότητάς** του και τη χρήση πολλής μνήμης για υπολογισμούς:

$$y(t+1) = 0.3y(t) + 0.05y(t)\sum_{i=1}^{3} y(t-i) + 1.5s(t-9)s(t) + 0.1$$

Έχοντας λοιπόν στη διάθεσή μας ένα σύνολο τιμών της συγκεκριμένης χρονοσειράς, θα εκπαιδεύσουμε το νευρωνικό δίκτυο, ώστε να πάρουμε μια πρόβλεψη για τις επόμενες τιμές και θα προσπαθήσουμε να ανακατασκευάσουμε το χώρο φάσης, ώστε να εξάγουμε γενικότερες πληροφορίες για το σύστημα που γεννά τη χρονοσειρά.

Υπάρχουν δύο διαφορετικές τεχνικές για την εκπαίδευση ενός νευρωνικού δικτύου: από το σύνολο των δεδομένων (**batch learning**) και σε πραγματικό χρόνο (**online learning**). Η κατανόηση των ομοιοτήτων και των διαφορών τους είναι σημαντική, προκειμένου να είμαστε σε θέση να δημιουργήσουμε ακριβή συστήματα πρόβλεψης.

Η online εκπαίδευση μπορεί να συνοψιστεί ως εξής: Για κάθε αντικείμενο του συνόλου εκπαίδευσης υπολογίζουμε τα βάρη και τις τιμές της μεροληψίας (δέλτα) και προσαρμόζουμε τα βάρη και τις τιμές μεροληψίας χρησιμοποιώντας αυτά τα δέλτα.

Δηλαδή, κατά την online εκπαίδευση, τα βάρη και οι τιμές μεροληψίας προσαρμόζονται για κάθε στοιχείο του συνόλου με βάση τη διαφορά μεταξύ της υπολογισμένης εξόδου και της εξόδου από τα δεδομένα εκπαίδευσης.

Απ' την άλλη, **η εκπαίδευση από το σύνολο δεδομένων μπορεί να συνοψιστεί ως εξής:** Για κάθε αντικείμενο του συνόλου εκπαίδευσης, υπολογίζουμε τα βάρη και τις τιμές της μεροληψίας (δέλτα), συσσωρεύουμε τα δέλτα και στη συνέχεια προσαρμόζουμε τα βάρη και τις τιμές μεροληψίας χρησιμοποιώντας αυτά τα συσσωρευμένα δέλτα.

Στην εκπαίδευση από το σύνολο των δεδομένων, οι προσαρμοσμένες τιμές δέλτα συσσωρεύονται από όλα τα δεδομένα εκπαίδευσης, για να δώσουν ένα σύνολο από τιμές δέλτα, και στη συνέχεια τα αθροισμένα δέλτα εφαρμόζονται σε κάθε βάρος και τιμή μεροληψίας.

(*Οι πιο σημαντικοί κώδικες για τη κατασκευή και τη λειτουργία του ESN υπάρχουν στο παράρτημα)*

2.2: Εκμάθηση από το σύνολο των δεδομένων (batch learning)

Παρακάτω θα δούμε πως λειτουργεί ένα ESN με τη μέθοδο εκμάθησης από το σύνολο των δεδομένων.

2.2.1: Συσχέτιση αριθμού νευρώνων- σφαλμάτων

Καταρχάς, θα μελετήσουμε τη συσχέτιση του αριθμού των νευρώνων μέσα στο reservoir και των σφαλμάτων του αποτελέσματος, με τη βοήθεια του πακέτου

ESNToolbox του Matlab. Το πακέτο αυτο, παράγει, εκαπαιδεύει και τεστάρει δεδομένα χρησιμοποιώντας την αρχιτεκτονική του ESN.

Αρχικά, χωρίζουμε το σύνολο των δεδομένων τιμών σε δύο σύνολα, σε αυτό που θα χρησιμοποιήσουμε για την **εκπαίδευση** του δικτύου και σε αυτό που θα χρησιμοποιήσουμε για να **ελέγξουμε** την αξιοπιστία της εκπαίδευσης αυτής. Ο διαχωρισμός γίνεται με τυχαίο τρόπο για να έχουμε αντιπροσωπευτικό δείγμα κατά την εκπαίδευση. Στη συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε το 50% των δεδομένων μας για εκπαίδευση.

Στα τρία παραδείγματα που ακολουθούν θα δούμε τη **σημαντικότητα του πλήθους των εσωτερικών νευρώνων στη σωστή εξαγωγή αποτελέσματος** με τη βοήθεια γραφικών παραστάσεων και μελετώντας το σφάλμα που προκύπτει.

Για τη καλύτερη και πιο αντικειμενική εξαγωγή συμπερασμάτων, πήραμε το **μέσο όρο 1000 σφαλμάτων** καθώς και τις **διασπορές** τους.
Στη συνέχεια βλέπουμε το script με το οποίο πήραμε τα δεδομένα μας:

```
%Αρχικοποιώ τα διανύσματα στα οποία θα αποθηκευτούν τα αποτελέσματα για τα
%σφάλματα εκπαίδευσης και δοκιμής, ώστε να έχουν 1000 θέσεις, όσες δηλαδή
%και οι επαναλήψεις που θα κάνω

clear all;
TrError= 1:1000;
TesError= 1:1000;


%Παραγωγή των δεδομένων
for i= 1:100


sequenceLength = 1000;


disp('Generating data .............');
disp(sprintf('Sequence Length'));


%Ορίζω την τάξη της σειράς NARMA και παράγω τις τιμές της χρονοσειράς
systemOrder = 3 ;


 [inputSequence outputSequence] = generate_NARMA_sequence(sequenceLength ,
systemOrder) ;


 %Χωρίζω τα δεδομένα σε training και test data
train_fraction = 0.5;


%Χρησιμοποιώ 30% για training και 50%  για testing
[trainInputSequence, testInputSequence] = ...
    split_train_test(inputSequence,train_fraction);
[trainOutputSequence,testOutputSequence] = ...
    split_train_test(outputSequence,train_fraction);


%Παραγωγή του esn
nInputUnits = 2; nInternalUnits = 300; nOutputUnits = 1;
```

```
%Καθορισμός της ποσότητας των μονάδων εισόδου, εξόδου και εσωτερικού
esn = generate_esn(nInputUnits, nInternalUnits, nOutputUnits, ...
    'spectralRadius',0.5,'inputScaling',[0.1;0.1],'inputShift',[0;0], ...
    'teacherScaling',[0.3],'teacherShift',[-0.2],'feedbackScaling', 0, ...
    'type', 'plain_esn');




esn.internalWeights = esn.spectralRadius * esn.internalWeights_UnitSR;


%Εκπαίδευση του ESN
nForgetPoints = 100 ; % discard the first 100 points
[trainedEsn stateMatrix] = ...
    train_esn(trainInputSequence, trainOutputSequence, esn, nForgetPoints) ;


%Αποθήκευση του εκαπαιδευμένου ESN και εμφάνιση των διαγραμμάτων των
καταστάσεων των μονάδων του  reservoir.
save_esn(trainedEsn, 'esn_narma_demo_1');


%Υπολογισμός της εξόδου του εκπαιδευμένου ESN στα δεδομένα για την
εκαπαίδευση και το τεστάρισμα
nPoints = 200 ;
%plot_states(stateMatrix,[1 2 3 4], nPoints, 1, 'traces of first 4 reservoir
units') ;



%Διαγραφή των πρώτων 100 παροδικών αποτελεσμάτων
nForgetPoints = 100 ;
predictedTrainOutput = test_esn(trainInputSequence, trainedEsn,
nForgetPoints);
predictedTestOutput = test_esn(testInputSequence,  trainedEsn,
nForgetPoints) ;


%Κατασκευή διαγραμμάτων εισόδων και εξόδων
nPlotPoints = 100 ;
plot_sequence(trainOutputSequence(nForgetPoints+1:end,:),
predictedTrainOutput, nPlotPoints,...
   'training: teacher sequence (red) vs predicted sequence (blue)');
plot_sequence(testOutputSequence(nForgetPoints+1:end,:),
predictedTestOutput, nPlotPoints, ...
    'testing: teacher sequence (red) vs predicted sequence (blue)') ;


%Υπολογισμός του σφάλματος εκπαίδευσης
trainError = compute_NRMSE(predictedTrainOutput, trainOutputSequence);
disp(sprintf('train NRMSE = %s', trainError))
TrError(i)=trainError;


%Υπολογισμός του σφάλματος δοκιμής
testError = compute_NRMSE(predictedTestOutput, testOutputSequence);
disp(sprintf('test NRMSE = %s', testError))
TesError(i)=testError;


end


TrError;
TesError;
```

```
%Υπολογισμός μέσης τιμής και διασποράς των διανυσμάτων των σφαλμάτων
mean(TrError)
mean(TesError)
var(TrError)
var(TesError)
```

Τα αποτελέσματα που παίρνουμε είναι τα εξής:

- **10 νευρώνες**

Αρχικά κάνουμε μία δοκιμή μόνο με 10 νευρώνες στο reservoir. Ο αριθμός αυτός είναι **σχετικά μικρός** και γι αυτό δεν αναμένουμε πολύ καλά αποτελέσματα. Ωστόσο παρατηρούμε ότι ο χρόνος διεκπεραίωσης όλης της διαδικασίας είναι πολύ μικρός. Ας δούμε όμως αναλυτικά τα αποτελέσματα:

Βλέπουμε το διάνυσμα με το σφάλμα της εκπαίδευσης (training error). Το σφάλμα αυτό είναι η ρίζα του κανονικοποιημένου μέσου τετραγωνικού σφάλματος (Normalized root mean squared error - NRMSE) , το οποίο δίνεται απ τον τύπο:

$$NRMSE = \frac{\sqrt{\langle (y-\hat{y})^2 \rangle}}{\max(\hat{y}) - \min(\hat{y})}$$

Όπου ŷ είναι η τιμή που προέβλεψε το νευρωνικό δίκτυο.

```
TrError =

  Columns 1 through 14

    0.2217    0.2061    0.2131    0.2222    0.2244    0.2183    0.2341    0.2303    0.2223    0.2293    0.2275

  Columns 15 through 28

    0.2230    0.2278    0.2228    0.2082    0.2216    0.2287    0.2206    0.2220    0.2124    0.2202    0.2309

  Columns 29 through 42

    0.2131    0.2127    0.2232    0.2214    0.2172    0.2367    0.2169    0.2233    0.2189    0.2099    0.2281

  Columns 43 through 56

    0.2256    0.2280    0.2240    0.2179    0.2210    0.2213    0.2266    0.2101    0.2281    0.2164    0.2340

  Columns 57 through 70

    0.2154    0.2238    0.2127    0.2032    0.2313    0.2163    0.2095    0.2211    0.2026    0.2149    0.2259
```

Βλέπουμε λοιπόν ότι το σφάλμα μας βρίσκεται περίπου στο 0.2, που είναι αρκετά μεγάλο. Στα ίδια πλαίσια κυμαίνεται και το σφάλμα που προκύπτει κατά τον έλεγχο του δικτύου (testing error):

```
TesError =

  Columns 1 through 14

    0.2141    0.2392    0.2284    0.2279    0.2381    0.2411    0.2290    0.2342    0.2238    0.2337    0.2235

  Columns 15 through 28

    0.2201    0.2150    0.2461    0.2196    0.2199    0.2224    0.2211    0.2352    0.2260    0.2317    0.2268

  Columns 29 through 42

    0.2267    0.2342    0.2248    0.2367    0.2394    0.2202    0.2218    0.2313    0.2277    0.2242    0.2345

  Columns 43 through 56

    0.2253    0.2313    0.2345    0.2394    0.2221    0.2257    0.2434    0.2293    0.2157    0.2294    0.2282

  Columns 57 through 70

    0.2344    0.2281    0.2483    0.2289    0.2395    0.2503    0.2332    0.2444    0.2236    0.2296    0.2356
```

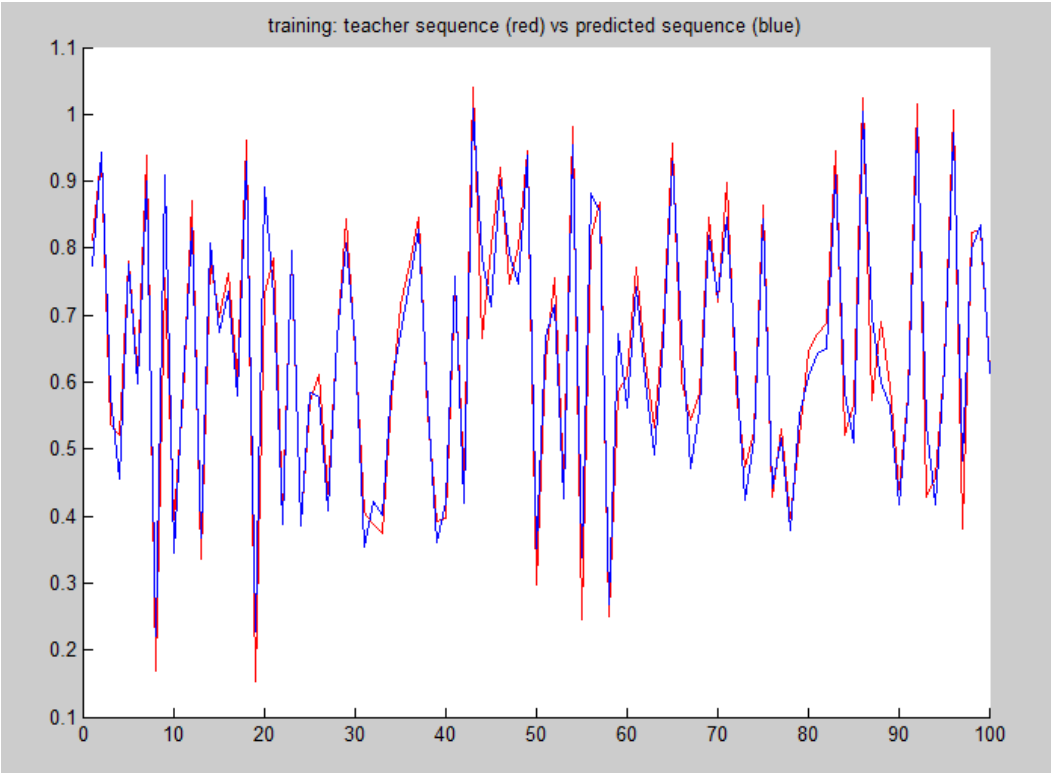
Για να έχουμε μια καλύτερη άποψη, παίρνουμε τα στατιστικά των δύο σφαλμάτων, δηλαδή τη μέση τιμή και τη διασπορά των 1000 τιμών:

```
>> mean(TrError)

ans =

    0.2208

>> mean(TesError)

ans =

    0.2281
```

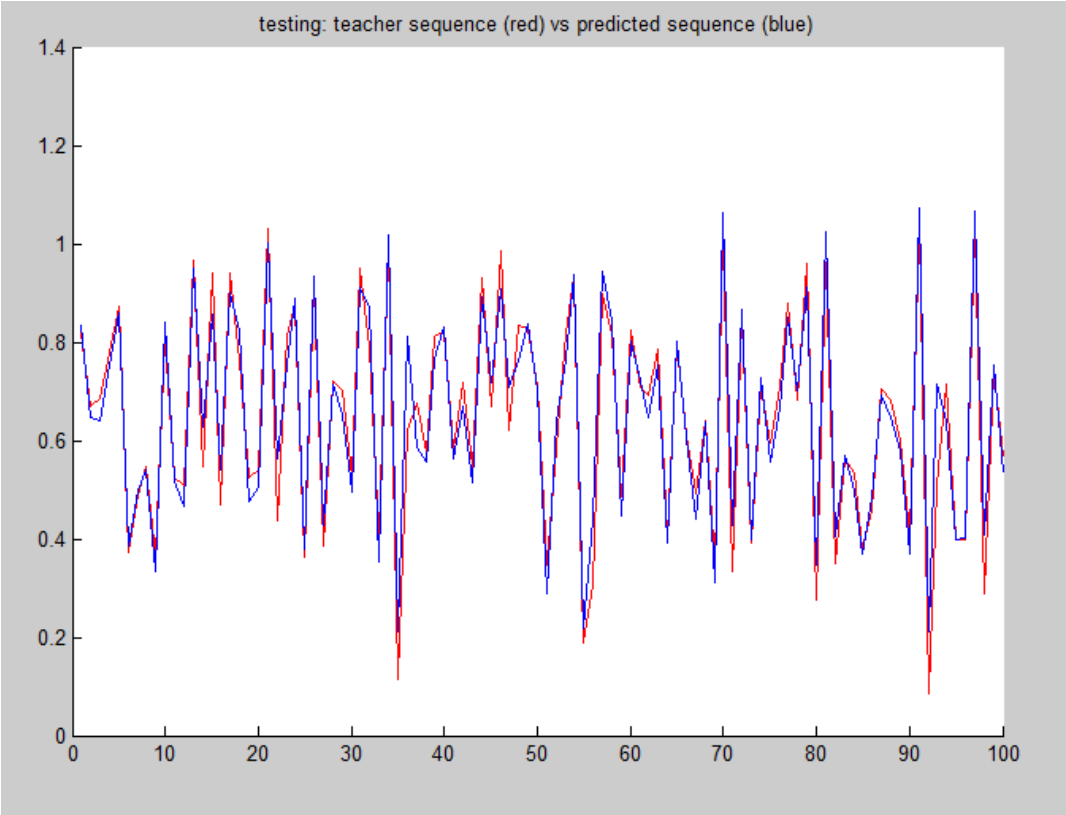
```
>> var(TrError)

ans =

   6.8540e-005

>> var(TesError)

ans =

   7.1910e-005
```

Παρατηρούμε ότι η μέση τιμή των δύο σφαλμάτων διαφέρει ελάχιστα και η διασπορά τους είναι πολύ μικρή. Καταλήγουμε λοιπόν στο συμπέρασμα ότι με reservoir 10 νευρώνων, το σφάλμα μας είναι περίπου 0,22.

Παρακάτω μπορούμε να δούμε τα γραφήματα της ακολουθίας εκπαίδευσης με αυτή που έχει προβλεφθεί κατά τη διαδικασία εκπαίδευσης (training) και δοκιμής (testing) .



Σχήμα 2.1: Γράφημα ακολουθιών κατά την εκπαίδευση (10 νευρώνες).



Σχήμα 2.2: Γράφημα ακολουθιών κατά τη δοκιμή (10 νευρώνες).

Όπως βλέπουμε από τα παραπάνω γραφήματα, το δίκτυο καταφέρνει να προβλέψει αρκετά καλά τις τιμές τις ακολουθίας, χωρίς να έχουμε όμως πλήρη ταύτιση.

