

Ατομική Διπλωματική Εργασία

**ΠΡΟΒΛΕΨΗ ΖΗΤΗΣΗΣ ΦΥΣΙΚΟΥ ΑΕΡΙΟΥ ΜΕ ΤΗ ΧΡΗΣΗ
ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ**

Κυπριανός Χατζηδημητρίου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ



ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Δεκέμβριος 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόβλεψη ζήτησης φυσικού αερίου με τη χρήση Νευρωνικών Δικτύων

Κυπριανός Χατζηδημητρίου

Επιβλέπων Καθηγητής
Δρ. Χριστοδούλου Χρίστος

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

Δεκέμβριος 2010

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή, Δρ. Χρίστο Χριστοδούλου, που με εμπιστεύθηκε για την εκπόνηση αυτής της διπλωματικής εργασίας. Επίσης τον ευχαριστώ για τη συνεργασία, την υπομονή και την καθοδήγηση που μου πρόσφερε κατά τη διάρκεια της εργασίας αυτής.

Πιστεύω ότι χωρίς τη δική του στήριξη και εποπτεία ίσως θα μου ήταν πολύ δύσκολο να φέρω εις πέρας την ολοκλήρωση αυτής της διπλωματικής εργασίας.

Σας ευχαριστώ,

Κυπριανός Χατζηδημητρίου

Περίληψη

Η διπλωματική εργασία αυτή εστιάζεται στη χρήση νευρωνικών δικτύων για την επίλυση ενός σημαντικού προβλήματος της καθημερινότητας του πραγματικού κόσμου. Πιο συγκεκριμένα μελετήσαμε την αποδοτικότητα των νευρωνικών δικτύων για την πρόβλεψη της ζήτησης του φυσικού αερίου.

Τα δεδομένα που χρησιμοποιήσαμε καλύπτουν ένα φάσμα επτά ετών και προέρχονται από μεγάλη Βρετανική εταιρία παροχής φυσικού αερίου επομένως έχουμε να κάνουμε με πραγματικά δεδομένα. Έτσι δημιουργήσαμε ένα εύχρηστο εργαλείο – Gas prediction Software- το οποίο χρησιμοποιεί νευρωνικά δίκτυα για την πρόβλεψη της ζήτησης του φυσικού αερίου. Συγκεκριμένα στην έρευνα αυτή επικεντρωθήκαμε σε δύο αρχιτεκτονικές νευρωνικών δικτύων, τα δίκτυα συναρτήσεων αξονικών βάσεων – Radial Basis Functions (RBF) – και τα δίκτυα με ανατροφοδότηση από την έξοδο – Jordan Recurrent Neural Networks-. Όσον αφορά την πρώτη αρχιτεκτονική υλοποιήσαμε δύο παραλλαγές της. Στην πρώτη παραλλαγή οι αρχικές θέσεις των κέντρων καθορίζονταν τυχαία μέσα από το σύνολο των δεδομένων εκπαίδευσης σ' αντίθεση με τη δεύτερη όπου οι θέσεις καθορίζονταν μέσω αλγορίθμου ομαδοποίησης και συγκεκριμένα χάρτες αυτο-οργάνωσης (Self organized Maps) Kohonen . Η δεύτερη αρχιτεκτονική βασίζεται στην παραδοχή ότι η επόμενη έξοδος καθορίζεται από τη προηγούμενη καθώς επίσης και από τις τρέχουσες εισόδους. Έτσι το δεύτερο δίκτυο στο οποίο πειραματιστήκαμε ήταν ένα πολυεπίπεδο perceptron στο οποίο η τρέχουσα έξοδος ήταν μια από τις εισόδους για την επόμενη χρονική στιγμή. Το δίκτυο αυτό εκπαιδεύτηκε με τη χρήση μιας παραλλαγής του αλγορίθμου ανάστροφης μετάδοσης του λάθους (Back Propagation) , τον αλγόριθμο Back Propagation Through Time (BPTT) .