- **300 νευρώνες**

Στη συνέχεια αυξάνουμε τον αριθμό των νευρώνων σε 300. Με την αύξηση αυτή περιμένουμε πολύ πιο ακριβείς προβλέψεις και συνεπώς μικρότερα σφάλματα αλλά και περισσότερο χρόνο λειτουργίας του νευρωνικού μας δικτύου.

```
Generating data .............
Sequence Length 1000
train NRMSE = 1.940601e-002
test NRMSE = 9.990775e-002
Generating data .............
Sequence Length 1000
train NRMSE = 1.877634e-002
test NRMSE = 8.266432e-002
Generating data .............
Sequence Length 1000
train NRMSE = 1.886967e-002
test NRMSE = 9.336308e-002
Generating data .............
Sequence Length 1000
train NRMSE = 1.890038e-002
test NRMSE = 8.968956e-002
Generating data .............
Sequence Length 1000
train NRMSE = 1.663945e-002
test NRMSE = 8.871299e-002
Generating data .............
Sequence Length 1000
train NRMSE = 1.763884e-002
test NRMSE = 7.577136e-002
```

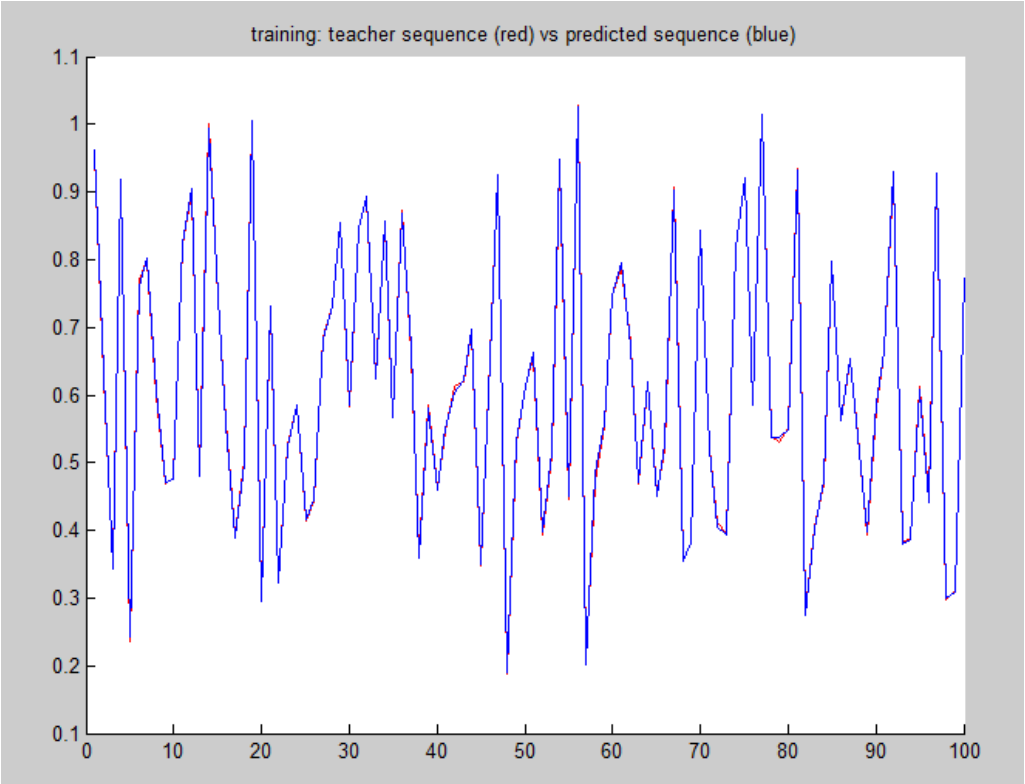
Επαναλαμβάνοντας την ίδια διαδικασία όπως και προηγουμένως παίρνουμε τα εξής αποτελέσματα για τα σφάλματα:

```
>> mean(TrError)

ans =

    0.0175

>> mean(TesError)

ans =

    0.0904
```

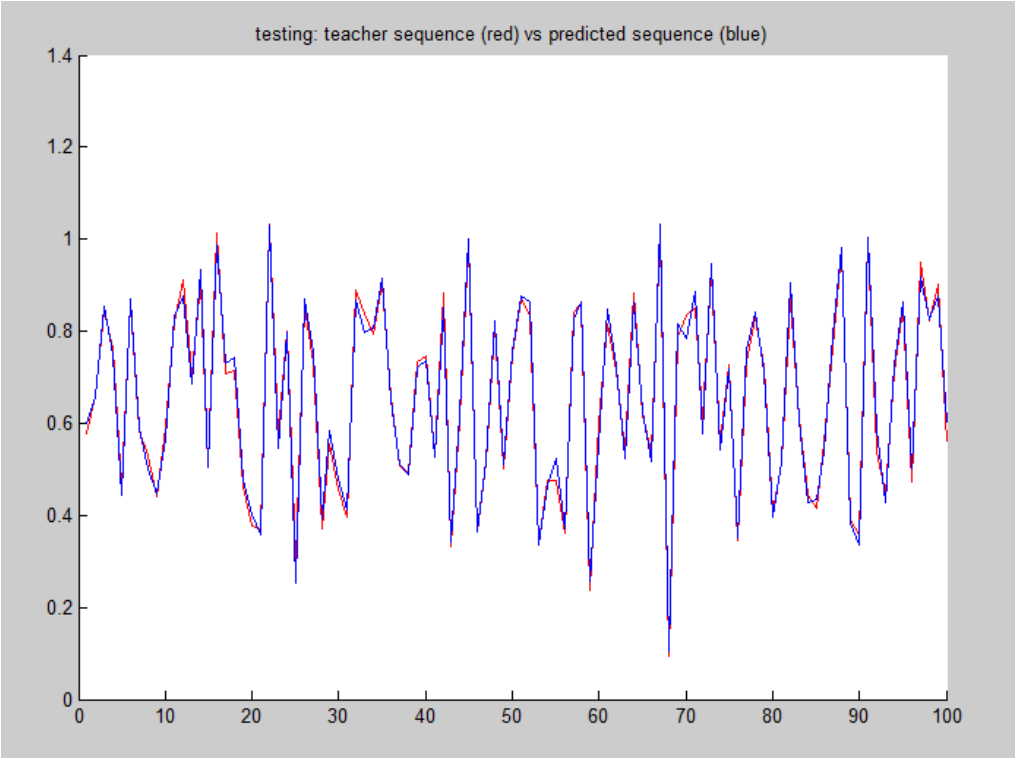
```
>> var(TrError)

ans =

  4.5811e-006

>> var(TesError)

ans =

  1.0746e-004
```

Παρατηρούμε ότι όντως το σφάλμα έχει μειωθεί πολύ σε σχέση με το προηγούμενο παράδειγμα, αφού το μεν training error είναι 0.0175 κατά μέσο όρο, ενώ το δε testing error είναι 0.0904. Επίσης να σημειώσουμε εδώ ότι παρόλο που ο χρόνος επεξεργασίας των δεδομένων αυξήθηκε, μπορούμε να πούμε ότι **η διαδικασία δεν έγινε τόσο αργή** ώστε να το θεωρήσουμε σαν μειονέκτημα.

Ας δούμε και τα σχετικά διαγράμματα για να έχουμε ένα μέτρο σύγκρισης:



Σχήμα 2.3: Γράφημα ακολουθιών κατά την εκπαίδευση (300 νευρώνες).



Σχήμα 2.4: Γράφημα ακολουθιών κατά τη δοκιμή (300 νευρώνες).

Στα γραφήματα βλέπουμε μια πολύ καλύτερη πρόβλεψη των πραγματικών τιμών. Ειδικά κατά την εκπαίδευση, το γράφημα της προβλεπόμενης ακολουθίας σχεδόν ταυτίζεται με αυτό της εκπαίδευσης, κάτι που δικαιολογεί και το ελάχιστο σφάλμα που βρήκαμε.

- **500 νευρώνες**

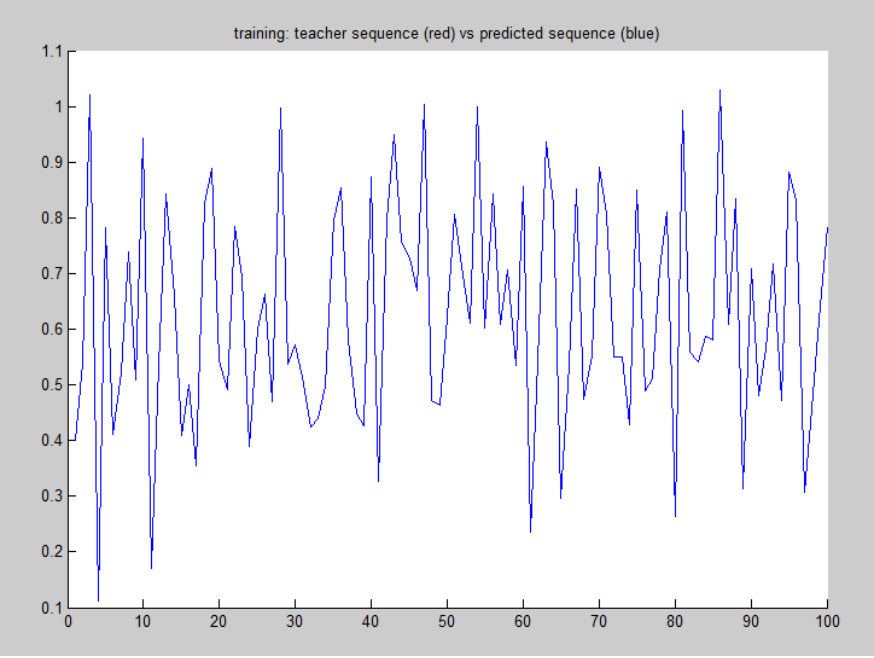
Αυξάνουμε ακόμα περισσότερο τον αριθμό των νευρώνων σε 500 και θέτουμε το ESN μας σε λειτουργία. Ο χρόνος επεξεργασίας είναι πλέον πολύ μεγάλος, με συνολική διάρκεια που ξεπερνάει τα 10-15 λεπτά, γεγονός που δεν το κάνει καθόλου εύχρηστο. Επιπλέον τα αποτελέσματα είναι παρόμοια με αυτά που πήραμε χρησιμοποιώντας τους 300 νευρώνες.

Πιο αναλυτικά, όπως βλέπουμε και παρακάτω, έχουμε ένα training error σχεδόν μηδενικό και ένα testing error παρόμοιο με τη προηγούμενη περίπτωση

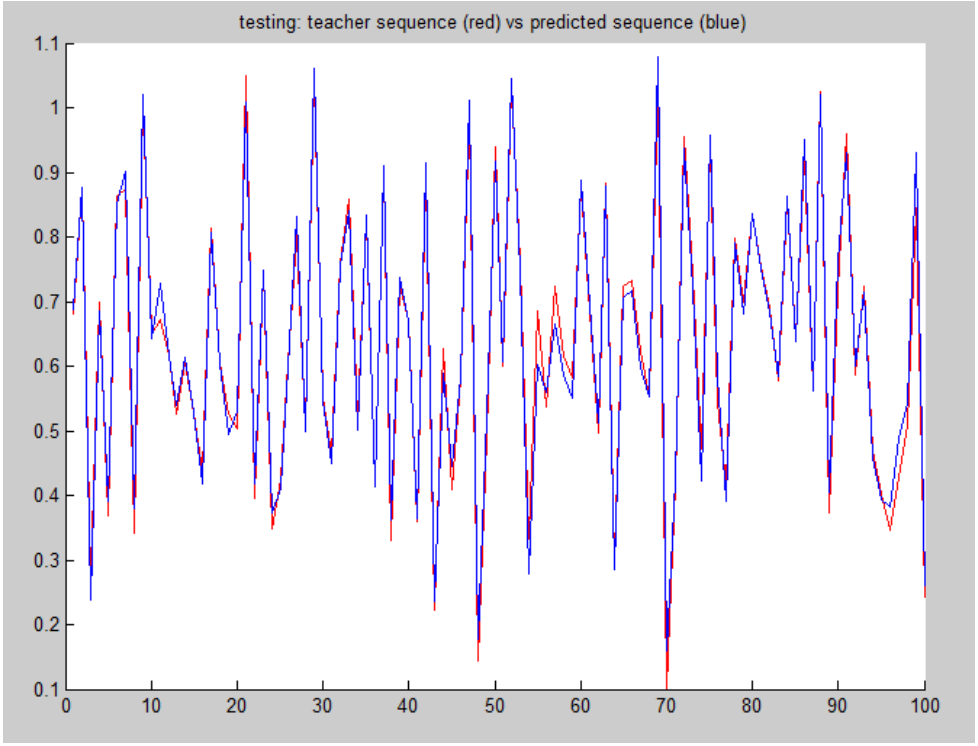
```
>> mean(TrError)

ans =

  6.5316e-009

>> mean(TesError)

ans =

    0.1446
```

```
>> var(TrError)

ans =

  2.0815e-017

>> var(TesError)

ans =

  4.1266e-004
```

Στα αντίστοιχα διαγράμματα, παρατηρούμε ότι κατά την εκπαίδευση έχουμε πλήρη ταύτιση τιμών (σχεδόν μηδενικό σφάλμα), ενώ κατά τη δοκιμή η εικόνα είναι ίδια με αυτή του διαγράμματος των 300 νευρώνων:



Σχήμα 2.5: Γράφημα ακολουθιών κατά την εκπαίδευση (500 νευρώνες).



Σχήμα 2.6: Γράφημα ακολουθιών κατά τη δοκιμή (500 νευρώνες).

Συμπεραίνουμε λοιπόν, ότι από τα τρία νευρωνικά δίκτυα που χρησιμοποιήσαμε, καλύτερο ήταν αυτό με τους 300 νευρώνες, καθώς μας έδωσε πολύ καλές προβλέψεις σε σχετικά μικρό χρόνο λειτουργίας. Ωστόσο, ανάλογα με το πρόβλημα που έχουμε να αντιμετωπίσουμε και την ευαισθησία στο αποτέλεσμά μας, μπορούμε να αλλάζουμε το reservoir με όποιον τρόπο μας βολεύει, έτσι ώστε να πετυχαίνουμε κάθε φορά το βέλτιστο αποτέλεσμα.

2.2.2 Συσχέτιση ποσοστού δεδομένων για εκπαίδευση- σφαλμάτων

Ένας άλλος παράγοντας που παίζει σημαντικό ρόλο στην αποδοτικότητα του ESN μας είναι το **ποσοστό των δεδομένων που θα χρησιμοποιηθούν κατά την εκπαίδευση**. Γενικά όσο περισσότερο εκπαιδευθεί το νευρωνικό δίκτυο, τόσο πιο καλές προβλέψεις θα κάνει.

Θα δούμε αρχικά λίγες επιπλέον πληροφορίες για την εκπαίδευση του ESN, και στη συνέχεια με παραδείγματα και χρησιμοποιώντας το ESN Toolbox όπως και προηγουμένως, θα ελέγξουμε την αποδοτικότητα τριών ESN με διαφορετική εκπαίδευση το καθένα.

Το έργο της εκπαίδευσης ενός νευρωνικού δικτύου από τα ήδη υπάρχοντα δεδομένα μπορεί να χωριστεί σε τρία στάδια:

❖ **Κατασκευή του ESN**

Κατασκευάζουμε ένα τυχαίο δυνάμικο reservoir διάστασης N. Στο συγκεκριμένο παράδειγμα χρησιμοποιήσαμε N=300, δηλαδή reservoir με 300 input units.

❖ **Συλλογή των καταστάσεων του reservoir**

Τρέχουμε το reservoir τόσες φορές, όσος ο αριθμός των δεδομένων που θα χρησιμοποιήσουμε για την εκπαίδευση. Αυτό έχει σαν αποτέλεσμα να πάρουμε μια ακολουθία καταστάσεων του reservoir N-διαστάσεων. Κάθε στοιχείο του διανύσματος είναι ένας μη γραμμικός μετασχηματισμός των δεδομένων εισόδου.

❖ **Υπολογισμός βαρών εξόδου**

Χρησιμοποιώντας γραμμική παλινδρόμηση στα βάρη της εξόδου των δεδομένων εκμάθησης στις καταστάσεις του reservoir, υπολογίζουμε τα βάρη των εξόδων των δεδομένων. Δημιουργούνται έτσι σωστές συνδέσεις μεταξύ του reservoir και των εξόδων. Η εκπαίδευση έχει τελειώσει και το reservoir είναι έτοιμο για χρήση.

Στη προκειμένη περίπτωση θα χωρίσουμε τα δεδομένα μας με τέτοιο τρόπο ώστε τη πρώτη φορά να πάρουμε το 30 % για εκπαίδευση, τη δεύτερη το 50% και τη τρίτη το 80%. Για να κάνουμε αυτή τη διαφοροποίηση αλλάζουμε στον κώδικα κατασκευής του ESN τον αντίστοιχο αριθμό στη μεταβλητή train fraction.

train_fraction = 0.5;

Παρακάτω βλέπουμε τα αποτελέσματα που πήραμε:

- **Ποσοστό εκπαίδευσης: 30%**

```
>> mean(TrError)

ans =

  4.8584e-010

>> mean(TesError)

ans =

    0.1415
```

- **Ποσοστό εκπαίδευσης: 50%**

```
>> mean(TrError)

ans =

    0.0175

>> mean(TesError)

ans =

    0.0904
```

- **Ποσοστό εκπαίδευσης: 80%**

```
>> mean(TrError)

ans =

    0.0291

>> mean(TesError)

ans =

    0.0584
```

Όπως ήταν αναμενόμενο το σφάλμα (Test error) μειώνεται όσο το ποσοστό της εκπαίδευσης του νευρωνικού δικτύου αυξάνεται. Συγκεκριμένα ξεκινώντας από 0.14, πέφτει στο 0.09 και στη συνέχεια στο 0.06 περίπου.

Εδώ να σημειώσουμε ότι επειδή ο τύπος του σφάλματος δεν είναι γραμμικός, η μείωσή του δεν γίνεται εύκολα. Οπότε όταν παρατηρούμε μεγάλη μείωση του σφάλματος, αντιλαμβανόμαστε ότι η επίδοσή του νευρωνικού δικτύου έχει αυξηθεί σημαντικά.

2.3: Εκμάθηση σε πραγματικό χρόνο (online learning)

Στη συνέχεια θα δούμε πως λειτουργεί ένα ESN με τη μέθοδο εκμάθησης σε πραγματικό χρόνο. Στον κώδικά μας προσθέτουμε την εξής εντολή, η οποία αλλάζει τη μέθοδο εκμάθησης σε online:

```
esn = generate_esn(nInputUnits, nInternalUnits, nOutputUnits, ...
'spectralRadius',0.4,'inputScaling',[0.1;0.5],'inputShift',[0;1], ...
'teacherScaling',[0.3],'teacherShift',[-0.2],'feedbackScaling',0, ...
'learningMode', 'online' , 'RLS_lambda',0.9999995 , 'RLS_delta',0.000001, ...
'noiseLevel' , 0.00000000)
```

2.3.1: Συσχέτιση αριθμού νευρώνων- σφαλμάτων

Όπως και στη προηγούμενη ενότητα, θα ασχοληθούμε με το πόσο αποδοτικό είναι το νευρωνικό δίκτυό μας σε σχέση με τον αριθμό των νευρώνων στο reservoir του. Και πάλι θα τρέξουμε το ESN μας για 10, 300 και 500 νευρώνες.

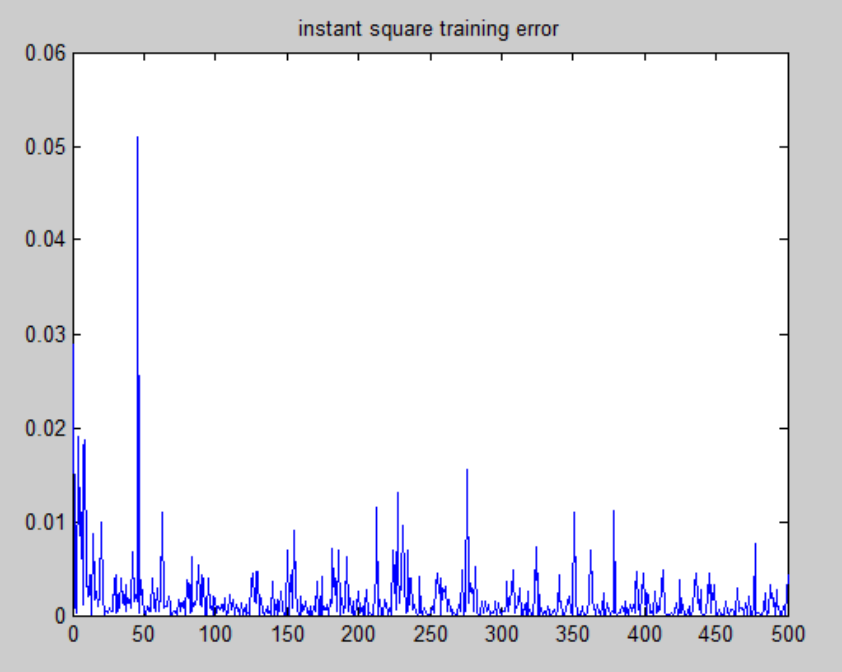
- **10 νευρώνες**

Έχοντας βάλει μόνο 10 νευρώνες στο ESN, τα αποτελέσματα που περιμένουμε δεν είναι τα καλύτερα, καθώς ο αριθμός των νευρώνων είναι πολύ μικρός. Ωστόσο η διαδικασία υπολογισμού είναι πολύ γρήγορη.

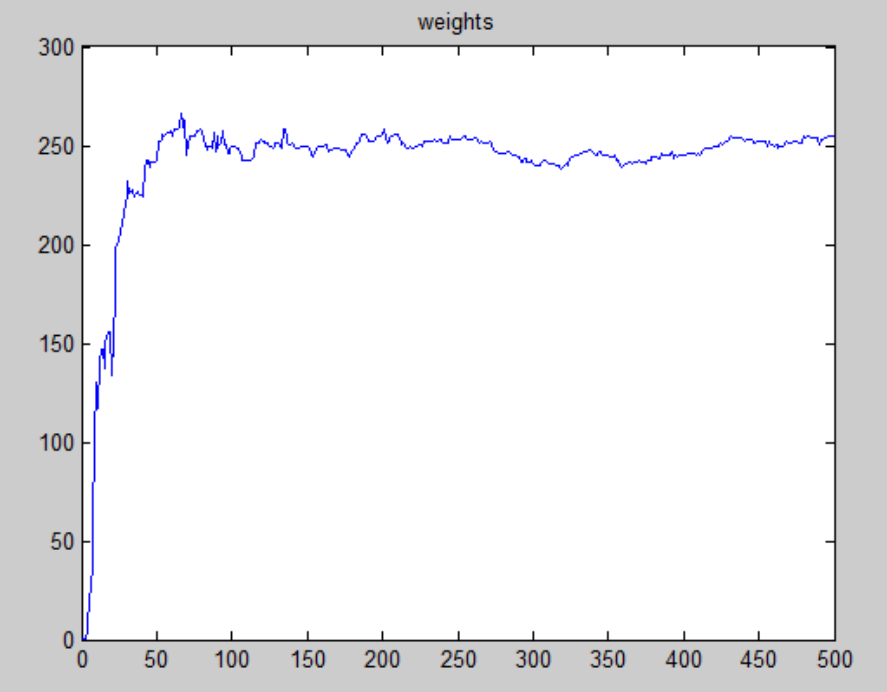
Και πάλι για να πάρουμε τα καλύτερα δυνατά αποτελέσματα, παίρνουμε τον μέσο όρο των αποτελεσμάτων 100 ESN. Παρακάτω βλέπουμε τα αποτελέσματα που παίρνουμε:

```
Generating data .............
Sequence Length 1000
train NRMSE = 3.400364e-001
test NRMSE = 3.820722e-001
Generating data .............
Sequence Length 1000
train NRMSE = 3.940829e-001
test NRMSE = 4.278471e-001
Generating data .............
Sequence Length 1000
train NRMSE = 4.412410e-001
test NRMSE = 4.854044e-001
Generating data .............
Sequence Length 1000
train NRMSE = 3.191777e-001
test NRMSE = 3.449840e-001
Generating data .............
Sequence Length 1000
train NRMSE = 2.894928e-001
test NRMSE = 2.775594e-001
```

Εκτός από τα γραφήματα που αφορούν τη σύγκριση της ακολουθίας εκπαίδευσης με αυτή που έχει προβλεφθεί κατά τη διαδικασία εκπαίδευσης (training) και δοκιμής (testing), μπορούμε να πάρουμε και γραφήματα που απεικονίζουν το μέσο τετραγωνικό σφάλμα εκπαίδευσης και τις τιμές που παίρνουν τα βάρη για τις 500 πρώτες τιμές κάθε φορά που τρέχουμε το κάθε ESN. Τα γραφήματα αυτά φαίνονται παρακάτω:

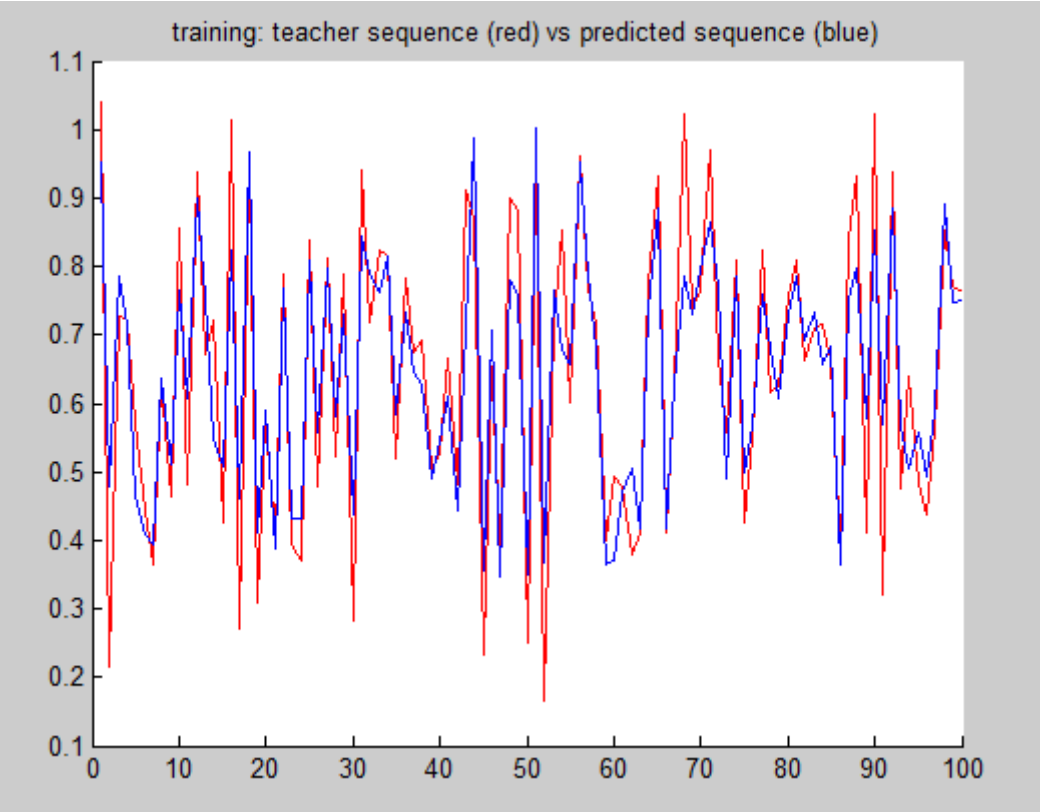


Σχήμα 2.7: Στιγμιαίο τετραγωνικό σφάλμα εκπαίδευσης για τις 500 πρώτες τιμές (10 νευρώνες)

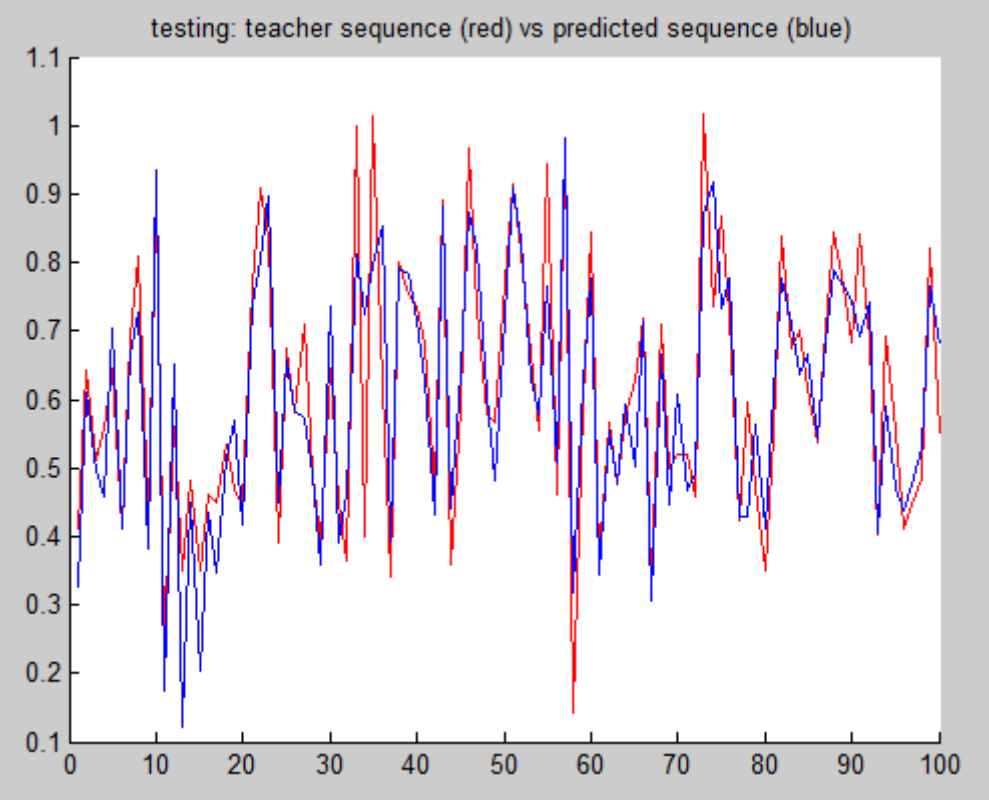


Σχήμα 2.8: Τιμές των βαρών των συνάψεων (10 νευρώνες)

Βλέπουμε ότι γενικά τα σφάλματα παίρνουν αρκετά μεγάλες τιμές ενώ τα βάρη παίρνουν τιμές γύρω από το 250. Ας δούμε όμως τη πρόβλεψη της χρονοσειράς καθώς και τον μέσο όρο των σφαλμάτων



Σχήμα 2.9: Γράφημα ακολουθιών κατά την εκπαίδευση (10 νευρώνες)



Σχήμα 2.10: Γράφημα ακολουθιών κατά τον έλεγχο (10 νευρώνες)

Όπως βλέπουμε από τα παραπάνω γραφήματα, το δίκτυο καταφέρνει να προβλέψει αρκετά καλά τις τιμές τις ακολουθίας αλλά η πρόβλεψη είναι χειρότερη από την αντίστοιχη που έγινε εκπαίδευση από τα δεδομένα. Για καλύτερα συμπεράσματα θα δούμε τα στατιστικά των σφαλμάτων εκπαίδευσης και δοκιμής.

```
mean_tr =

  Columns 1 through 11

    0.3960    0.2624    0.2928    0.6861    0.5343    0.4683    0.4397    0.2924    0.5365    0.6102    0.7755

  Columns 12 through 22

    0.4910    0.6416    0.4997    0.3112    0.4980    0.5654    0.3335    0.4366    0.4896    0.4693    0.4032

  Columns 23 through 33

    0.2764    0.3256    0.6252    0.2976    0.4705    0.5223    0.4877    0.3740    0.3645    0.3319    0.4175
```

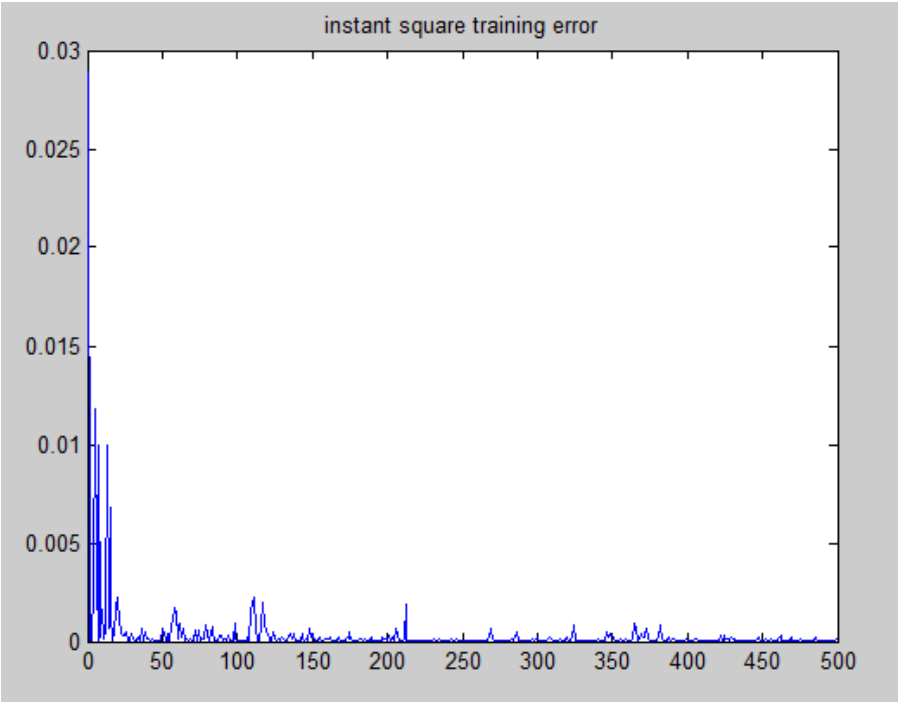
```
mean_tset =

  Columns 1 through 11

    0.4017    0.2863    0.2920    0.7117    0.5431    0.4747    0.4710    0.2908    0.5793    0.6568    0.7752

  Columns 12 through 22

    0.5040    0.6350    0.4890    0.3037    0.5001    0.5629    0.3705    0.4685    0.5267    0.4762    0.4275

  Columns 23 through 33

    0.2807    0.3155    0.6138    0.2970    0.5099    0.5562    0.4895    0.3631    0.4144    0.3550    0.4273
```

```
>> mean (mean_tr)

ans =

    0.4221

>> mean (mean_tset)

ans =

    0.4331
```

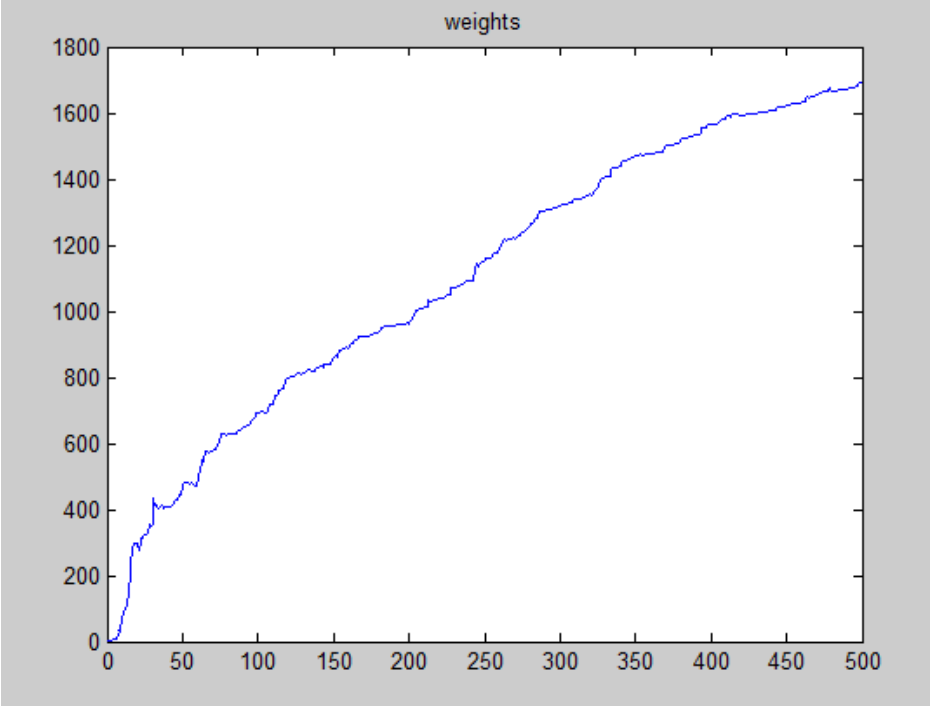
Ο μέσος όρος των σφαλμάτων εκπαίδευσης είναι 0,4221 ενώ ο αντίστοιχος της προηγούμενης μεθόδου ήταν 0,2208. Επίσης όσον αφορά τα σφάλματα ελέγχου, αυτή η μέθοδος μας έδωσε σφάλμα 0,4331 ενώ προηγουμένως είχαμε βρει 0,2281. Παρατηρούμε δηλαδή ότι η online μέθοδος εκπαίδευσης μας δίνει σχεδόν διπλάσιο σφάλμα.

- **300 νευρώνες**

Αυξάνουμε τον αριθμό των νευρώνων σε 300 και ακολουθούμε την ίδια διαδικασία. Τα αποτελέσματα που παίρνουμε είναι τα ακόλουθα:



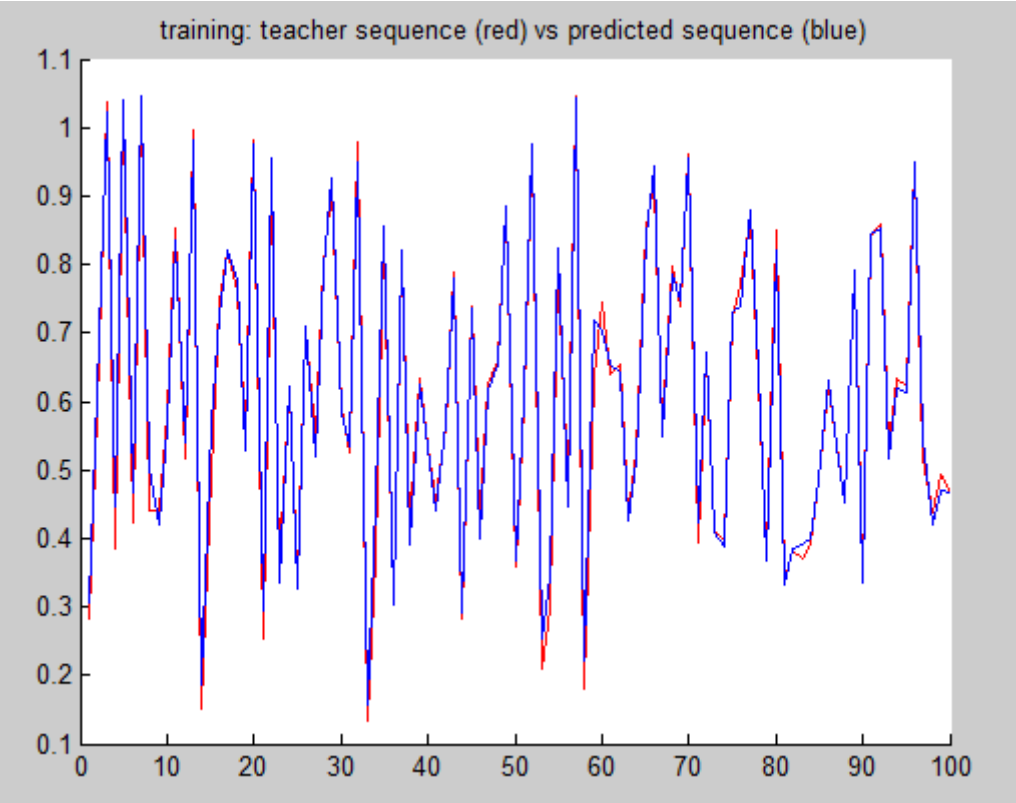
Σχήμα 2.11: Στιγμιαίο τετραγωνικό σφάλμα εκπαίδευσης για τις 500 πρώτες τιμές (300 νευρώνες)



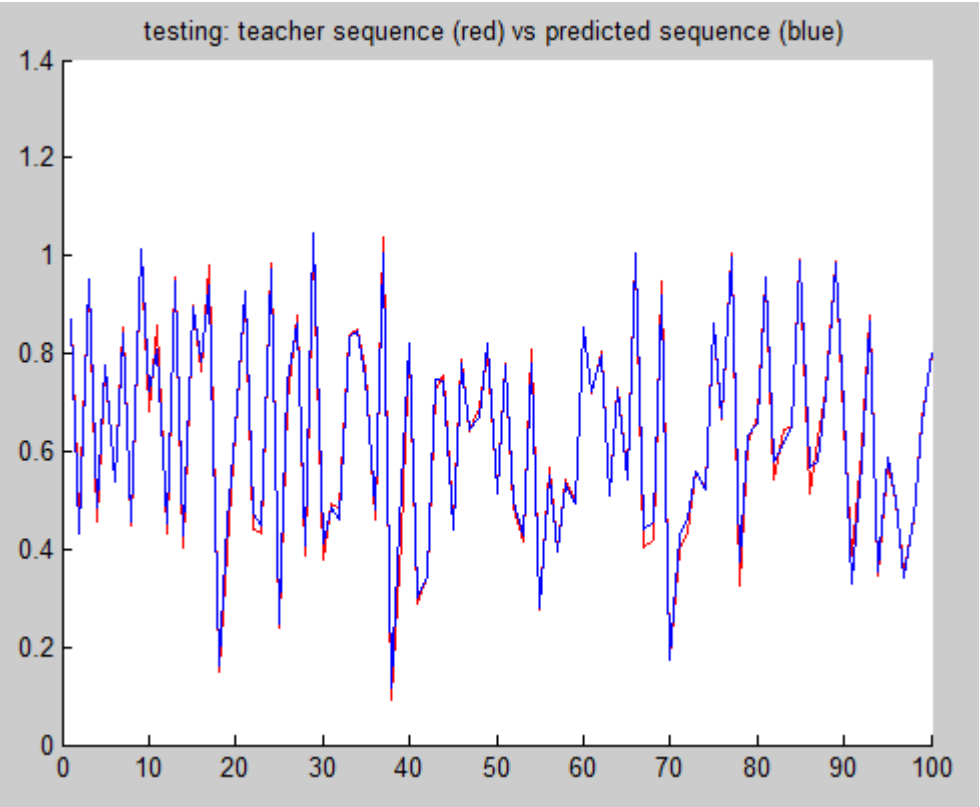
Σχήμα 2.12: Τιμές των βαρών των συνάψεων (300 νευρώνες)

Σε σχέση με την προηγούμενη δοκιμή με τους 10 νευρώνες βλέπουμε ότι τα σφάλματα έχουν μειωθεί σημαντικά και πλέον οι τιμές των βαρών των συνάψεων αυξάνεται σταδιακά μέχρι το 1700.

Τα διαγράμματα που φαίνονται παρακάτω μας δείχνουν και πάλι πολύ καλύτερη πρόβλεψη, όπως ήταν αναμενόμενο. Επίσης να σημειώσουμε ότι μόνο από τα γραφήματα δεν μπορούμε να κάνουμε απ ευθείας σύγκριση των δύο μεθόδων εκμάθησης, όπως έγινε στους 10 νευρώνες, καθώς και τα δύο έχουν κάνει πολύ καλή πρόβλεψη. Γι αυτό το λόγο θα πρέπει να κοιτάξουμε τα αναλυτικά στατιστικά των σφαλμάτων.



Σχήμα 2.13: Γράφημα ακολουθιών κατά την εκπαίδευση (300 νευρώνες)



Σχήμα 2.14: Γράφημα ακολουθιών κατά τον έλεγχο (300 νευρώνες)

Τα αποτελέσματα και τα στατιστικά φαίνονται παρακάτω:

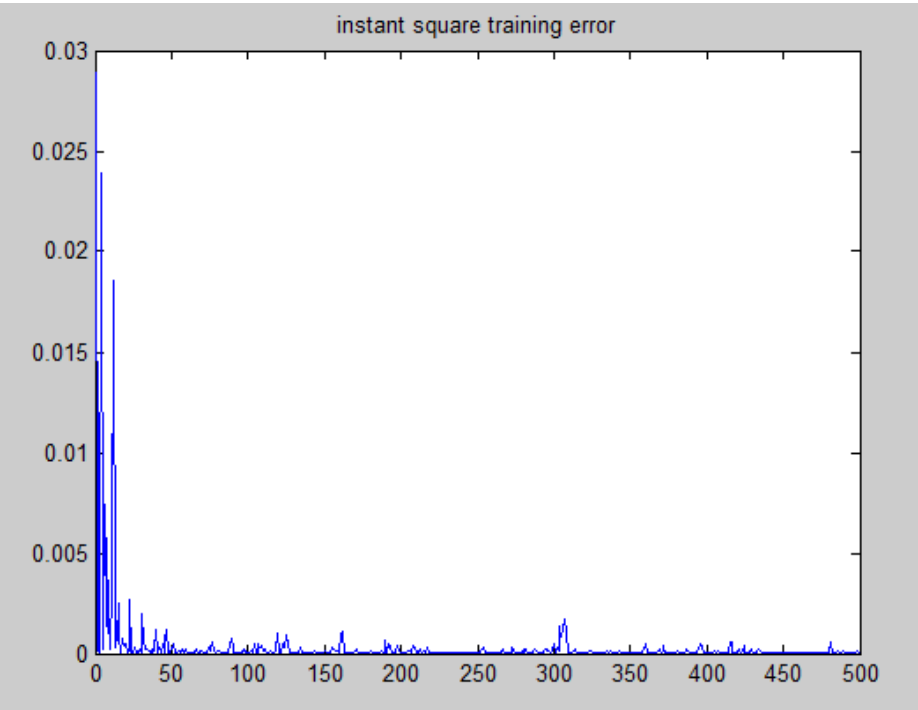
```
mean_tr =

  Columns 1 through 11

    0.1240    0.1155    0.1259    0.1123    0.1120    0.1100    0.1114    0.1302    0.1237    0.1296    0.1091

  Columns 12 through 22

    0.0989    0.1084    0.1004    0.1192    0.1028    0.1145    0.1218    0.1115    0.1082    0.1206    0.1021

  Columns 23 through 33

    0.1043    0.1114    0.1099    0.1340    0.1070    0.1263    0.1087    0.1110    0.1082    0.1188    0.1153

mean_tset =

  Columns 1 through 11

    0.1075    0.1311    0.1334    0.1091    0.1218    0.1288    0.1231    0.1519    0.1217    0.1343    0.1497

  Columns 12 through 22

    0.1246    0.1286    0.1199    0.1194    0.1104    0.1280    0.1189    0.1093    0.1204    0.1166    0.1267

  Columns 23 through 33

    0.1339    0.1351    0.1189    0.1436    0.1289    0.1384    0.1165    0.1332    0.1201    0.1371    0.1202
```

```
>> mean (mean_tr)

ans =

    0.1167

>> mean (mean_tset)

ans =

    0.1269
```

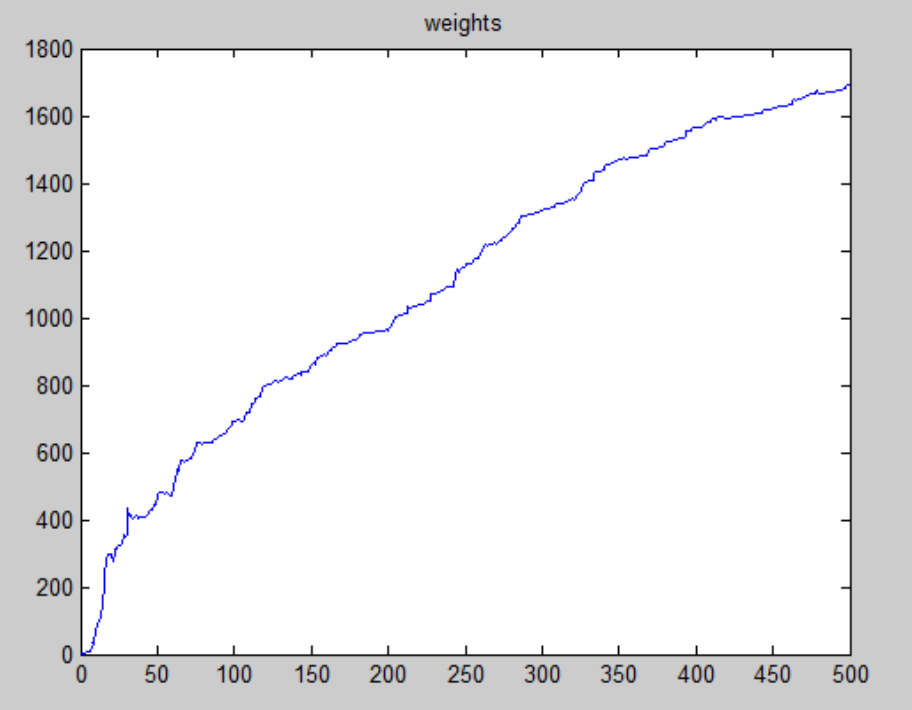
Ο μέσος όρος των σφαλμάτων εκπαίδευσης είναι 0,1167 και ο μέσος όρος σφαλμάτων δοκιμής 0,1269. Παρόλο που τα σφάλματα μειώθηκαν πολύ σε σχέση με τους 10 νευρώνες, και πάλι βλέπουμε ότι η προηγούμενη μέθοδος είχε καλύτερα αποτελέσματα με σφάλματα 0,0175 και 0,904 αντίστοιχα.