Μετά από πειραματισμούς και πολυπληθείς εκπαιδεύσεις των δικτύων βρήκαμε τις τιμές εκείνες των παραμέτρων οι οποίες μας δίνουν το υψηλότερο ποσοστό επιτυχίας. Τα αποτελέσματα που προέκυψαν ήταν αρκετά ικανοποιητικά για τη φύση του προβλήματος. Συγκεκριμένα επιτύχαμε ποσοστό επιτυχίας 85.057% για το δίκτυο συναρτήσεων αξονικών βάσεων - RBF[7][8][9], 83.56% για το υβριδικό δίκτυο Kohonen SOM [11] – RBF[7][8][9] και 98.034% για το δίκτυο με ανατροφοδότηση τύπου Jordan όπου για την πρόβλεψη της ζήτησης τη χρονική στιγμή $N+1$, χρησιμοποιούσαμε την προηγούμενη επιθυμητή τιμή της ζήτησης και όχι αυτή που είχε προβλεφθεί. Όταν κάναμε τις κατάλληλες τροποποιήσεις έτσι ώστε για την πρόβλεψη της επόμενης τιμής ζήτησης να χρησιμοποιούμε αυτή που είχε προβλεφθεί το ποσοστό αυτό μειώθηκε στο 94.013%.

Τέλος, από τα παραπάνω καθώς και απ' όλα όσα περιγράφονται στην αναφορά που ακολουθεί επιβεβαιώνουν την αποδοτικότητα των νευρωνικών δικτύων για το συγκεκριμένο πρόβλημα.

Περιεχόμενα

Κεφάλαιο 1	Εισαγωγή.....	1
	1.1 Περιγραφή προβλήματος	2
	1.2 Προηγούμενες έρευνες	3
	1.3 Περιγραφή μεθοδολογίας	5
Κεφάλαιο 2	Μεθοδολογία Έρευνας.....	7
	2.1 Θεωρητικό υπόβαθρο	8
	2.1.1 Δίκτυα Συναρτήσεων Αξονικών Βάσεων	8
	2.1.1.1 Κύρια χαρακτηριστικά Δικτύων	8
	2.1.1.2 Αλγόριθμοι εκπαίδευσης	12
	2.1.1.2.1 Αλγόριθμος με σταθερά κέντρα	12
	2.1.1.2.2 Αλγόριθμος με μεταβλητά κέντρα	12
	2.1.1.3 Μέθοδοι επιλογής των κέντρων	14
	2.1.1.4 Σύγκριση RBF με MLP	15
	2.1.2 Χάρτες Αυτό-οργάνωσης	16
	2.1.2.1 Αρχιτεκτονική του Δικτύου	16
	2.1.2.2 Λεπτομερής περιγραφή του αλγορίθμου	17
	2.1.3 Δίκτυα Ανατροφοδότησης	20
	2.1.3.1 Αρχιτεκτονική	21
	2.1.3.2 Εκπαίδευση δικτύου με ανάδραση	25
	2.2 Επεξεργασία Δεδομένων	29
	2.3 Προτεινόμενη Λύση	33
Κεφάλαιο 3	Αναλυτικός Σχεδιασμός – Υλοποίηση	39
	3.1 Περιγραφή	40
	3.2 Σχεδιασμός	40
	3.2.1 Διαγράμματα UML	41
	3.2.2 Επεκτασιμότητα	44
	3.3 Υλοποίηση	45
	3.3.1 Επεξήγηση κλάσεων	45