- **500 νευρώνες**

Αυξάνουμε τους νευρώνες σε 500 και θέτουμε το ESN σε λειτουργία. Τα αποτελέσμτα που παίρνουμε σχετικά με το τετραγωνικό σφάλμα και τα βάρη είναι παρόμοια με τα αποτελέσματα που πήραμε για τους 300 νευρώνες.

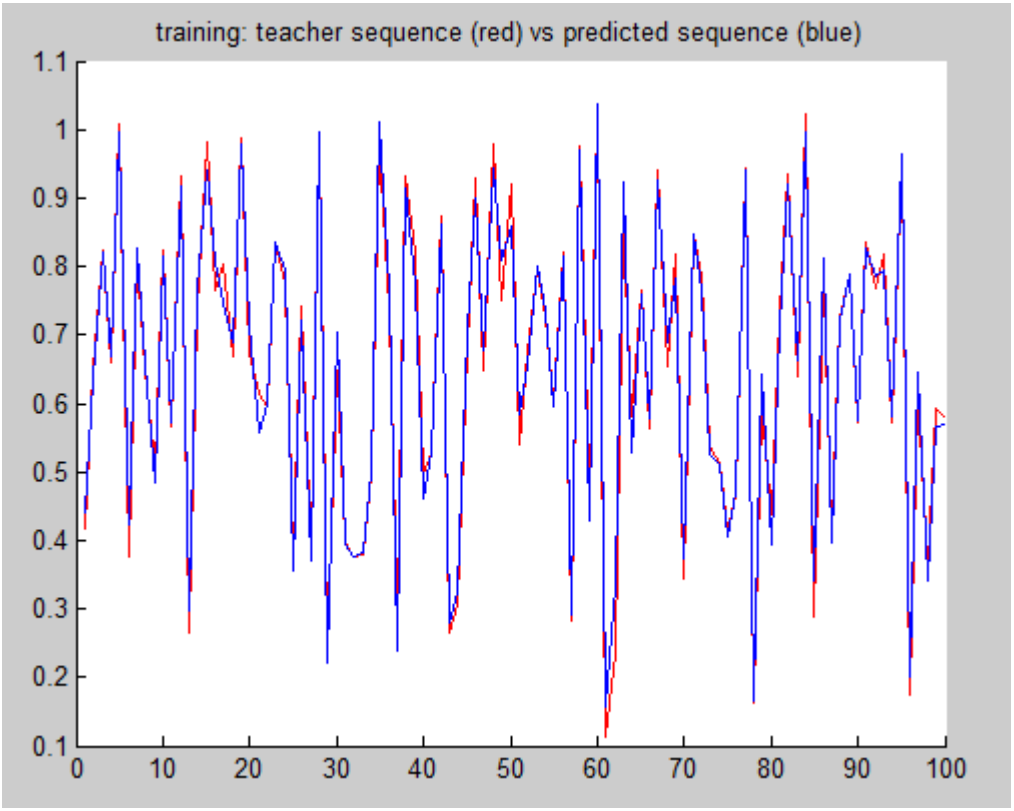


Σχήμα 2.15: Στιγμιαίο τετραγωνικό σφάλμα εκπαίδευσης για τις 500 πρώτες τιμές (500 νευρώνες)

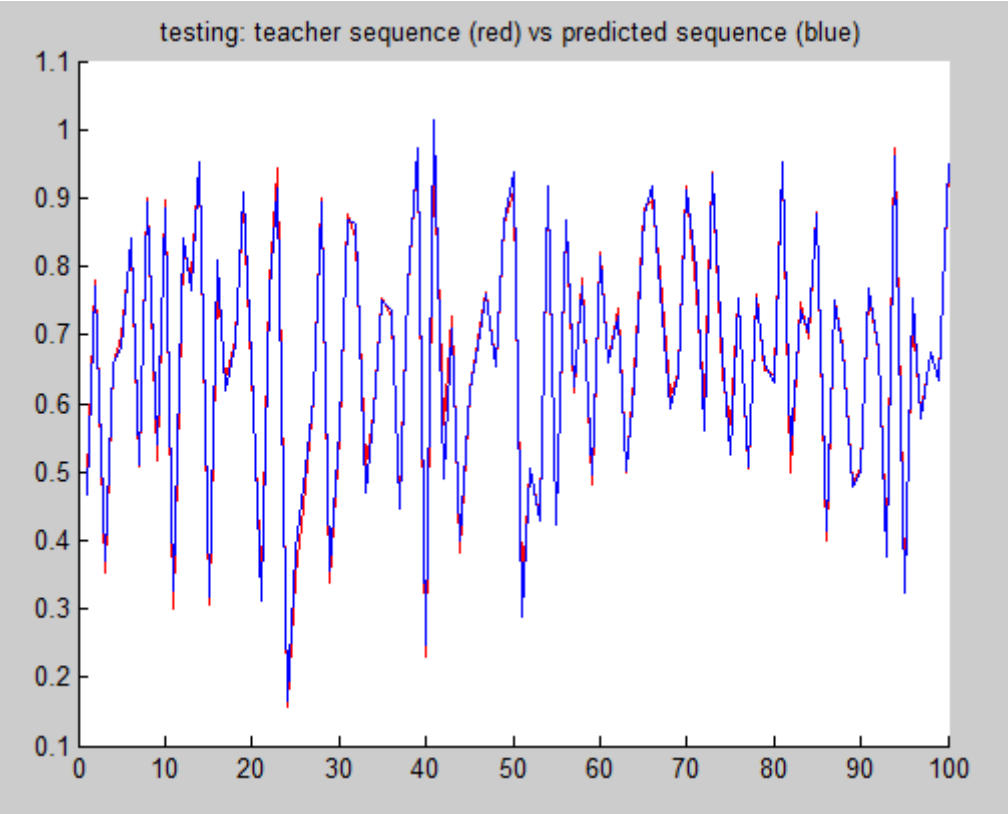


Σχήμα 2.16: Τιμές των βαρών των συνάψεων (300 νευρώνες)

Πλέον βλέπουμε και πάλι ότι η πρόβλεψη είναι πάρα πολύ καλή, όπως είναι αναμενόμενο, όμως για άλλη μια φορά δεν φτάνει σε επίδοση το ESN, που έχει εκπαιδευθεί από το σύνολο των δεδομένων.



Σχήμα 2.17: Γράφημα ακολουθιών κατά την εκπαίδευση (500 νευρώνες)



Σχήμα 2.18: Γράφημα ακολουθιών κατά την δοκιμή (500 νευρώνες)

Παρακάτω βλέπουμε κάποια από τα αποτελέσματα που μας δίνει το ESN, καθώς και τα στατιστικά των σφαλμάτων.

```
mean_tr =

  Columns 1 through 11

    0.1015    0.0979    0.1091    0.0955    0.1100    0.1007    0.1117    0.1126    0.1072    0.1075    0.1111

  Columns 12 through 22

    0.0973    0.0942    0.1099    0.1067    0.0992    0.1062    0.1089    0.1102    0.1165    0.1069    0.1130

  Columns 23 through 33

    0.1161    0.0951    0.1032    0.1019    0.0986    0.1033    0.1016    0.1012    0.1088    0.1080    0.1135

mean_tset =

  Columns 1 through 11

    0.1126    0.1109    0.1231    0.1184    0.1235    0.1212    0.1056    0.1236    0.1136    0.1148    0.1220

  Columns 12 through 22

    0.1156    0.1147    0.1199    0.1076    0.1171    0.1134    0.1154    0.1132    0.1017    0.1160    0.1452

  Columns 23 through 33

    0.1170    0.1289    0.1166    0.1226    0.1175    0.1087    0.1389    0.1153    0.1128    0.1323    0.1106
```

Ο μέσος όρος του σφάλματος εκπαίδευσης βλέπουμε ότι είναι 0,105 και ο μέσος όρος των σφαλμάτων ελέγχου 0,1154. Για τους 300 νευρώνες είχαμε ότι ο μέσος όρος των σφαλμάτων εκπαίδευσης ήταν 0,1167 και ο μέσος όρος σφαλμάτων δοκιμής 0,1269. Δηλαδή βλέπουμε ότι παρ όλο που η πολυπλοκότητα του νευρωνικού μας δικτύου αυξήθηκε πολύ, το σφάλμα μειώθηκε ελάχιστα.

Επίσης κάνοντας τη σύγκριση με τη μέθοδο εκμάθησης από το σύνολο δεδομένων και πάλι βλέπουμε ότι τα αποτελέσματα που παίρνουμε είναι χειρότερα, αφού το σφάλμα εκπαίδευσης ήταν σχεδόν μηδενικό και το σφάλμα ελέγχου παρόμοιο με αυτό που βρήκαμε σ αυτή τη περίπτωση.

Τέλος, να προσθέσουμε ότι ο χρόνος επεξεργασίας της χρονοσειράς ήταν πολύ μεγάλος, γεγονός που δυσκόλεψε πολύ την εξαγωγή αποτελεσμάτων. Καταλήγουμε λοιπόν ότι το ESN με τους 500 νευρώνες δεν είναι καθόλου εύχρηστο και δεν θα προτιμηθεί στη συνέχεια.

```
>> mean (mean_tr)

ans =

    0.1050

>> mean (mean_tset)

ans =

    0.1154
```

2.3.2 Συσχέτιση ποσοστού δεδομένων για εκπαίδευση- σφαλμάτων

Για να κλείσουμε αυτή την ενότητα, θα κάνουμε μια σύγκριση των σφαλμάτων που λαμβάνουμε χρησιμοποιώντας το ESN των 300 νευρώνων αλλάζοντας όμως κάθε φορά το ποσοστό των δεδομένων που χρησιμοποιούνται για την εκπαίδευση .

Αρχικά θα χρησιμοποιήσουμε το 30% των δεδομένων, στη συνέχεια το 50% και τέλος, το 80%.

- **Ποσοστό εκπαίδευσης: 30%**

```
>> mean(mean_tr)

ans =

    0.1225

>> mean(mean_tset)

ans =

    0.1406
```

- **Ποσοστό εκπαίδευσης: 50%**

```
>> mean (mean_tr)

ans =

    0.1167

>> mean (mean_tset)

ans =

    0.1269
```

- **Ποσοστό εκπαίδευσης: 80%**

```
mean(mean_tr)

ans =

    0.1103

>> mean(mean_tset)

ans =

    0.1129
```

Όπως είναι αναμενόμενο όσο μεγαλώνει το ποσοστό της εκπαίδευσης, τόσο μειώνεται και το σφάλμα ελέγχου. Με το 30% των δεδομένων στην εκπαίδευση, το σφάλμα είναι κατά μέσο όρο γύρω στο 0,14, στη συνέχεια με 50% εκπαίδευση έχουμε σφάλμα 0,1269 , ενώ με 80% εκπαίδευση, το σφάλμα μειώνεται ελάχιστα, στο 0,1129

Με την προηγούμενη μέθοδο, ξεκινώντας από 0.14, το σφάλμα είχε πέσει στο 0.09 και στη συνέχεια στο 0.06 περίπου, δηλαδή και πάλι συμπεραίνουμε ότι η μέθοδος εκμάθησης από το σύνολο δεδομένων είναι καλύτερη όσον αφορά το συγκεκριμένο τύπο νευρωνικών δικτύων.

Στη συνέχεια θα μελετήσουμε κατά πόσον η πολυπλοκότητα του reservoir επηρεάζει τα αποτελέσματα και τα σφάλματα εξόδου. Αυτό γίνεται με τον υπολογισμό των εκθετών Lyapunov, οι οποίοι μας καθορίζουν κατά πόσον το σύστημά μας είναι χαοτικό και αν αυτό ισχύει, πόσο χαοτικό είναι.

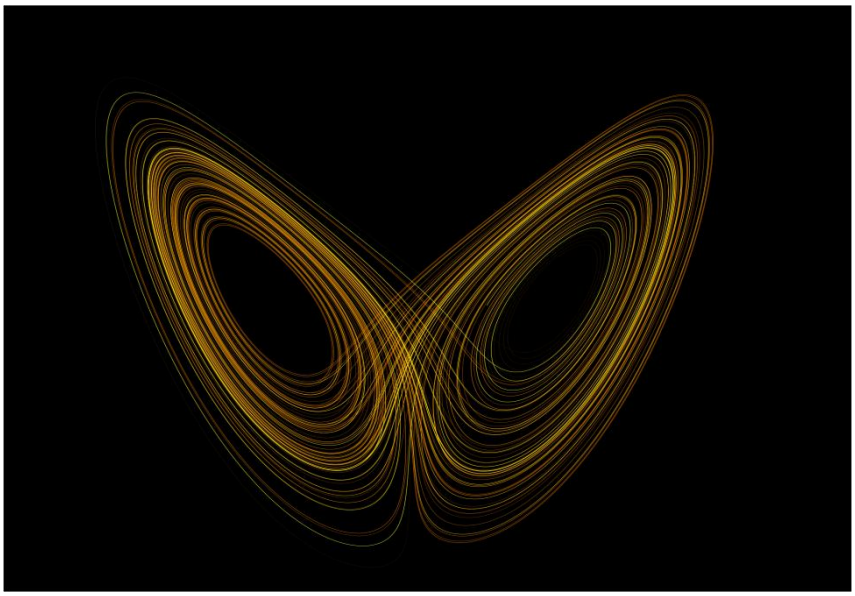
3. Οι εκθέτες Lyapunov στον προσδιορισμό χαοτικών συστημάτων

3.1 Γενικά

Η **Θεωρία του Χάους** είναι ένας τομέας στα μαθηματικά, με διάφορες εφαρμογές σε κλάδους επιστημών όπως η φυσική, η μηχανολογία, τα οικονομικά και η βιολογία. Η θεωρία του Χάους μελετά τη **συμπεριφορά ορισμένων μη γραμμικών δυναμικών συστημάτων, που είναι ιδιαίτερα ευαίσθητα στις αρχικές συνθήκες**. Μικρές διαφορές στις αρχικές συνθήκες (όπως αυτές που οφείλονται σε σφάλματα στρογγυλοποίησης σε αριθμητικούς υπολογισμούς) αποδίδουν πολύ διαφορετικά αποτελέσματα για τα δυναμικά συστήματα, καθιστώντας τη μακροπρόθεσμη πρόβλεψη αδύνατη σε γενικές γραμμές. Αυτό συμβαίνει παρ' όλο που αυτά τα συστήματα είναι αιτιοκρατικά (ντετερμινιστικά), πράγμα που σημαίνει ότι η μελλοντική συμπεριφορά τους καθορίζεται πλήρως από τις αρχικές συνθήκες τους, χωρίς να εμπλέκονται τυχαίες παράμετροι. Με άλλα λόγια, η ντετερμινιστική φύση αυτών των συστημάτων δεν τα κάνει προβλέψιμα. Αυτή η συμπεριφορά είναι γνωστή ως ντετερμινιστικό χάος, ή απλά χάος. Αυτό συνοψίζεται από τον Έντουαρντ Λόρεντζ ως εξής:

Χάος: Όταν το παρόν καθορίζει το μέλλον, αλλά η προσέγγιση του παρόντος δεν προσδιορίζει κατά προσέγγιση το μέλλον

Χαοτική συμπεριφορά μπορεί να παρατηρηθεί σε πολλά φυσικά συστήματα, όπως ο καιρός η ατμόσφαιρα, το ηλιακό σύστημα, οι τεκτονικές πλάκες, τα οικονομικά συστήματα και η εξέλιξη (μεταβολή) των πληθυσμών.



Σχήμα 3.1: Ο παράξενος ελκυστής της εξίσωσης Lorentz είναι χαρακτηριστικό παράδειγμα χαοτικού δυναμικού συστήματος.

Αν και δεν υπάρχει καθολικά αποδεκτός μαθηματικός ορισμός του χάους, ένας κοινά αποδεκτός ορισμός λέει ότι, για να χαρακτηριστεί η συμπεριφορά ενός συστήματος ως χαοτική, πρέπει να έχει τις ακόλουθες ιδιότητες:

i. Πρέπει να παρουσιάζει **ευαίσθητη εξάρτηση από τις αρχικές συνθήκες**

Ευαισθησία στις αρχικές συνθήκες σημαίνει ότι το κάθε σημείο σε ένα τέτοιο σύστημα είναι αυθαίρετα στενά προσεγγίσιμο από άλλα σημεία με σημαντικά διαφορετικές μελλοντικές τροχιές. Έτσι, μια αυθαίρετα μικρή διαταραχή της τρέχουσας τροχιάς μπορεί να οδηγήσει σε σημαντικά διαφορετική μελλοντική συμπεριφορά.

Μια συνέπεια της ευαισθησίας στις αρχικές συνθήκες είναι ότι αν αρχίσουμε με μόνο μια πεπερασμένη ποσότητα πληροφοριών σχετικά με το σύστημα (όπως γίνεται συνήθως στην πράξη), τότε, πέρα από ένα ορισμένο χρονικό διάστημα, το σύστημα δεν θα είναι πια προβλέψιμο. Αυτό είναι πλέον οικείο στην πρόβλεψη του καιρού, η οποία είναι γενικώς δυνατή μόνο περίπου μια εβδομάδα πριν.

Ο **εκθέτης Lyapunov** χαρακτηρίζει την **έκταση της ευαισθησίας στις αρχικές συνθήκες**. Ποσοτικώς, δύο τροχιές εντός του χώρου φάσης με αρχικό διαχωρισμό $\delta \mathbf{Z}_0$ αποκλίνουν ως εξής:

$$| \delta \boldsymbol{Z}_{(t)} | \approx e^{\lambda t} | \delta \boldsymbol{Z}_0 |$$

όπου λ είναι ο εκθέτης Lyapunov. Ο ρυθμός διαχωρισμού μπορεί να είναι διαφορετικός για διαφορετικούς προσανατολισμούς του αρχικού φορέα διαχωρισμού. Έτσι, υπάρχει ένα ολόκληρο φάσμα από εκθέτες Lyapunov - ο αριθμός των οποίων είναι ίσος με τον αριθμό των διαστάσεων του χώρου φάσης. Είναι σύνηθες να αναφέρεται μόνο ο μεγαλύτερος, δηλαδή ο μέγιστος εκθέτης Lyapunov (ΜΕΛ), διότι αυτός καθορίζει τη συνολική προβλεψιμότητα του συστήματος. **Ένας θετικός μέγιστος εκθέτης Lyapunov συνήθως λαμβάνεται ως ένδειξη ότι το σύστημα είναι χαοτικό**.

ii. Πρέπει να είναι **τοπολογικά μεταβατικό**

Τοπολογική μεταβατικότητα (ή τοπολογική ανάμειξη), σημαίνει ότι το σύστημα θα εξελιχθεί με την πάροδο του χρόνου, έτσι ώστε κάθε συγκεκριμένη περιοχή ή ανοιχτό σύνολο του χώρου φάσης **τελικά θα συμπίπτει με οποιαδήποτε άλλη περιοχή**. Αυτή η μαθηματική έννοια της "ανάμειξης" αντιστοιχεί στην κοινή διαίσθηση, και η ανάμειξη των έγχρωμων βαφών ή υγρών είναι ένα παράδειγμα ενός χαοτικού συστήματος.

iii. Πρέπει να εμφανίζει ένα **πυκνό σύνολο που αποτελείται από όλες τις περιοδικές τροχιές του συστήματος**

Η πυκνότητα των περιοδικών τροχιών σημαίνει ότι **κάθε σημείο στο χώρο προσεγγίζεται αυθαίρετα στενά από περιοδικές τροχιές**.

Όπως αναφέραμε και προηγουμένως, υπολογίζοντας τους εκθέτες Lyapunov, μπορούμε να βγάλουμε συμπεράσματα σχετικά με το κατά πόσον η χρονοσειρά που μελετάμε είναι χαοτική, καθώς και πόσο χαοτική είναι. Στόχος λοιπόν της ενότητας αυτής είναι να υπολογίσουμε τους εκθέτες αυτούς και να καταλήξουμε σε συμπεράσματα σχετικά με την χαοτικότητα της χρονοσειράς μας και το αντίστοιχο σφάλμα που αποδίδει κατά τη πρόβλεψή της με το ESN.

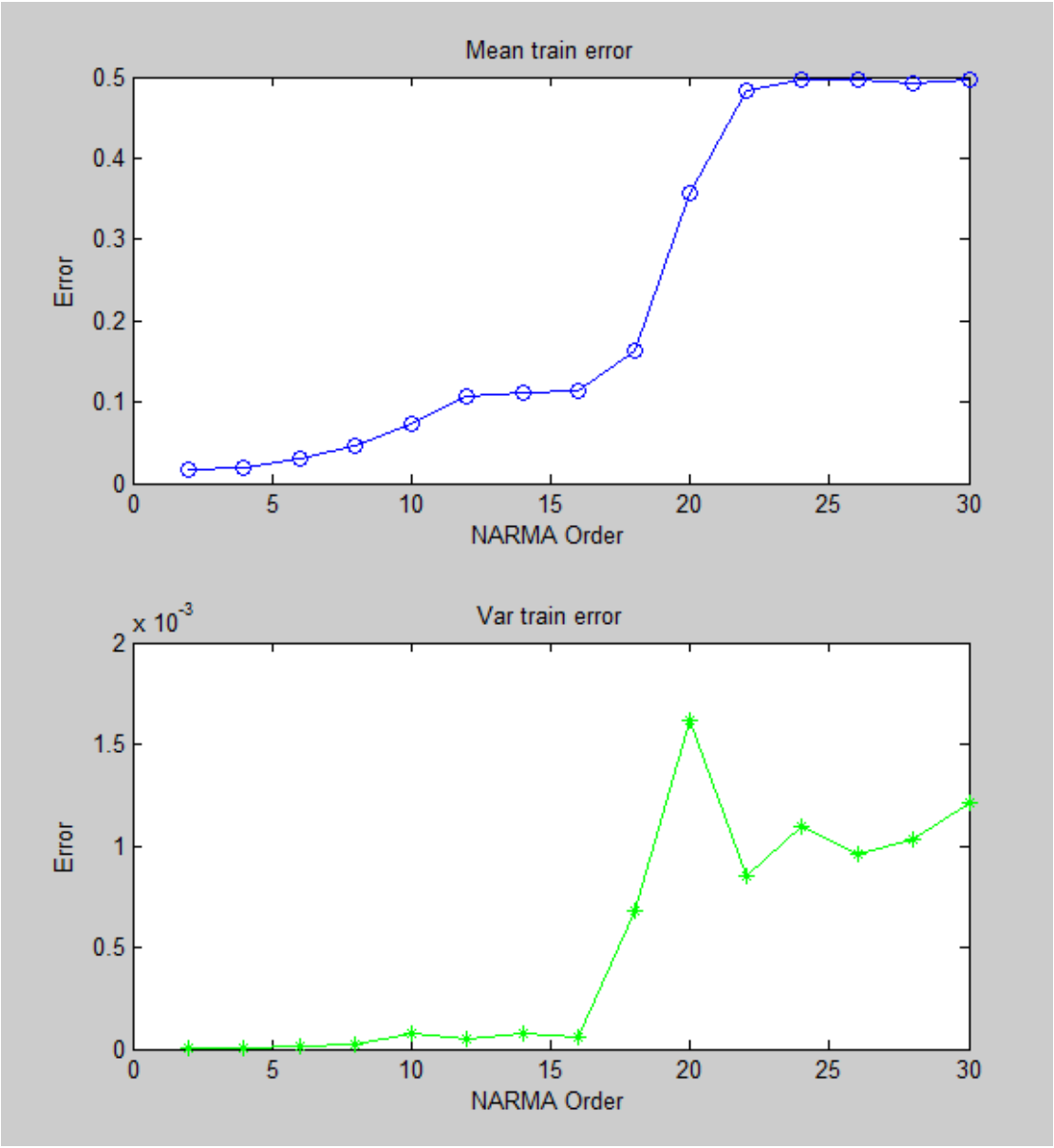
3.2 Σύγκριση πολυπλοκότητας – σφαλμάτων

Αρχικά θα δούμε μια σύγκριση της πολυπλοκότητας της χρονοσειρά μας και των σφαλμάτων που δημιουργούν. Συγκεκριμένα, θα μελετήσουμε χρονοσειρές NARMA τάξης 2,4,…,30 και θα δούμε πως αυξάνεται το σφάλμα εκπαίδευσης και ελέγχου ανάλογα με την τάξη της χρονοσειράς μας. Η τάξη της χρονοσειράς αναφέρεται στον αριθμό των εξαρτήσεων από τις προηγούμενες τιμές. Για παράδειγμα, για μια σειρά NARMA τάξης 5, η έκτη παρατήρησή της εξαρτάται από τις προηγούμενες πέντε. Αντιλαμβανόμαστε λοιπόν ότι καθώς αυξάνεται η τάξη της χρονοσειράς, αυξάνεται και η πολυπλοκότητά της.

Για τη σύγκριση αυτή θα χρησιμοποιήσουμε και πάλι το Matlab. Αρχικά δημιουργώ 15 διαφορετικά ESN, καθένα από τα οποία λειτουργεί με μια χρονοσειρά NARMA διαφορετικής τάξης. Για κάθε ESN υπολογίζω το σφάλμα εκπαίδευσης και ελέγχου 100 χρονοσειρών NARMA και βρίσκω το μέσο όρο των σφαλμάτων και τη διασπορά τους, για να έχω πιο αμερόληπτα αποτελέσματα. Τέλος, κατασκευάζω διαγράμματα για να έχουμε καλύτερη εικόνα των αποτελεσμάτων.

Ο κώδικας για την κατασκευή των χρονοσειρών και των διαγραμμάτων βρίσκεται στο παράρτημα.

Παρακάτω βλέπουμε τα αποτελέσματα που παίρνουμε:

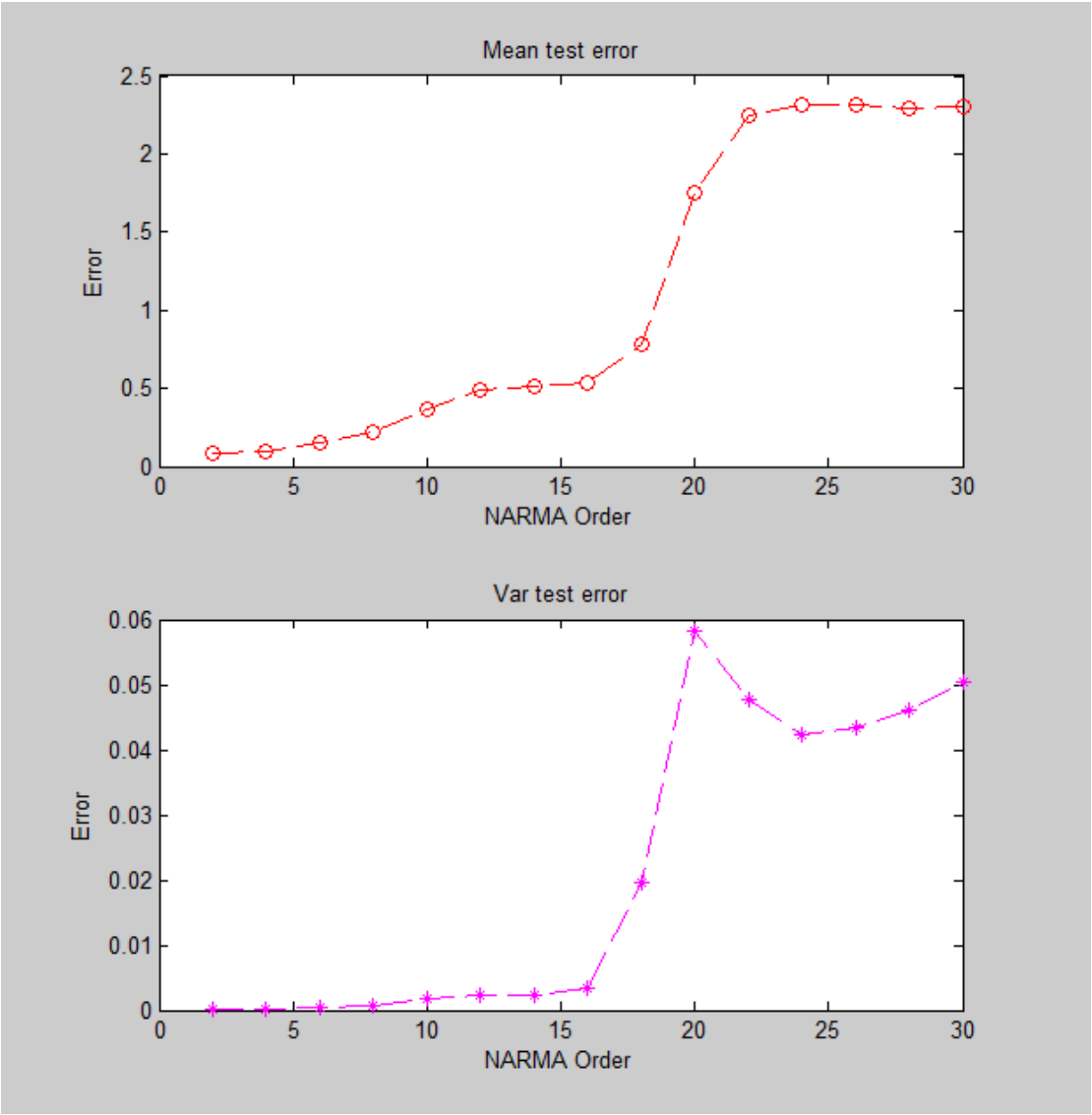


Σχήμα 3.2: Μέση τιμή και διασπορά σφαλμάτων εκπαίδευσης για τα 15 ESN

Παρατηρούμε ότι η τιμή του σφάλματος αυξάνεται σταδιακά καθώς αυξάνεται και η τάξη της χρονοσειράς, δηλαδή η πολυπλοκότητά της. Αξίζει να σημειώσουμε την απότομη αύξηση του σφάλματος στις χρονοσειρές τάξης 18, 20 και 22 και την εξομάλυνση του πάλι μέχρι την τάξη 30.

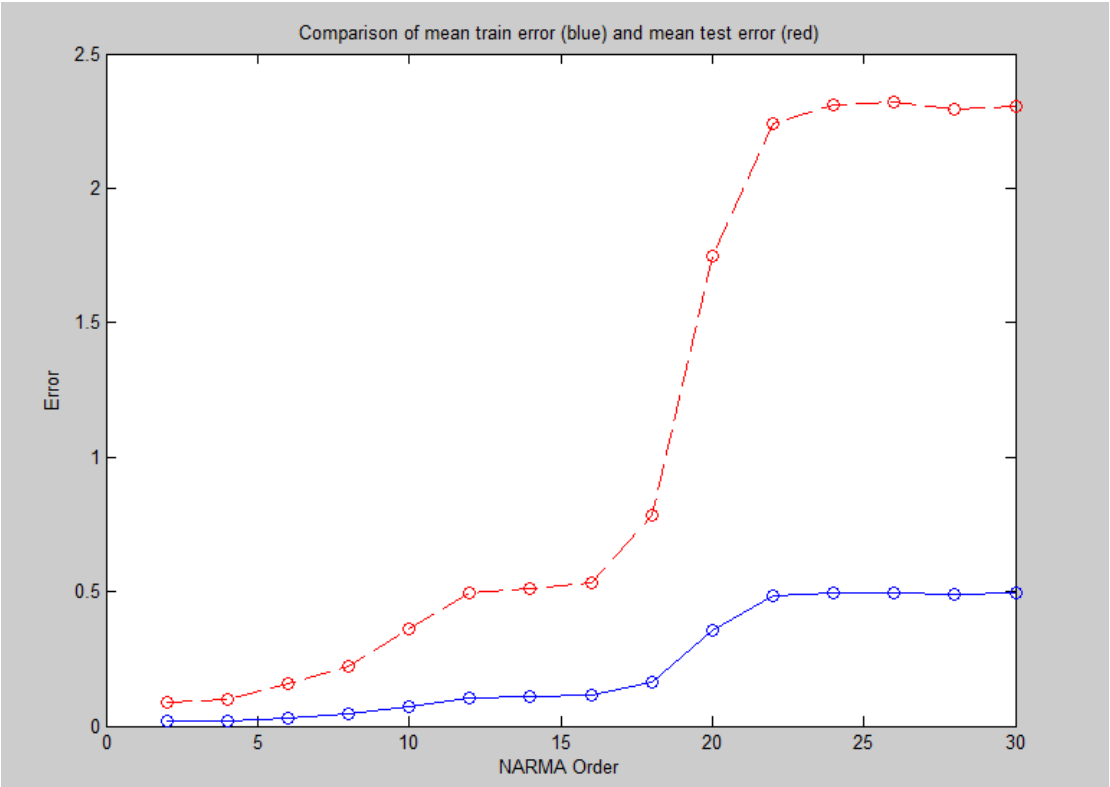
Όσον αφορά τη διασπορά των σφαλμάτων, βλέπουμε ότι κυμαίνεται σε πολύ μικρές τιμές.

Στα διαγράμματα που ακολουθούν βλέπουμε τα αντίστοιχα διαγράμματα και για το σφάλμα ελέγχου. Τα συμπεράσματα είναι ίδια και εδώ, δηλαδή όπως είναι αναμενόμενο, το σφάλμα αυξάνεται με την αύξηση της τάξης της χρονοσειράς μας.



Σχήμα 3.3: Μέση τιμή και διασπορά σφαλμάτων ελέγχου για τα 15 ESN

Στη συνέχεια θα δούμε το κοινό διάγραμμα των μέσων όρων των σφαλμάτων εκπαίδευσης και ελέγχου. Με κόκκινη διακεκομμένη γραμμή συμβολίζεται το σφάλμα ελέγχου, ενώ με μπλε το σφάλμα εκπαίδευσης.



Σχήμα 3.4: Σύγκριση μέσης τιμής σφαλμάτων εκπαίδευσης και ελέγχου

Παρατηρούμε ότι όποια και να είναι η τάξη της χρονοσειράς, το σφάλμα ελέγχου είναι πάντα μεγαλύτερο από το σφάλμα εκπαίδευσης. Το μεγαλύτερο εύρος της διαφοράς των δύο σφαλμάτων παρατηρείται για χρονοσειρές τάξης μεγαλύτερης από 18.

3.3 Υπολογισμός εκθετών Lyapunov

3.3.1 Γενικά

Για ένα δυναμικό σύστημα, **η ευαισθησία στις αρχικές συνθήκες ποσοτικοποιείται από τους εκθέτες Lyapunov**. Για παράδειγμα, ας θεωρήσουμε δύο τροχιές ενός ελκυστή με γειτονικές αρχικές συνθήκες. Όταν ο ελκυστής είναι χαοτικός, οι τροχιές αποκλίνουν, κατά μέσο όρο, με μια εκθετική απόκλιση που χαρακτηρίζεται από το μεγαλύτερο εκθέτη Lyapunov. Αυτή η έννοια επίσης γενικεύεται για το φάσμα των εκθετών Lyapunov, λ_i (i = 1, 2, ..., n), αν θεωρήσουμε μια μικρή n-διάσταστη σφαίρα αρχικών συνθηκών, όπου n είναι ο αριθμός των εξισώσεων που χρησιμοποιούνται για να περιγράψουν το σύστημα.

Καθώς ο χρόνος (t) προχωρά, η σφαίρα εξελίσσεται σε ένα ελλειψοειδές του οποίου οι κύριοι άξονες του διαστέλλονται (ή συστέλλονται) με ρυθμούς που καθορίζονται από τους εκθέτες Lyapunov.

Η παρουσία ενός θετικού εκθέτη είναι επαρκής για τη διάγνωση χάους και αντιπροσωπεύει τη **τοπική αστάθεια** σε μια συγκεκριμένη κατεύθυνση. Γενικά για την ύπαρξη ελκυστή, η συνολική δυναμική πρέπει να είναι συνολικά σταθερή, δηλαδή το συνολικό ποσοστό της συστολής πρέπει να μην υπερκαλύπτει το συνολικό ποσοστό της διαστολής. Έτσι, ακόμη και όταν υπάρχουν αρκετοί θετικοί εκθέτες Lyapunov, το άθροισμα σε όλο το φάσμα είναι αρνητικό.

Ο Wolf και οι συνεργάτες του, στο « Determining Lyapunov exponents from a time series», Physica D 16 (1985) μας εξηγεί το φάσμα των εκθετών Lyapunov γεωμετρικά ως εξής:

Αρχικά κατατάσσουμε τους n κύριους άξονες με τέτοια σειρά ώστε να ξεκινάμε από το **πιο γρήγορα διαστελλόμενο** και να καταλήγουμε στον **πιο γρήγορα συστελλόμενο**. Επομένως οι εκθέτες θα είναι διατεταγμένοι με την εξής σειρά $\boldsymbol{\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n}$ όπου τα $\boldsymbol{\lambda_1}$ και $\boldsymbol{\lambda_n}$ αντιστοιχούν στους πιο γρήγορα διαστελλόμενους και συστελλόμενους κύριους άξονες, αντίστοιχα.

Στη συνέχεια παρατηρούμε ότι το μήκους του πρώτου κύριου άξονα είναι ανάλογο προς $\boldsymbol{e^{\lambda 1}}^{t}$, η περιοχή που καθορίζεται από τους δύο πρώτους κύριους άξονες είναι ανάλογη του $\boldsymbol{e^{(\lambda 1 + \lambda 2)t}}$ και ο όγκος που καθορίζεται από τους k κύριους άξονες είναι ανάλογος με $\boldsymbol{e^{(\lambda 1 + \lambda 2 + \ldots + \lambda k)\, t}}$

Έτσι, το φάσμα Lyapunov μπορεί να οριστεί έτσι ώστε η εκθετική ανάπτυξη ενός στοιχείου k-όγκου να δίνεται από το άθροισμα των k μεγαλύτερων εκθετών Lyapunov. Να σημειώσουμε ότι η πληροφορία που δημιουργείται από το σύστημα παριστάνεται ως μεταβολή του όγκου που ορίζεται από τους διαστελλόμενους κύριους άξονες.

Όταν οι εξισώσεις που περιγράφουν το δυναμικό σύστημα είναι διαθέσιμες, μπορούμε να υπολογίσουμε όλο το φάσμα Lyapunov με τη βοήθεια υπολογιστή. Η μέθοδος αυτή περιλαμβάνει αριθμητική επίλυση n εξισώσεων του συστήματος για n + 1 κοντινές αρχικές συνθήκες. Μετράται η ανάπτυξη ενός αντίστοιχου συνόλου διανυσμάτων, και καθώς το σύστημα εξελίσσεται, τα διανύσματα συνεχώς ορθοκανονικοποιούνται χρησιμοποιώντας τη διαδικασία Gram-Schmidt.

Αυτό εγγυάται ότι μόνο ένα διάνυσμα έχει συνιστώσα στην κατεύθυνση της πιο γρήγορης συστολής, δηλαδή, οι φορείς διατηρούν τον κατάλληλο προσανατολισμό στον χώρο. Σε πειραματικές συνθήκες, ωστόσο, οι εξισώσεις της κίνησης είναι συνήθως άγνωστες και η προσέγγιση αυτή δεν είναι εφαρμόσιμη.

Όπως αναφέρθηκε και προηγουμένως, δεν μπορούμε να υπολογίσουμε όλο το φάσμα Lyapunov, επιλέγοντας αυθαίρετες κατευθύνσεις για να μετρήσουμε τον διαχωρισμού των κοντινών αρχικών συνθηκών. Θα πρέπει να μετρήσουμε το διαχωρισμό κατά μήκος των κατευθύνσεων Lyapunov που αντιστοιχούν στους κύριους άξονες του ελλειψοειδούς που αναφέρθηκε προηγουμένως. Αυτές οι **κατευθύνσεις Lyapunov** εξαρτώνται από τη ροή του συστήματος και καθορίζονται χρησιμοποιώντας τον Ιακωβιανό πίνακα, δηλαδή, την **εφαπτομενική απεικόνιση κάθε σημείου που μας ενδιαφέρει κατά μήκος της ροής**. Άρα θα πρέπει να διατηρήσουμε το σωστό προσανατολισμό στο χώρο χρησιμοποιώντας μια κατάλληλη προσέγγιση της εφαπτομενικής απεικόνισης. Αυτή η απαίτηση ωστόσο, δεν είναι αναγκαία όταν θέλουμε να υπολογίσουμε μόνο το μεγαλύτερο εκθέτη Lyapunov.