Κεφάλαιο 4	Επίδειξη Χρήσης Εργαλείου Εκτίμησης Ζήτησης.....	53
4.1	Γενική περιγραφή	54
4.2	Εισαγωγή δεδομένων εκπαίδευσης – ελέγχου	54
4.3	Αλγόριθμοι εκπαίδευσης	56
4.3.1	Αλγόριθμός μεταβλητών κέντρων σε δίκτυο RBF	56
4.3.2	Αλγόριθμος back propagation through time	58
4.4	Λοιπές Λειτουργίες	59
Κεφάλαιο 5	Πειράματα - Αποτελέσματα.....	61
5.1	Εισαγωγή	62
5.2	Αλλαγές στις παραμέτρους του δικτύου RBF	63
5.2.1	Αλλαγές στους ρυθμούς μάθησης	63
5.2.2	Αλλαγές στον αριθμό των κέντρων	68
5.3	Αλλαγές στις παραμέτρους του υβριδικού δικτύου SOM-RBF	73
5.3.1	Αλλαγές στους ρυθμούς μάθησης	74
5.4	Αλλαγές στα σύνολα δεδομένων εκπαίδευσης και επαλήθευσης	79
5.5	Αλλαγές στις διαστάσεις του πλέγματος του αλγορίθμου Kohonen SOM	86
5.6	Αλλαγές στις παραμέτρους του δικτύου με ανάρδραση-BPTT	88
5.6.1	Αλλαγές στο ρυθμό μάθησης	89
5.6.2	Αλλαγές στον αριθμό των κρυφών νευρώνων	93
5.6.3	Αλλαγές στον αριθμό των χρονικών βημάτων	98
5.6.4	Αλλαγή των τιμών της χρονικής σταθεράς (time constant)	102
5.6.5	Τροποποίηση στον τρόπο υπολογισμού της επόμενης τιμής της ζήτησης	104
5.7	Σύγκριση επιθυμητού - πραγματικού αποτέλεσμα	106
5.8	Χρήση του Μέσου Απόλυτου Σχετικού Σφάλματος (MARE) για τον υπολογισμό του σφάλματος	118

Κεφάλαιο 6 Συμπεράσματα	122
6.1 Συμπεράσματα	123
6.2 Μελλοντική Εργασία	127
 Βιβλιογραφία	 128
 Παραρτήματα.....	 A-1
Παράρτημα Α: Κώδικας για ανάγνωση και εγγραφή δεδομένων	A-1
Παράρτημα Β: Κώδικας υλοποίησης δικτύου RBF	B-1
Παράρτημα Γ: Κώδικας υλοποίησης δικτύου Jordan	Γ-1
Παράρτημα Δ: Κώδικας υλοποίησης διεπαφής – GUI	Δ-1
Παράρτημα Ε: Δεδομένα εκπαίδευσης και επαλήθευσης	E-1

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή Προβλήματος	2
1.2 Προηγούμενες έρευνες	3
1.3 Περιγραφή μεθοδολογίας	5

1.3 Περιγραφή Προβλήματος

Αυτή η διπλωματική εργασία έχει ως σκοπό την πρόβλεψη της ζήτησης του φυσικού αερίου με τη χρήση Νευρωνικών Δικτύων. Όπως είναι γνωστό το φυσικό αέριο αποτελεί μια από τις κύριες πηγές ενέργειας για τις πλείστες Ευρωπαϊκές χώρες. Έτσι και η χώρα μας ως μέλος της Ευρωπαϊκής Ένωσης σύντομα θα έχει την ευχέρεια να χρησιμοποιεί το φυσικό αέριο στα συστήματα θέρμανσης και συνεπώς και ως πηγή ενέργειας. Μεγάλες εταιρίες είναι φυσικό ν' ασχοληθούν μ' αυτά τα συστήματα και να παρέχουν τις υπηρεσίες τους στους κατοίκους της χώρας μας. Σίγουρα τα συστήματα θ' αποτελούνται από πολύπλοκες και τεράστιες σωληνώσεις και θα μεταφέρουν το φυσικό αέριο από τον παροχέα στις οικίες των τελικών χρηστών.

Το φυσικό αέριο έχει το μειονέκτημα να κινείται αργά μέσα στις διασωληνώσεις. Επομένως πάντοτε πρέπει να υπολογίζεται και αυτός ο επιπλέον χρόνος που απαιτείται για να φτάσει στον προορισμό του. Έτσι η σωστή και έγκαιρη πρόβλεψη της ζήτησης του φυσικού αερίου θα είναι ευεργετική για τις εταιρίες που θ' ασχοληθούν με τη μεταφορά του προς τις οικίες των χρηστών. Πιο συγκεκριμένα με τη σωστή πρόβλεψη οι εταιρίες θα μειώσουν τα έξοδα και θ' αυξήσουν την παραγωγικότητα τους αφού θα διοχετεύουν τη σωστή ποσότητα φυσικού αερίου με αποτέλεσμα ν' αποφεύγουν τις περιττές σπατάλες. Επιπλέον δεν θα παρουσιάζονται προβλήματα δυσαρεστημένων πελατών και έτσι οι πελάτες θα συνεχίζουν τη συνεργασία τους με την εταιρία. Στην αντίθετη περίπτωση όπου θ' απουσιάζει η έγκαιρη πρόβλεψη της ζήτησης, σίγουρα οι εταιρίες στην προσπάθεια τους να ικανοποιήσουν πλήρως τις ανάγκες των πελατών τους θα διοχετεύουν μεγαλύτερη ποσότητα αερίου, με αποτέλεσμα να έχουν πολύ περισσότερα αχρείαστα έξοδα.