Μπορούμε να αναμένουμε ότι δύο τυχαία επιλεγμένα αρχικές συνθήκες θα αποκλίνουν εκθετικά σε βαθμό που προκύπτει από το μεγαλύτερο εκθέτη Lyapunov. Με άλλα λόγια, μπορούμε να αναμένουμε ότι **ένα τυχαίο διάνυσμα αρχικών συνθηκών θα συγκλίνει στην πιο ασταθή πολλαπλότητα**, δεδομένου ότι η εκθετική αύξηση στην κατεύθυνση αυτή κυριαρχεί στην διαστολή (ή τη συστολή) κατά μήκος των άλλων κατευθύνσεων Lyapunov. Έτσι, ο μεγαλύτερος εκθέτης Lyapunov μπορεί να οριστεί χρησιμοποιώντας την ακόλουθη εξίσωση όπου $d(t)$ είναι η μέση απόκλιση σε χρόνο t και C είναι μια σταθερά που ομαλοποιεί τον αρχικό διαχωρισμό:

$$d(t)=Ce^{\lambda_1 t}$$

Για τις πειραματικές εφαρμογές, ένας αριθμός ερευνητών έχουν προτείνει αλγορίθμους που υπολογίζουν το μεγαλύτερο εκθέτη Lyapunov, το θετικό φάσμα Lyapunov, δηλαδή, μόνο τους θετικούς εκθέτες, ή το πλήρες φάσμα Lyapunov. Κάθε μέθοδος από αυτές όμως έχει τουλάχιστον ένα από τα ακόλουθα μειονεκτήματα:

- ✓ είναι **αναξιόπιστη** για **μικρά σύνολα δεδομένων**,
- ✓ είναι **υπολογιστικά απαιτητική,**
- ✓ είναι σχετικά **δύσκολο να εφαρμοστεί**

Μία άλλη μέθοδος υπολογισμού των εκθετών προτάθηκε από τον Sato και τους συνεργάτες του και θα τη δούμε αναλυτικά παρακάτω:

3.3.2 Η μέθοδος του Sato για τον υπολογισμό του μέγιστου εκθέτη Lyapunov

Η μέση εκθετική απόκλιση της απόστασης των γειτονικών τροχιών μελετάται σε λογαριθμική τροχιά μέσω του σφάλματος πρόβλεψης (prediction error):

$$p(k) = \frac{1}{Nt_s} \sum_{n=1}^{N} \log_2 \left(\frac{\|\mathbf{y}^{n+k} - \mathbf{y}^{nn+k}\|}{\|\mathbf{y}^{n} - \mathbf{y}^{nn}\|} \right)$$

Όπου y^{nn} είναι ο πιο κοντινός γείτονας του y^{n}. Η εξάρτηση του σφάλματος πρόβλεψης από τον αριθμό των k χρονικών στιγμών μπορεί να χωριστεί σε τρεις φάσεις:

- **Φάση 1:** Η γειτονική τροχιά **συγκλίνει** στην κατεύθυνση που αντιστοιχεί στον μέγιστο εκθέτη Lyapunov
- **Φάση 2:** Η απόσταση **αυξάνεται εκθετικά** με ρυθμό $e^{(\lambda_1 ts k)}$ μέχρι να υπερβεί το εύρος της εγκυρότητας της γραμμικής προσέγγισης της ροής γύρω από την τροχιά αναφοράς $\{y^{n+k}\}$.
- **Φάση 3:** Η απόσταση **αυξάνεται πιο αργά** από την προηγούμενη εκθετική αύξηση μέχρι να μειωθεί και πάλι λόγω των αναδιπλώσεων του ελκυστή.

Αν η δεύτερη φάση είναι αρκετά μεγάλη, εμφανίζεται ένα **γραμμικό τμήμα** με κλίση λ_1 στο διάγραμμα $k\text{-}p(k)$. Αυτό όχι μόνο επιτρέπει μια πρόβλεψη του μέγιστου εκθέτη Lyapunov, αλλά ακόμα μας δίνει μια άμεση πιστοποίηση της εκθετικής αύξησης των αποστάσεων, έτσι ώστε να ξεχωρίζουμε το ντετερμινιστικό χάος από τις στοχαστικές διαδικασίες, όπου ο διαχωρισμός των τροχιών είναι μη εκθετικός.

Για να φτάσουμε όμως σε αυτό το τελικό στάδιο, πρέπει αρχικά να ακολουθήσουμε μια διαδικασία **ανακατασκευής του χώρου των καταστάσεων** του δυναμικού μας μοντέλου και στη συνέχεια υπολογισμού διάφορων παραμέτρων.

Στα μαθηματικά μοντέλα των δυναμικών συστημάτων, η δυναμική τους περιγράφεται από τον χώρο των καταστάσεών τους, του οποίου το ακέραιο μέρος της διάστασής του δίνεται από τον αριθμό των μεταβλητών του μοντέλου. Στα πειράματα ωστόσο μετράται μόνο μια μεταβλητή συναρτήσει του χρόνου και συνεπώς **ο χώρος των καταστάσεων είναι συνήθως άγνωστος**. Άρα πώς να καταλήξουμε στον ελκυστή που χαρακτηρίζει το σύστημα;

Αυτό το κενό μεταξύ των θεωρητικών εννοιών και των παρατηρούμενων ποσοτήτων γέμισε το 1980 όταν οι **Packard, Crutchfield, Farmer** και **Shaw** δημοσίευσαν το άρθρο τους "**Geometry from a time series**", όπου για πρώτη φορά η ανακατασκευή του χώρου των καταστάσεων συναρτήσει του χρόνου εφαρμόστηκε σε χρονοσειρές.

Αποδείχτηκε ότι είναι δυνατόν να ανακατασκευάσουμε από μια βαθμωτή χρονοσειρά, ένα καινούριο χώρο καταστάσεων που είναι αμφιμορφικά ισοδύναμος με τον κανονικό χώρο των καταστάσεων του πειραματικού συστήματος. (αμφιμορφισμός είναι ένας ισομορφισμός σε μία λεία πολλαπλότητα). Βασισμένοι σε αυτές τις ανακατασκευασμένες καταστάσεις, η χρονοσειρά μπορεί να αναλυθεί από τη σκοπιά της ντετερμινιστικής μη γραμμικής δυναμικής. Είναι δυνατόν να μοντελοποιήσουμε και να προβλέψουμε τη δυναμική και να τη χαρακτηρίσουμε με βάση τις διαστάσεις και τους εκθέτες Lyapunov για παράδειγμα. Στη συνέχεια θα αναφερθούμε σε κάποια από αυτές τις περιπτώσεις.

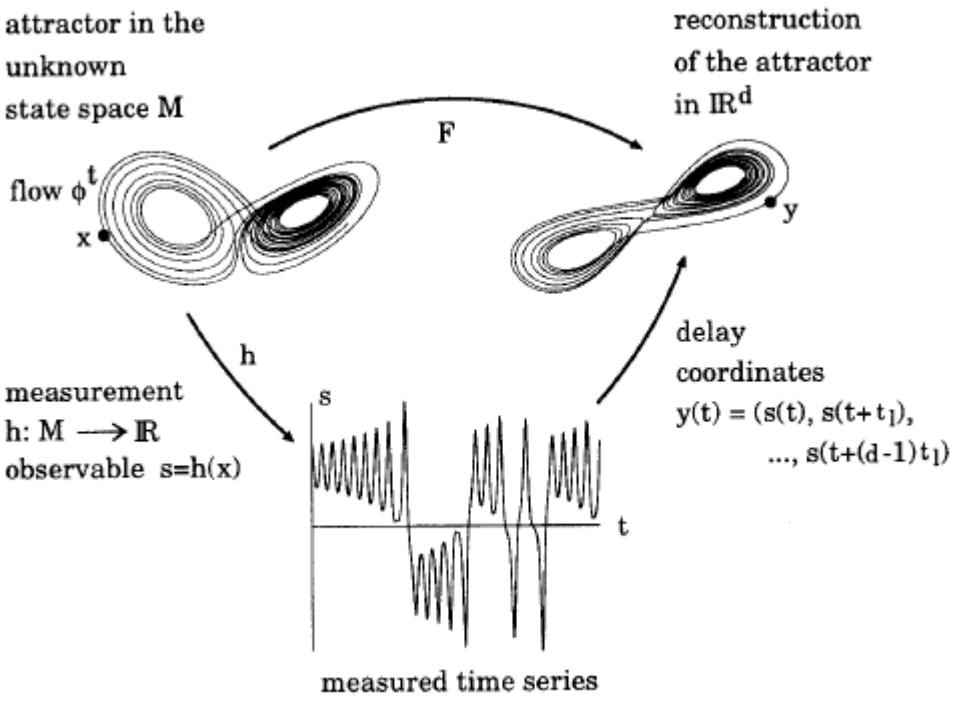
3.3.2.1 Ανακατασκευή του χώρου των καταστάσεων

Έστω M μια ομαλή C^2 m-διάστατη πολλαπλότητα που αποτελεί τον πραγματικό χώρο των καταστάσεων του δυναμικού συστήματος που ερευνούμε και έστω φ^t: $M \rightarrow M$ η ταυτοτική του απεικόνιση. Έστω ότι μπορούμε να μετρήσουμε μία βαθμωτή ποσότητα $s(t)= h(x(t))$ που δίνεται από μια συνάρτηση $h: M \rightarrow \mathbb{R}$, όπου $x(t)=\varphi_t(x(0))$. Τότε μπορούμε να κατασκευάσουμε μια 1-1 απεικόνιση συντεταγμένων καθυστέρησης (delay coordinates)

$$F: M \rightarrow R_d$$

$$x \rightarrow y = F(x) = (s(t), s(t+t1), \dots, s(t+(d-1)t1)$$

που απεικονίζει μια κατάσταση x από τον κανονικό χώρο καταστάσεων M σε ένα σημείο y του ανακατασκευασμένου χώρου των καταστάσεων $\mathbb{R}^d$, όπου d είναι η διάσταση εμβύθισης και η μεταβλητή t_1 μας δίνει το χρόνο καθυστέρησης (ή την υστέρηση φάσης (lag)) που χρησιμοποιήθηκε. Το επόμενο σχήμα μας δίνει μία απεικόνιση αυτής της κατασκευής.



Σχήμα 3.5: Ανακατασκευή των καταστάσεων από βαθμωτή χρονοσειρά

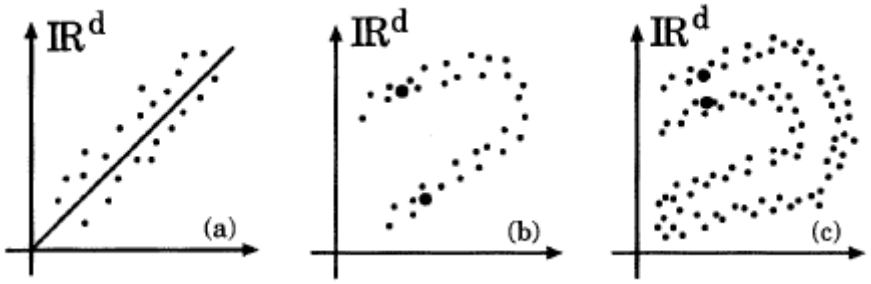
3.3.2.2 Επιρροή των παραμέτρων της ανακατασκευής

Η ανακατασκευή χρησιμοποιώντας αυτές τις συντεταγμένες βασίζεται σε δύο παραμέτρους: την διάσταση εμβύθισης (embedding dimension) *d* και τον χρόνο καθυστέρησης (delay time) t_l. Θα συζητήσουμε τώρα την επιρροή των δύο αυτών παραμέτρων στην ανακατασκευή αυτή.

1) Επιλογή του χρόνου καθυστέρησης

Όσον αφορά την επιλογή του χρόνου καθυστέρησης t_1 διαφορετικοί χρόνοι μας δίνουν ανακατασκευές του ελκυστή που είναι αμφιμορφικά (diffeomorphically) παρόμοιοι αλλά γεωμετρικά διαφορετικοί. Όταν ο χρόνος καθυστέρησης είναι πολύ μικρός, οι συντεταγμένες $y^{in} = s^{n+(i-1)t}$ της κάθε ανακατασκευασμένης κατάστασης y^n δεν διαφέρουν σημαντικά η μία από την άλλη και συνεπώς τα σημεία της κατάστασης βρίσκονται πάνω σε μία διαγώνιο . Σ αυτή τη περίπτωση, κάθε έρευνα για πιθανή fractal μορφή του εκλυστή (για παράδειγμα η εκτίμηση της διάστασης) γίνεται δύσκολη, γιατί θα πρέπει να μεγεθύνουμε το σύνολο των σημείων που μελετάμε που είναι πολύ μικρότερο από τη διάμετρο του συνόλου που βρίσκονται κάθετα στη διαγώνιο.

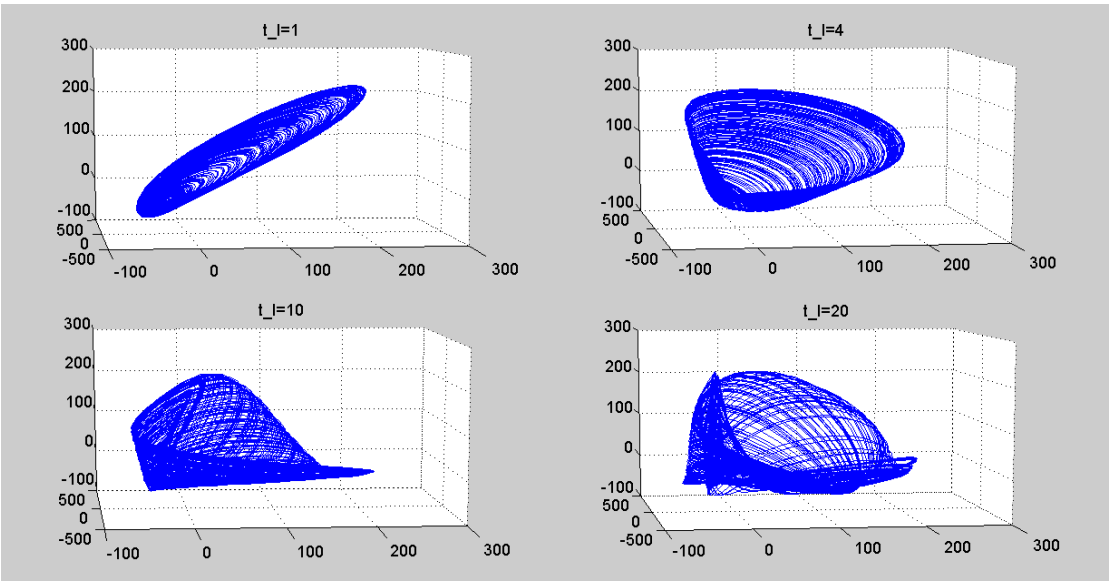
Σε πολύ μικρές κλίμακες ωστόσο, αυτές οι έρευνες γίνονται σχεδόν αδύνατες, λόγω του περιορισμένου αριθμού των δεδομένων και της αναπόφευκτης παρουσίας θορύβου.



Σχήμα 3.6: Διάγραμμα τριών ανακατασκευών με διαφορετικούς χρόνους καθυστέρησης: (a) πολύ μικρός, (b) κανονικός, (c) πολύ μεγάλος

Η κατάσταση βελτιώνεται όταν ο χρόνος αυξάνεται. Τότε ο ελκυστής αναδιπλώνεται και το εσωτερικό του «φανερώνεται», όπως φαίνεται και στο παρακάτω σχήμα, όπου έχουμε χρησιμοποιήσει χρονοσειρά που έχουμε πάρει από τον ταλαντωτή colpitts.

-Ο κώδικας για την κατασκευή του ελκυστή βρίσκεται στο παράρτημα-



Σχήμα 3.7: Ο ελκυστής του χαοτικού ταλαντωτή colpitts, για διάφορες τιμές του χρόνου καθυστέρησης.

Όταν ο χρόνος καθυστέρησης αυξηθεί ακόμα περισσότερο, ο ανακατασκευασμένος ελκυστής **αναδιπλώνεται και πάλι**. Αυτή η αναδίπλωση δεν συμβαίνει για περιοδικά σήματα και οι ρίζες του είναι ο μηχανισμός της «συστολής και αναδίπλωσης», που είναι χαρακτηριστικό για τα χαοτικά συστήματα με ευαίσθητη εξάρτηση από τις αρχικές συνθήκες. Ωστόσο, δεν θέλουμε επιπλέον αναδιπλώσεις στην ανακατασκευή γιατί φέρνουν τις καταστάσεις (states) πιο κοντά στον ανακατασκευασμένο χώρο των καταστάσεων, οι οποίες δεν είναι τόσο κοντά στον κανονικό χώρο καταστάσεων.

Πιο συγκεκριμένα, με παρουσία θορύβου, που τα σημεία της κατάστασης είναι επιπλέον μετατοπισμένα από τις «σωστές θέσεις», κάτι που μπορεί να οδηγήσει σε περιπτώσεις που οι καταστάσεις είναι λανθασμένες. Ακόμα και χωρίς θόρυβο, μια πολύ μεγάλη τιμή του t_l μπορεί να οδηγήσει σε κάποια αναδίπλωση που δίνει κόμβους του ελκυστή όπου η ανακατασκευή δεν είναι 1-1 πια. Σε αυτή την περίπτωση μπορούμε να αυξήσουμε τη διάσταση d, για να «φτιάξουμε» την εμβύθιση (embedding) αλλά αυτό θα προκαλέσει και περισσότερη ευαισθησία στην μείωση του χρόνου καθυστέρησης.

2) Επιλογή της διάστασης εμβύθισης

Θεώρημα: Έστω (M, g) είναι μια πολλαπλότητα Riemann (Riemannian manifold) και $f: M^m \rightarrow \mathbb{R}^n$ μια σύντομη C^∞ -εμβύθιση στην Ευκλείδειο χώρο $\mathbb{R}^n$, όπου $n \geq m + 1$.

Στη συνέχεια, για κάποιο αυθαίρετο ε> 0 υπάρχει μια εμβύθιση $f_\varepsilon : \mathbf{M^m} \rightarrow \mathbb{R}^n$ η οποία είναι:

i) τάξης C^1,

ii) ισομετρία: για κάθε δύο διανύσματα v, w ∈ T_x (M) στον εφαπτόμενο χώρο που εφάπτεται στο x ∈ M, να ισχύει:

$$g(v,w)= <df_\varepsilon(v), df_\varepsilon(w)>$$

iii) ε-κοντά στην f:

$$|f(x) - f_\varepsilon(x)|< \varepsilon,$$ για κάθε x ϵ M

Όταν η επιλεγόμενη διάσταση d είναι πολύ μικρή, οι συνθήκες του θεωρήματος εμβύθισης δεν ικανοποιούνται. Απ την άλλη, για πολύ μεγάλη διάσταση εμφανίζονται πρακτικά προβλήματα λόγω του συγκεκριμένου αριθμού των σημείων που κατασκευάζουν όλο και μικρότερα σύνολα στον $\mathbb{R}^d$, όταν το d αυξάνεται. Συνεπώς, η καλύτερη επιλογή είναι η μικρότερη τιμή της διάστασης που μας δίνει μια καλή ανακατασκευή.

3. Επιλογή παραμέτρων ανακατασκευής

Πολλοί αλγόριθμοι έχουν προταθεί για την εύρεση των καταλληλότερων τιμών για τα d και t_l. Οι περισσότερες από τις μεθόδους για τον προσδιορισμό του d βασίζονται σε συνεχείς δοκιμές για τις επαγόμενες ροές στον χώρο ανακατασκευής ή στην ίδια την εμβύθιση. Η κύρια ιδέα είναι να ελέγξουμε **κατά πόσον τα γειτονικά σημεία απεικονίζονται σε γειτονικά σημεία.**

Για να βρούμε τις βέλτιστες τιμές για τα t_l υπάρχει μία κλάση αλγορίθμων που αφορά τη γεωμετρία του ανακατασκευασμένου ελκυστή, και συγκεκριμένα την αναδίπλωση του. Η πιο διαδεδομένη διαδικασία έχει να κάνει με την ελαχιστοποίηση του πλεονασμού των συντεταγμένων των ανακατασκευασμένων καταστάσεων χρησιμοποιώντας γραμμικές συναρτήσεις αυτοσυσχέτισης ή θεωρητικές έννοιες, όπως οι κοινές πληροφορίες. Επειδή οι κακές τιμές για το t_l μπορούν να καταστρέψουν την εμβύθιση, τα τεστ για την εκτίμηση της διάστασης d μπορούν επίσης να χρησιμοποιηθούν για να καθοριστεί το t_l.

Μέχρι στιγμής, δεν έχει καθοριστεί κάποια μέθοδος για τη βέλτιστη εκτίμηση των παραμέτρων εμβύθισης. Ο λόγος γι αυτό μάλλον είναι το γεγονός ότι δεν υπάρχει ζεύγος (d,t_l) που να είναι βέλτιστο για όλες τις εφαρμογές. Επίσης υπάρχουν ενδείξεις ότι το μέγεθος dt_l είναι πιο σημαντικό από τις μεμονωμένες τιμές d και t_l[2].

Επιπλέον πολλοί από τους προτεινόμενους αλγορίθμους χρειάζονται πολύ λίγο χρόνο για τους υπολογισμούς, γι αυτό το λόγο μπορούμε να επαναλάβουμε τους υπολογισμούς για πολλές τιμές των d και t_l και συνεπώς να εξάγουμε **όχι μόνο τις βέλτιστες παραμέτρους ανακατασκευής** αλλά επίσης και κάποια τεστ για τη **δύναμη της μεθόδου που χρησιμοποιούμε** καθώς και κάποιες πρώτες πληροφορίες για τα δεδομένα μας.

3.3.3 Υπολογισμός εκθετών Lyapunov με χρήση Matlab

Σκοπός της ενότητας αυτής είναι να συσχετίσουμε το μέτρο της χαοτικότητας της χρονοσειράς μας με το σφάλμα που μας δίνει η πρόβλεψή της από το ESN. Η συσχέτιση αυτή είναι πολύ χρήσιμη και έχει πολλές εφαρμογές όπως για παράδειγμα στην κρυπτογραφία.

Αν κάποιος θέλει για παράδειγμα να στείλει ένα μήνυμα σε κάποιο άλλο άτομο χωρίς να το διαβάσει ένας τρίτος, μπορεί να κωδικοποιήσει το μήνυμά του σε μια χρονοσειρά και να της προσθέσει θόρυβο. Ο θόρυβος αυτός θα είναι επίσης

σε μορφή χρονοσειράς (πχ σειρά που παίρνουμε από laser κάποιας συγκεκριμένης συχνότητας). Ο παραλήπτης του μηνύματος, ξέροντας τη συχνότητα του θορύβου, μπορεί να παράγει την ίδια ακριβώς συχνότητα και να αφαιρέσει το θόρυβο, διαβάζοντας το μήνυμα. Το ζητούμενο είναι να βρούμε πόσο πολύπλοκο πρέπει να είναι το σήμα μας, ώστε να μην μπορεί να το προβλέψει ένας τρίτος και κατά συνέπεια να διαβάσει το κρυπτογραφημένο μας μήνυμα.

Να σημειώσουμε εδώ ότι δεν είναι ανάγκη να προβλεφθεί κατά 100% η χρονοσειρά. Ανάλογα με το είδος του μηνύματος και τη πληροφορία που θέλουμε να εξάγουμε, μπορούμε να αρκεστούμε σε κάποιο ποσοστό πρόβλεψης άρα και σε κάποιο ποσοστό σφάλματος, το οποίο μπορεί να διαφέρει ανάλογα με το πρόβλημα.

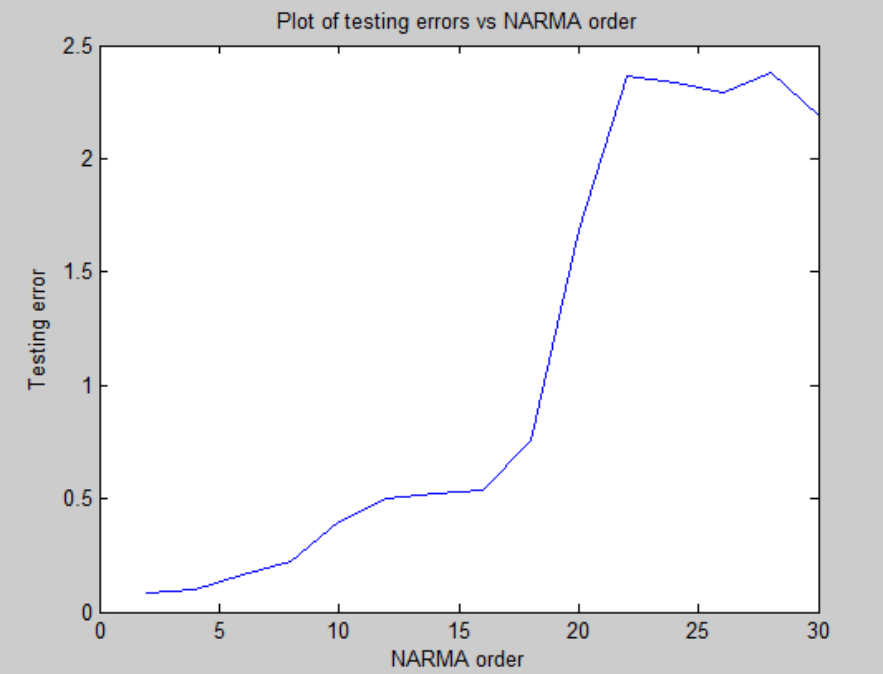
Θα δούμε λοιπόν καταρχάς τα σφάλματα που προκύπτουν και τους εκθέτες Lyapunov κάθε χρονοσειράς NARMA.

Καταρχάς, για τις χρονοσειρές NARMA τάξης 2,4,…,30 υπολογίζουμε τα σφάλματα δοκιμής και τα βάζουμε όλα σε ένα διάνυσμα με την εντολή

```
b=[mTes2,mTes4,mTes6,mTes8,mTes10,mTes12,mTes14,mTes16,mTes18,mTes20,
                mTes22,mTes24,mTes26,mTes28,mTes30]
```

Το αποτέλεσμα που παίρνουμε είναι το εξής:

```
Columns 1 through 11

  0.0863    0.0965    0.1629    0.2198    0.3998    0.4986    0.5236    0.5374    0.7586    1.6863    2.3678

Columns 12 through 15

  2.3333    2.2951    2.3802    2.1900
```



Σχήμα 3.8: Διάγραμμα του σφάλματος δοκιμής σε συνάρτηση με την τάξη της χρονοσειράς

Για τον υπολογισμό των εκθετών Lyapunov θα χρησιμοποιήσουμε τον αλγόριθμο του Wolf, ο οποίος εξηγείται παρακάτω:

Αλγόριθμος:

1. Ενσωμάτωση των δεδομένων.

2. Επιλογή σημείου x (t_0) κάπου στη μέση της τροχιάς.

3. Εύρεση του πλησιέστερου γείτονα σε εκείνο το σημείο. Έστω z_0 (t_0).

4. Υπολογισμός || z_0 (t_0)- x (t_0) || = L_0

5. Ακολουθούμε την "απόκλιση της τροχιάς» - διακεκομμένη γραμμή στο σχήμα - προς τα εμπρός στο χρόνο, υπολογίζοντας το || z_0 (t_0)- x (t_0) || = L_0 (i) και αυξάνοντας το i, μέχρι L_0 (i)> ϵ. Καλούμε αυτή τη τιμή L' και αυτό το χρόνο t_1.

6. Εύρεση του z_1 (t_1), που είναι ο «κοντινότερος γείτονας" του x (t_1), και επιστρέφουμε στο βήμα 4.

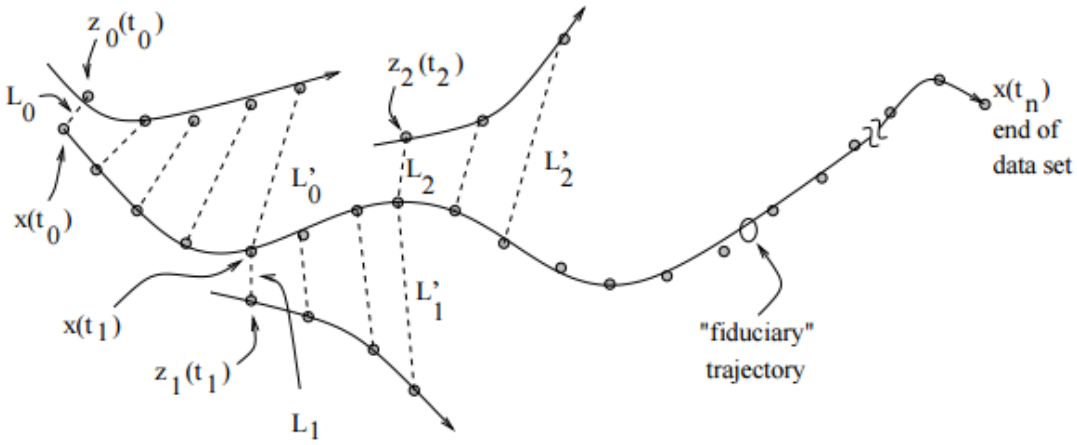
Επαναλαμβάνουμε τη διαδικασία μέχρι t=t_n κρατώντας ταυτόχρονα όλες τις τιμές L_i και L'_i

Υπολογίζουμε τον μέγιστο θετικό εκθέτη Lyapunov με τη βοήθεια της σχέσης

$$\lambda_1 \approx \frac{1}{N\Delta t}\sum_{1}^{M-1} log_2 \frac{L'_i}{L_i}$$

όπου M είναι ο αριθμός των φορών που τρέξαμε τον παραπάνω βρόχο και N είναι ο αριθμός των χρονικών στιγμών.

Μια Σχηματική αναπαράσταση του παραπάνω αλγόριθμου βλέπουμε στο σχήμα 3.9:



Σχήμα 3.9: Αναπαράσταση του αλγορίθμου υπολογισμού του μέγιστου θετικού εκθέτη του Wolf

Ο κώδικας στο Matlab για τον αλγόριθμο του Wolf, δίνεται παρακάτω:

```
function lam = lyapunov(x,dt)

% Εισαγωγή των δεδομένων μέσω διανύσματος x
[ndata nvars]=size(x);


% Ορισμός φράγματος για το σφάλμα
N2 = floor(ndata/2);
N4 = floor(ndata/4);
TOL = 1.0e-6;


exponent = zeros(N4+1,1);


% Το 2ο τεταρτημόριο των δεδομένων πρέπει να παίξουν ενεργό ρόλο
for i=N4:N2
   dist = norm(x(i+1,:)-x(i,:));
   indx = i+1;
   for j=1:ndata-5
       if (i ~= j) && norm(x(i,:)-x(j,:))<dist
           dist = norm(x(i,:)-x(j,:));
           indx = j; % πιο κοντινό σημείο!
       end
   end

% Υπολογισμός τάξης της διαστολής (πχ μεγαλύτερη ιδιοτιμή)
   expn = 0.0;
   for k=1:5
       if norm(x(i+k,:)-x(indx+k,:))>TOL && norm(x(i,:)-x(indx,:))>TOL
           expn = expn + (log(norm(x(i+k,:)-x(indx+k,:)))-log(norm(x(i,:)-
x(indx,:))))/k;
       end
   end
   exponent(i-N4+1)=expn/5;
```

```
end

% Υπολογισμός του συνολικού μέσου πάνω από τα N4 δεδομένα
sum=0;
for i=1:N4+1
    sum = sum+exponent(i);
end

% Επιστροφή της μέσης τιμής
lam =sum/((N4+1)*dt);
```

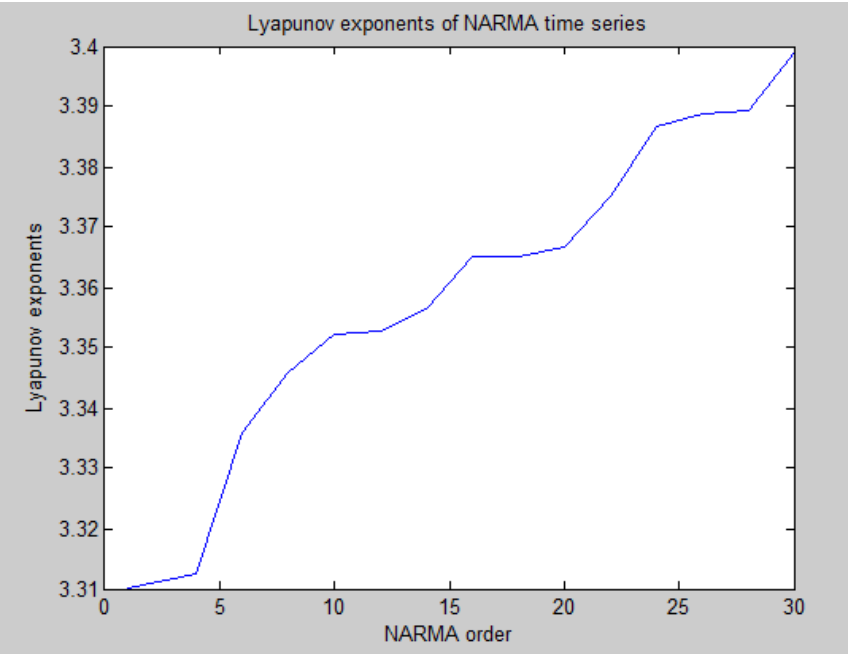
Να σημειώσουμε επίσης ότι επειδή οι χρονοσειρές είναι χαοτικές, θα πρέπει να πάρουμε ένα πολύ μεγάλο αριθμό σημείων για να συγκλίνει η μέθοδος στο σωστό εκθέτη Lyapunov. Στη συγκεκριμένη περίπτωση, πήραμε για κάθε χρονοσειρά 10.000 σημεία. *Η παραγωγή τους έγινε με τη συνάρτηση generate_NARMA_sequence που υπάρχει στο παράρτημα.* Η συνάρτηση αυτή δέχεται σαν ορίσματα την τάξη καθώς και το μέγεθος της χρονοσειράς που θέλουμε να κατασκευάσουμε.

Για τις συγκεκριμένες χρονοσειρές λοιπόν παίρνουμε λοιπόν τα εξής αποτελέσματα:

```
lyapun =

  Columns 1 through 11

    3.3101    3.3124    3.3359    3.3458    3.3522    3.3528    3.3564    3.3650    3.3650    3.3668    3.3752

  Columns 12 through 15

    3.3867    3.3888    3.3894    3.3990
```

Όπως είναι αναμενόμενο, καθώς αυξάνεται η τάξη της σειράς NARMA αυξάνεται και ο μέγιστος εκθέτης Lyapunov, δηλαδή και η χαοτικότητα της χρονοσειράς. Ωστόσο επειδή ο μηχανισμός παραγωγής των σημείων είναι σταθερός, δεν παρατηρούμε κάποια πολύ μεγάλη διαφορά στις τιμές.

Για να έχουμε μια καλύτερη εικόνα των αποτελεσμάτων, τα αναπαριστούμε γραφικά, σε συνάρτηση με τη τάξη της χρονοσειράς NARMA.



Σχήμα 3.10: Εκθέτες Lyapynov συναρτήσει της τάξης της χρονοσειράς NARMA

Βλέπουμε λοιπόν την αυξητική τάση που έχει η χαοτικότητα της χρονοσειράς καθώς αυξάνεται και η τάξη της.

Συμπερασματικά λοιπόν, βλέπουμε ότι με την αύξηση της τάξης της χρονοσειράς μας, αυξάνεται και η δυσκολία πρόβλεψής της. Ωστόσο, ανάλογα με την ακρίβεια και κατά συνέπεια με το σφάλμα που θέλουμε, μπορούμε να αλλάξουμε πολύ εύκολα την πολυπλοκότητά της και να οδηγηθούμε στα βέλτιστα για εμάς αποτελέσματα.

4. Συμπεράσματα

Τα νευρωνικά δίκτυα και γενικότερα ο κλάδος του machine learning είναι ένας νέος και πολλά υποσχόμενος επιστημονικός κλάδος που τα τελευταία χρόνια έχει αναπτυχθεί πάρα πολύ.

Όλες οι εφαρμογές των νευρωνικών δικτύων έχουν προκύψει τα τελευταία λίγα χρόνια και μερικές από αυτές ήδη βρίσκονται ως έτοιμα προϊόντα στην αγορά και χρησιμοποιούνται ευρέως. Είναι βέβαιο ότι τα επόμενα χρόνια ένας πολύ μεγαλύτερος αριθμός θα ακολουθήσει, αφού ακόμη το πεδίο αυτό βρίσκεται σε νηπιακή ηλικία. Οι εφαρμογές αυτές περιλαμβάνουν αναγνώριση προτύπων, υπολογισμό συναρτήσεων, βελτιστοποίηση, πρόβλεψη, αυτόματο έλεγχο και πολλά άλλα θέματα.

Μέσω αυτής της διπλωματικής εργασίας, αφού είδαμε κάποια είδη νευρωνικών δικτύων, αναλύσαμε πως μπορούμε να χρησιμοποιήσουμε αυτά τα δίκτυα πρόβλεψης χρονοσειρών με τον βέλτιστο δυνατό τρόπο, αλλάζοντας κάθε φορά είτε τον τρόπο εκπαίδευσης, είτε το ποσοστό των δεδομένων που χρησιμοποιούνται για την εκπαίδευση. Καταλήξαμε επιπλέον σε συμπεράσματα σχετικά με την απόδοση του νευρωνικού μας δικτύου και τον αλγόριθμο εκπαίδευσής του.

Είδαμε επίσης πως σχετίζεται η χαοτικότητα των χρονοσειρών με τους εκθέτες Lyapunov και το σφάλμα πρόβλεψής τους από το ESN. Πιο συγκεκριμένα, αυξάνοντας την τάξη k της χρονοσειράς NARMA που μελετάμε, δηλαδή τον αριθμό των i-k αρχικών τιμών που πρέπει να πάρουμε για να βρούμε την i τιμή, βλέπουμε ότι αυξάνεται και η πολυπλοκότητά της, κάνοντας τη διαδικασία πρόβλεψής της πιο επίπονη διαδικασία. Βρίσκοντας όμως τη χρυσή τομή ανάμεσα στο βαθμό πολυπλοκότητάς της και το σφάλμα που επιθυμούμε για να έχουμε μια «καλή πρόβλεψη», μπορούμε να χρησιμοποιήσουμε τα μοντέλα αυτά σε πολλές εφαρμογές της καθημερινότητάς μας.

5. Παράρτημα (Κώδικες Matlab)

5.1 ESN Toolbox

✓ Κατασκευή ESN

```
function esn = generate_esn(nInputUnits, nInternalUnits, nOutputUnits, varargin)
% Creates an ESN set up for use in multiple-channel output association tasks.
% The number of input, internal, and output
% units have to be set. Any other option is set using the format
% 'name_of_options1',value1,'name_of_option2',value2, etc.
%
%%%%% input arguments:
% nInputUnits: the dimension of the input
% nInternalUnits: size of the Esn
% nOutputUnits: the dimension of the output
%
%%%%% optional arguments:
% 'inputScaling': a nInputUnits x 1 vector
%
% 'inputShift': a nInputUnits x 1 vector.
%
% 'teacherScaling': a nOutputUnits x 1 vector
%
% 'teacherShift': a nOutputUnits x 1 vector.
%
% 'noiseLevel': a small number containing the amount of uniform noise to be
%  added when computing the internal states
%
% 'learningMode': a string ('offline_singleTimeSeries', 'offline_multipleTimeSeries' or 'online')
%     1. Case 'offline_singleTimeSeries': trainInput and trainOutput each represent a
%        single time series in an array of size sequenceLength x sequenceDimension
%     2. Case 'offline_multipleTimeSeries': trainInput and trainOutput each represent a
%        collection of K time series, given in cell arrays of size K x 1, where each cell is an
%        array of size individualSequenceLength x sequenceDimension
%     3. Case 'online': trainInput and trainOutput are a single time
%        series, output weights are adapted online
%
% 'reservoirActivationFunction': a string ("tanh", "identity", "sigmoid01") ,
%
% 'outputActivationFunction': a string("tanh", "identity", "sigmoid01") ,
%
% 'inverseOutputActivationFunction': the inverse to
%    outputActivationFunction, one of 'atanh', 'identity', 'sigmoid01_inv'.
%    When choosing the activation function, make sure the inverse
%    activation function is corectly set.
%
% 'methodWeightCompute': a string ('pseudoinverse', 'wiener_hopf'). It
%    specifies which method to use to compute the output weights given the
%    state collection matrix and the teacher
%
% 'spectralRadius': a positive number less than 1.
%
% 'feedbackScaling': a nOutputUnits x 1 vector, indicating the scaling
%     factor to be applied on the output before it is fed back into the network
%
% 'type': a string ('plain_esn', 'leaky_esn' or 'twi_esn')
% 'trained': a flag indicating whether the network has been trained already
% 'timeConstants': option used in networks with type == "leaky_esn", "leaky1_esn" and "twi_esn".
%                      Is given as column vector of size esn.nInternalUnitsm, where each entry
%                      signifies a time constant for a reservoir neuron.
% 'leakage': option used in networks with type == "leaky_esn" or "twi_esn"
% 'RLS_lambda': option used in online training(learningMode == "online")
% 'RLS_delta': option used in online training(learningMode == "online")
%
% for more information on the Echo State network approach take a look at
% the following tutorial :
% http://www.faculty.iu-bremen.de/hjaeger/pubs/ESNTutorialRev.pdf


% Version 1.0, April 30, 2006
% Copyright: Fraunhofer IAIS 2006 / Patent pending
% Revision 1, June 6, 2006, H. Jaeger
% Revision 2, Feb 23, 2007, H. Jaeger
% Revision 3, June 27, 2007, H. Jaeger
% Revision 4, July 1, 2007, H. Jaeger:
```

```
%    - changed esn.timeConstant to esn.timeConstants
%    - deleted esn.retainment
%    - deleted esn.internalWeights (to enforce that this is set outside
%    this generation script)
% Revision 5, July 29, 2007, H. Jaeger: bugfix (for cases of zero input or
%    output length)
% Revision 6, Jan 28, 2009, H. Jaeger: bugfix (deleted defunct error
%                            catching routine for leaky / twiesn
%                            reservoirs)



%%%% set the number of units
esn.nInternalUnits = nInternalUnits;
esn.nInputUnits = nInputUnits;
esn.nOutputUnits = nOutputUnits;


connectivity = min([10/nInternalUnits 1]);
nTotalUnits = nInternalUnits + nInputUnits + nOutputUnits;


esn.internalWeights_UnitSR = generate_internal_weights(nInternalUnits, ...
                                                       connectivity);
esn.nTotalUnits = nTotalUnits;


% input weight matrix has weight vectors per input unit in colums
esn.inputWeights = 2.0 * rand(nInternalUnits, nInputUnits)- 1.0;


% output weight matrix has weights for output units in rows
% includes weights for input-to-output connections
esn.outputWeights = zeros(nOutputUnits, nInternalUnits + nInputUnits);


%output feedback weight matrix has weights in columns
esn.feedbackWeights = (2.0 * rand(nInternalUnits, nOutputUnits)- 1.0);


%init default parameters
if nInputUnits > 0
    esn.inputScaling  = ones(nInputUnits, 1);  esn.inputShift    = zeros(nInputUnits, 1);
else
    esn.inputScaling = []; esn.inputShift = [];
end
if nOutputUnits > 0
    esn.teacherScaling = ones(nOutputUnits, 1); esn.teacherShift  = zeros(nOutputUnits, 1);
else
    esn.teacherScaling = []; esn.teacherShift  =  [];
end
esn.noiseLevel = 0.0 ;
esn.reservoirActivationFunction = 'tanh';
esn.outputActivationFunction = 'identity' ; % options: identity or tanh or sigmoid01
esn.methodWeightCompute = 'pseudoinverse' ; % options: pseudoinverse and wiener_hopf
esn.inverseOutputActivationFunction = 'identity' ; % options:
                                                   % identity or
                                                   % atanh or sigmoid01_inv
esn.spectralRadius = 1 ;
esn.feedbackScaling = zeros(nOutputUnits, 1);
esn.trained = 0 ;
esn.type = 'plain_esn' ;
esn.timeConstants = ones(esn.nInternalUnits,1);
esn.leakage = 0.5;
esn.learningMode = 'offline_singleTimeSeries' ;
esn.RLS_lambda = 1 ;


args = varargin;
nargs= length(args);
for i=1:2:nargs
  switch args{i},
   case 'inputScaling', esn.inputScaling = args{i+1} ;
   case 'inputShift', esn.inputShift= args{i+1} ;
   case 'teacherScaling', esn.teacherScaling = args{i+1} ;
   case 'teacherShift', esn.teacherShift = args{i+1} ;
   case 'noiseLevel', esn.noiseLevel = args{i+1} ;
   case 'learningMode', esn.learningMode = args{i+1} ;
   case 'reservoirActivationFunction',esn.reservoirActivationFunction=args{i+1};
   case 'outputActivationFunction',esn.outputActivationFunction=  ...
                       args{i+1};
   case 'inverseOutputActivationFunction', esn.inverseOutputActivationFunction=args{i+1};
   case 'methodWeightCompute', esn.methodWeightCompute = args{i+1} ;
   case 'spectralRadius', esn.spectralRadius = args{i+1} ;
   case 'feedbackScaling',  esn.feedbackScaling = args{i+1} ;
   case 'type' , esn.type = args{i+1} ;
   case 'timeConstants' , esn.timeConstants = args{i+1} ;
   case 'leakage' , esn.leakage = args{i+1} ;
   case 'RLS_lambda' , esn.RLS_lambda = args{i+1};
```

```
   case 'RLS_delta' , esn.RLS_delta = args{i+1};

   otherwise error('the option does not exist');
  end
end

%%%% error checking
% check that inputScaling has correct format
if size(esn.inputScaling,1) ~= esn.nInputUnits
    error('the size of the inputScaling does not match the number of input units');
end
if size(esn.inputShift,1) ~= esn.nInputUnits
    error('the size of the inputScaling does not match the number of input units');
end
if size(esn.teacherScaling,1) ~= esn.nOutputUnits
    error('the size of the teacherScaling does not match the number of output units');
end
if size(esn.teacherShift,1) ~= esn.nOutputUnits
    error('the size of the teacherShift does not match the number of output units');
end
if length(esn.timeConstants) ~= esn.nInternalUnits
    error('timeConstants must be given as column vector of length esn.nInternalUnits');
end
if ~strcmp(esn.learningMode,'offline_singleTimeSeries') &&...
        ~strcmp(esn.learningMode,'offline_multipleTimeSeries') && ...
        ~strcmp(esn.learningMode,'online')
    error('learningMode should be either "offline_singleTimeSeries", "offline_multipleTimeSeries"
or "online" ') ;
end
if ~((strcmp(esn.outputActivationFunction,'identity') && ...
        strcmp(esn.inverseOutputActivationFunction,'identity')) || ...
        (strcmp(esn.outputActivationFunction,'tanh') && ...
        strcmp(esn.inverseOutputActivationFunction,'atanh')) || ...
        (strcmp(esn.outputActivationFunction,'sigmoid01') && ...
        strcmp(esn.inverseOutputActivationFunction,'sigmoid01_inv')))  ...
error('outputActivationFunction and inverseOutputActivationFunction do not match');
end
```