Όμως η πρόβλεψη της ζήτησης δεν είναι εύκολη διαδικασία αφού η ζήτηση του φυσικού αερίου εξαρτάται από αρκετούς παράγοντες. Για την ορθή πρόβλεψη πρέπει να είναι γνωστές οι καιρικές συνθήκες για τη συγκεκριμένη μέρα όπως επίσης και η εποχή του χρόνου στην οποία βρισκόμαστε. Επιπλέον η πρόβλεψη επηρεάζεται από τη μέρα της εβδομάδος καθώς επίσης και την ώρα της συγκεκριμένης μέρας. Για παράδειγμα το Σαββατοκύριακο όπου οι περισσότεροι δεν δουλεύουν αναμένεται να έχουμε μεγαλύτερη ζήτηση σε σύγκριση με τις καθημερινές μέρες. Επίσης στις

καθημερινές αναμένεται να υπάρχει μεγαλύτερη ζήτηση τις απογευματινές και νυχτερινές ώρες αφού εκείνες τις ώρες οι χρήστες θα σχολάσουν από τις εργασίες τους και θα βρίσκονται στις οικίες τους.

Τέλος, μπορούμε να πούμε ότι αν τα πιο πάνω συνδυαστούν με αποδοτικό τρόπο θα μπορούμε να έχουμε ορθή και έγκαιρη πρόβλεψη της ζήτησης του φυσικού αερίου πράγμα που θα ικανοποιεί τους χρήστες αφού θα έχουν την ποσότητα του φυσικού αερίου που επιθυμούν καθώς επίσης και τις εταιρίες που θα μειώσουν τα έξοδα και θα αυξήσουν την παραγωγικότητα τους.

1.2 Προηγούμενες έρευνες

Στο παρελθόν έχουν γίνει αρκετές μελέτες και χρησιμοποιήθηκαν διάφορες μέθοδοι είτε μέσω νευρωνικών δικτύων είτε μέσω άλλων στατιστικών μεθόδων [1], για την επίλυση αυτού του προβλήματος. Οι έρευνες αυτές έδειξαν ότι τα νευρωνικά δίκτυα είχαν μεγαλύτερη επιτυχία σε σύγκριση με τις στατιστικές μεθόδους. Συγκεκριμένα οι προβλέψεις έγιναν σε δεδομένα ζήτησης με χρονικό παράθυρο δυο ετών. Σ' αυτή την έρευνα έκαναν τρία διαφορετικά είδη προβλέψεων. Συγκεκριμένα έκαναν καθημερινή, εβδομαδιαία και μηνιαία πρόβλεψη. Τα αποτελέσματα της έρευνας παρουσιάζονται στον πιο κάτω πίνακα:

Είδος Πρόβλεψης	Ποσοστό Αποτυχίας	
	Νευρωνικά Δίκτυα	Στατιστικές Μέθοδοι
Καθημερινή	4,04 %	4,56 %
Εβδομαδιαία	7,04 %	10,64 %
Μηνιαία	7,73 %	21,64 %

Πίνακας 1.2.1: Σύγκριση NN με στατιστικές μεθόδους [1]