- ✓ Εκπαίδευση ESN

```
function [trained_esn, stateCollection] = ...
    train_esn(trainInput, trainOutput , esn, nForgetPoints)
% TRAIN_ESN Trains the output weights of an ESN
% In the offline case, it computes the weights using the method
% esn.methodWeightCompute(for ex linear regression using pseudo-inverse)
% In In the online case, RLS is being used.
%
% inputs:
% trainInput = input vector of size nTrainingPoints x nInputDimension
% trainOutput = teacher vector of size nTrainingPoints x
% nOutputDimension
% esn = an ESN structure, through which we run our input sequence
% nForgetPoints - the first nForgetPoints will be disregarded


%
% outputs:
% trained_esn = an Esn structure with the option trained = 1 and
% outputWeights set.
% stateCollection = matrix of size (nTrainingPoints-nForgetPoints) x
% nInputUnits + nInternalUnits
% stateCollectMat(i,j) = internal activation of unit j after the
% (i + nForgetPoints)th training point has been presented to the network
% teacherCollection is a nSamplePoints * nOuputUnits matrix that keeps
% the expected output of the ESN
% teacherCollection is the transformed(scaled, shifted etc) output see
% compute_teacher for more documentation


%
% Created April 30, 2006, D. Popovici
% Copyright: Fraunhofer IAIS 2006 / Patent pending
% Revision 1, June 30, 2006, H. Jaeger
% Revision 2, Feb 23, 2007, H. Jaeger



trained_esn = esn;
switch trained_esn.learningMode
    case 'offline_singleTimeSeries'
```

```
        % trainInput and trainOutput each represent a single time series in
        % an array of size sequenceLength x sequenceDimension
        if strcmp(trained_esn.type, 'twi_esn')
            if size(trainInput,2) > 1
                trained_esn.avDist = ...
                    mean(sqrt(sum(((trainInput(2:end,:) - trainInput(1:end - 1,:))').^2)));
            else
                trained_esn.avDist = mean(abs(trainInput(2:end,:) - trainInput(1:end - 1,:)));
            end
        end
        stateCollection = compute_statematrix(trainInput, trainOutput, trained_esn, nForgetPoints)
;
        teacherCollection = compute_teacher(trainOutput, trained_esn, nForgetPoints) ;
        trained_esn.outputWeights = feval(trained_esn.methodWeightCompute, stateCollection,
teacherCollection) ;


    case  'offline_multipleTimeSeries'
        % trainInput and trainOutput each represent a collection of K time
        % series, given in cell arrays of size K x 1, where each cell is an
        % array of size individualSequenceLength x sequenceDimension


        % compute total size of sample points to be used
        sampleSize = 0;
        nTimeSeries = size(trainInput, 1);
        for i = 1:nTimeSeries
            sampleSize = sampleSize + size(trainInput{i,1},1) - max([0, nForgetPoints]);
        end


        % collect input+reservoir states into stateCollection
        stateCollection = zeros(sampleSize, trained_esn.nInputUnits + trained_esn.nInternalUnits);
        collectIndex = 1;
        for i = 1:nTimeSeries
            if strcmp(trained_esn.type, 'twi_esn')
                if size(trainInput{i,1},2) > 1
                    trained_esn.avDist = ...
                        mean(sqrt(sum(((trainInput{i,1}(2:end,:) - trainInput{i,1}(1:end -
1,:))').^2)));
                else
                    trained_esn.avDist = mean(abs(trainInput{i,1}(2:end,:) - trainInput{i,1}(1:end
- 1,:)));
                end
            end
            stateCollection_i = ...
                compute_statematrix(trainInput{i,1}, trainOutput{i,1}, trained_esn,
nForgetPoints);
            l = size(stateCollection_i, 1);
            stateCollection(collectIndex:collectIndex+l-1, :) = stateCollection_i;
            collectIndex = collectIndex + l;
        end


        % collect teacher signals (including applying the inverse output
        % activation function) into teacherCollection
        teacherCollection = zeros(sampleSize, trained_esn.nOutputUnits);
        collectIndex = 1;
        for i = 1:nTimeSeries
            teacherCollection_i = ...
                compute_teacher(trainOutput{i,1}, trained_esn, nForgetPoints);
            l = size(teacherCollection_i, 1);
            teacherCollection(collectIndex:collectIndex+l-1, :) = teacherCollection_i;
            collectIndex = collectIndex + l;
        end


        % compute output weights
        trained_esn.outputWeights = ...
            feval(trained_esn.methodWeightCompute, stateCollection, teacherCollection) ;



    case 'online'
        nSampleInput = length(trainInput);
        stateCollection = zeros(nSampleInput, trained_esn.nInternalUnits +
trained_esn.nInputUnits);
        SInverse = 1 / trained_esn.RLS_delta * eye(trained_esn.nInternalUnits +
trained_esn.nInputUnits) ;
        totalstate = zeros(trained_esn.nTotalUnits,1);
        internalState = zeros(trained_esn.nInternalUnits,1) ;
        error = zeros(nSampleInput , 1) ;
        weights = zeros(nSampleInput , 1) ;
        for iInput = 1 : nSampleInput
            if trained_esn.nInputUnits > 0
                in = [diag(trained_esn.inputScaling) * trainInput(iInput,:)' + esn.inputShift];  %
in is column vector
            else in = [];
            end
```

```
            %write input into totalstate
            if esn.nInputUnits > 0
                totalstate(esn.nInternalUnits+1:esn.nInternalUnits+esn.nInputUnits) = in;
            end

            % update totalstate except at input positions

            % the internal state is computed based on the type of the network
            switch esn.type
                case 'plain_esn'
                    typeSpecificArg = [];
                case 'leaky_esn'
                    typeSpecificArg = [];
                case 'twi_esn'
                    if  esn.nInputUnits == 0
                        error('twi_esn cannot be used without input to ESN');
                    end
                    typeSpecificArg = esn.avDist;
            end
            internalState = feval(trained_esn.type , totalstate, trained_esn, typeSpecificArg ) ;
            netOut =
feval(trained_esn.outputActivationFunction,trained_esn.outputWeights*[internalState;in]);
            totalstate = [internalState;in;netOut];
            state = [internalState;in] ;
            stateCollection(iInput, :) = state';
            phi = state' * SInverse ;
            %            u = SInverse * state ;
            %            k = 1 / (lambda + state'*u)*u ;
            k = phi'/(trained_esn.RLS_lambda + phi * state );
            e = trained_esn.teacherScaling * trainOutput(iInput,1) + trained_esn.teacherShift -
netOut(1) ;
            % collect the error that will be plotted
            error(iInput , 1 ) = e*e ;
            % update the weights
            trained_esn.outputWeights(1,:) = trained_esn.outputWeights(1,:) + (k*e)' ;
            % collect the weights for plotting
            weights(iInput , 1) = sum(abs(trained_esn.outputWeights(1,:))) ;
            %            SInverse = 1 / lambda * (SInverse - k*(state' * SInverse)) ;
            SInverse = ( SInverse - k * phi ) / trained_esn.RLS_lambda ;
        end



        figure;
        plot(error) ;
        title('instant square training error') ;
        figure;
        plot(weights) ;
        title('weights') ;
end

trained_esn.trained = 1 ;
```

- ✓ Έλεγχος ESN

```
function outputSequence = test_esn(inputSequence, esn, nForgetPoints, varargin)
% test_esn runs a trained ESN on a particular inputSequence


% input args:
% inputSequence: is a nTrainingPoints * nInputUnits matrix that contains
% the input we will run the esn on
% esn: the trained ESN structure
% nForgetPoints: nr of initial time points to be discarded
%
% optional input argument:
% there may be one optional input, the starting vector by which the esn is
% started. The starting vector must be given as a column vector of
% dimension esn.nInternalUnits + esn.nOutputUnits + esn.nInputUnits  (that
% is, it is a total state, not an internal reservoir state). If this input
% is desired, call test_esn with fourth input 'startingState' and fifth
% input the starting vector.
%
% ouput:
% outputSequence is an array of size (size(inputSequence, 1)-nForgetPoints)
% x esn.nOutputUnits
```

```
% Created April 30, 2006, D. Popovici
% Copyright: Fraunhofer IAIS 2006 / Patent pending
% revision 1, June 6, 2006, H. Jaeger
% revision 2, June 23, 2007, H. Jaeger (added optional start state input)


if esn.trained == 0
    error('The ESN is not trained. esn.trained = 1 for a trained network') ;
end


if nargin == 3 % case where no starting state is specified
    stateCollection = compute_statematrix(inputSequence, [], esn, nForgetPoints) ;
else % case where the last (fourth and fifth) input argument gives a starting vector
    args = varargin;
    nargs= length(args);
    for i=1:2:nargs
        switch args{i},
            case 'startingState', totalstate = args{i+1} ;
            otherwise error('the option does not exist');
        end
    end
    stateCollection = ...
        compute_statematrix(inputSequence, [], esn, nForgetPoints, 'startingState', totalstate) ;
end


outputSequence = stateCollection * esn.outputWeights' ;
%%%% scale and shift the outputSequence back to its original size
nOutputPoints = length(outputSequence(:,1)) ;
outputSequence = feval(esn.outputActivationFunction, outputSequence);
outputSequence = outputSequence - repmat(esn.teacherShift',[nOutputPoints 1]) ;
outputSequence = outputSequence / diag(esn.teacherScaling) ;
```

- ✓ Υπολογισμός σφαλμάτων

```
function err = compute_error(estimatedOutput, correctOutput)
% Computes the NRMSE between estimated and correct ESN outputs.
%
% input arguments:
% estimatedOutput: array of size N1 x outputDimension, containing network
% output data. Caution: it is assumed that these are un-rescaled and
% un-shifted, that is, the transformations from the original data format
% via esn.teacherScaling and esn.teacherShift are undone. This happens
% automatically when the estimatedOutput was obtained from calling
% test_esn.
%
% correctOutput: array of size N2 x outputDimension, containing the
% original teacher data.
%
% output:
% err: a row vector of NRMSE's, each corresponding to one of the output
% dimensions.
%
% If length(correctOutput) > length(estimatedOutput), the first
% elements from correctOutput are deleted. This accounts for cases where
% some (nForgetPoints many) initial transient data points were cancelled
% from estimatedOutput, as occurs in calls to test_esn.
%
% Version 1.0, June 6, 2006, H. Jaeger
% Copyright: Fraunhofer IAIS 2006 / Patents pending



nEstimatePoints = length(estimatedOutput) ;

nForgetPoints = length(correctOutput) - nEstimatePoints ;

correctOutput = correctOutput(nForgetPoints+1:end,:) ;

correctVariance = var(correctOutput) ;
meanerror = sum((estimatedOutput - correctOutput).^2)/nEstimatePoints ;

err = (sqrt(meanerror./correctVariance)) ;
```

✓ Παραγωγή χρονοσειρών NARMA

```
systemOrder = 2 ; % set the order of the NARMA equation
sequenceLength=5000;
[inputSequence outputSequence] = generate_myNARMA_sequence(sequenceLength , systemOrder) ;
a=outputSequence
 dlmwrite('NAR_2.dat',a)
```

.
.
.

```
systemOrder = 30 ; % set the order of the NARMA equation
sequenceLength=5000;
[inputSequence outputSequence] = generate_myNARMA_sequence(sequenceLength , systemOrder) ;
a=outputSequence
 dlmwrite('NAR_30.dat',a)
```

✓ Υπολογισμός εκθετών Lyapunov

```
function lam = lyapunov(x,dt)
% calculate lyapunov coefficient of time series


[ndata nvars]=size(x);


N2 = floor(ndata/2);
N4 = floor(ndata/4);
TOL = 1.0e-6;


exponent = zeros(N4+1,1);


for i=N4:N2  % second quartile of data should be sufficiently evolved
   dist = norm(x(i+1,:)-x(i,:));
   indx = i+1;
   for j=1:ndata-5
       if (i ~= j) && norm(x(i,:)-x(j,:))<dist
           dist = norm(x(i,:)-x(j,:));
           indx = j; % closest point!
       end
   end
   expn = 0.0; % estimate local rate of expansion (i.e. largest eigenvalue)
   for k=1:5
       if norm(x(i+k,:)-x(indx+k,:))>TOL && norm(x(i,:)-x(indx,:))>TOL
           expn = expn + (log(norm(x(i+k,:)-x(indx+k,:)))-log(norm(x(i,:)-x(indx,:))))/k;
       end
   end
   exponent(i-N4+1)=expn/5;
end


plot(exponent);  % plot the estimates for each initial point (fairly noisy)


sum=0;  % now, calculate the overal average over N4 data points ...
for i=1:N4+1
    sum = sum+exponent(i);
end


lam=sum/((N4+1)*dt);  % return the average value
% if lam > 0, then system is chaotic
```

5.2 TSToolbox

✓ Εύρεση χρόνου καθυστέρησης

```
function rs = amutual(s, maxtau, bins)


%tstoolbox/@signal/amutual
%   Syntax:
%     * amutual(s, maxtau, bins)
%
%   Input arguments:
%     * maxtau - maximal delay (should be much smaller than the lenght of
%       s) (optional)
%     * bins - number of bins used for histogram calculation (optional)
%
%   Auto mutual information function for real scalar signals, can be used
%   to determine a proper delay time for time-delay reconstruction. The
%   default value for maxtau is 25% of the input signal's length. The
%   default number of bins is 128.
%
%               [missing equation, see html/pdf documentation]
%
% Copyright 1997-2001 DPI Goettingen, License http://www.physik3.gwdg.de/tstool/gpl.txt




error(nargchk(1,3,nargin));



if (ndim(s) > 1) | (~isreal(data(s)))
    error('Input signal must be a scalar time series');
end


N = dlens(s,1);


if nargin < 2
    maxtau = ceil(N/40);    % default : use 2.5 percent of input signals length
end


if nargin < 3
    bins = 128;
end


x = data(s);
[y,i] = sort(x);        % transformation to rang values
y(i) = (0:N-1)/N;       % and scaling to [0 1)


x = amutual(y, maxtau, bins);
rs = signal(core(x), s);    % special constructor calling syntax for working routines
a = getaxis(s, 1);
a = setfirst(a, 0);
rs = setaxis(rs, 1, a);
rs = setyunit(rs, unit('Bit'));
rs = addhistory(rs, ['Auto mutual information']);
rs = addcommandlines(rs, 's = amutual(s', maxtau, bins);
```

✓ Script για τη δημιουργία των ελκυστών του σχήματος σελ. 45

```
s=signal('colpitts.dat','ascii');
a= amutual(s,5);
e1 = embed(s, 3, 1);
subplot(2,2,1);
view(e1)
title('t_l=1')
e2 = embed(s, 3, 4);
subplot(2,2,2);
view(e2)
title('t_l=4')
subplot(2,2,3);
e3 = embed(s, 3, 10);
view(e3)
title('t_l=10')
subplot(2,2,4);
```

```
e4 = embed(s, 3, 20);
view(e4)
title('t_l=20')
```

- ✓ Script για τη δημιουργία των διαγραμμάτων του σχήματος σελ. 48

```
a=[mTr2,mTr4,mTr6,mTr8,mTr10,mTr12,mTr14,mTr16,mTr18,mTr20,mTr22,mTr24,mTr26,mTr28,mTr30]

b=[mTes2,mTes4,mTes6,mTes8,mTes10,mTes12,mTes14,mTes16,mTes18,mTes20,mTes22,mTes24,mTes26,mTes28,m
Tes30]
c=[vTr2,vTr4,vTr6,vTr8,vTr10,vTr12,vTr14,vTr16,vTr18,vTr20,vTr22,vTr24,vTr26,vTr28,vTr30]

d=[vTes2,vTes4,vTes6,vTes8,vTes10,vTes12,vTes14,vTes16,vTes18,vTes20,vTes22,vTes24,vTes26,vTes28,v
Tes30]


x=[2:2:30];
    figure
    subplot(2,2,1)
    plot(x,a,'b-o')
    title('Mean train error')
    xlabel('NARMA Order')
    ylabel('Error')
    subplot(2,2,2)
    plot(x,b,'r--o')
    title('Mean test error')
    xlabel('NARMA Order')
    ylabel('Error')
    subplot(2,2,3)
    plot(x,c,'g-*')
    title('Var train error')
    xlabel('NARMA Order')
    ylabel('Error')
    subplot(2,2,4)
    plot(x,d,'m--*')
    title('Var test error')
    xlabel('NARMA Order')
    ylabel('Error')

figure
plot(x,a,'b-o')
hold on
plot(x,b,'r--o')
title('Comparison of mean train error (blue) and mean test error (red)')
xlabel('NARMA Order')
ylabel('Error')
```

6. Βιβλιογραφία

Α) Ξένη βιβλιογραφία

- Eugen Diaconescu (2008), *The use of NARX Neural Networks to predict Chaotic Time Series ,* Faculty University of Pitesti Targu din Vale, Nr. 1 Romania

- Ali Rodan, Peter Tino (2010), *Minimum complexity echo state network*

- Elizabeth Bradley (2010) *Chaotic dynamics,* University of Colorado

- R.J.Frank, N.Davey, S.P.Hunt (2001), *Time Series Prediction and Neural Networks*
  Department of Computer Science, University of Hertfordshire, Hatfield, UK

- B. Cessac (2009), *A view of Neural Networks as dynamical systems*

- Gilles Wainrib (2015), *A local Echo State Property through the largest Lyapunov exponent,* Ecole Normale Superieure, Departement d'Informatique, Paris, France.
- Michael T. Rosenstein, James J. Collins, and Carlo J. De Luca (1993), *Practical method for calculating largest Lyapunov exponents from small data sets*

- Ulrich Parlitz (1998), *Nonlinear time-series analysis,* Dittes Physikalisches Institut, Universitat Gottingen

Β) Ελληνική βιβλιογραφία

- Γεώργιος Π. Παύλος, Μιχαήλ Αθανασίου (2002), *Εισαγωγή στη θεωρία πολυπλοκότητας και στα πολύπλοκα συστήματα*
- Πάνος Αργυράκης (2001), *Νευρωνικά δίκτυα και εφαρμογές*

Γ) Ιστοσελίδες

- http://wikipedia.org
- http://psych.utoronto.ca/users/reingold/courses/ai/cache/neural4.html
- https://visualstudiomagazine.com/articles/2014/08/01/batch-training.aspx

Δ) ESN Toolbox

D. Popovici, Fraunhofer, H. Jaeger, University of Gehnt, Belgium