Σε μια άλλη έρευνα που έγινε από τους Musilek et al. [2], έγινε υβριδοποίηση των νευρωνικών δικτύων μαζί με το σύστημα προβλέψεων ELVIRA, το οποίο είναι ένα σύστημα λογικής ανάλυσης που συνδυάζει διάφορες στατιστικές μεθόδους, ανάλυση περιπτώσεων (case-based reasoning) και συστήματα κανόνων (rule based). Στην έρευνα αυτή οι ερευνητές περνούσαν τα δεδομένα από το νευρωνικό δίκτυο πριν να τα επεξεργαστούν με το στατιστικό μοντέλο. Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα αυτής της έρευνας:

Προβλέψεις	Ποσοστό Επιτυχίας	
	Αρχικό ποσοστό επιτυχίας	Νέο ποσοστό επιτυχίας
25 % απόκλιση από τις πραγματικές	95,7 %	99.8 %
10 % απόκλιση από τις πραγματικές	80,3 %	94 %

Πίνακας 1.2.2: Αποτελέσματα έρευνα Musilek et Al. [2]

Στην έρευνα που δημοσιεύτηκε το 2000 [9] οι ερευνητές στην προσπάθεια τους να φτιάξουν ένα σύστημα για πρόβλεψη ζήτησης φυσικού αερίου παρουσίασαν ένα σύστημα το οποίο «σπάζει» σε δύο στάδια. Συγκεκριμένα στο πρώτο στάδιο υπάρχουν δύο τεχνητά νευρωνικά δίκτυα (artificial Neural Networks) , το ένα από τα οποία είναι πολυστρωματικό δίκτυο εμπρόσθιας τροφοδότησης (multilayer feed forward network). Η εκπαίδευση των πιο πάνω έγινε με τον αλγόριθμο back propagation αλλά χρησιμοποίησαν μια διαφορετική στρατηγική για αλλαγή των βαρών [9]. Στο δεύτερο στάδιο υβριδοποίησαν τα δυο δίκτυα που αναπτύχθηκαν στο πρώτο στάδιο. Τα αποτελέσματα έδειξαν ότι το κάθε ένα από τα δύο δίκτυα του πρώτου σταδίου είχαν πολύ καλά αποτελέσματα στην πρόβλεψη. Επιπλέον με το συνδυασμό των πιο πάνω δικτύων η ακρίβεια της πρόβλεψης βελτιώθηκε περίπου κατά μέσο όρο 4% σε σύγκριση με την απόδοση του κάθε δικτύου ξεχωριστά.

Μία τέταρτη έρευνα που έγινε [3] αναφέρει ότι είναι πολύ σημαντική η ύπαρξη των δεδομένων παλαιότερων χρόνων όπου μπορούμε να τα χρησιμοποιήσουμε σαν βάση για την ορθή πρόβλεψη της ζήτησης του φυσικού αερίου. Σ' αυτό στηρίζεται και η προηγούμενη διπλωματική εργασία που έγινε από το Χρίστο Αιμίλιο Πρασίτη (2009) [4]. Συγκεκριμένα σ' αυτήν την εργασία χρησιμοποιήθηκαν πραγματικά δεδομένα από μια μεγάλη εταιρία παροχής φυσικού αερίου στη Μεγάλη Βρετανία. Τα δεδομένα αυτά καλύπτουν ένα φάσμα 7 ετών. Σ' αυτή την έρευνα αφού τα δεδομένα έτυχαν της κατάλληλης επεξεργασίας χρησιμοποιήθηκε ένα νευρωνικό δίκτυο τριών επιπέδων (3-layer Perceptron Neural Network) το οποίο εκπαιδεύτηκε με τον αλγόριθμο ανάστροφης μετάδοσης του λάθους (Back Propagation Learning Algorithm). Στη συγκεκριμένη έρευνα το πρόβλημα προσεγγίστηκε με δυο τεχνικές, με την τεχνική του κινητού παραθύρου στο χρόνο καθώς και μ' ένα στατικό δίκτυο. Στην πρώτη τεχνική –

στατικό δίκτυο - χρησιμοποιήθηκαν τα δεδομένα που παρουσιάζουν την θερμοκρασία ,την ταχύτητα των ανέμων, ο μήνας , η μέρα καθώς επίσης και η μέση ζήτηση , σε αντίθεση με την δεύτερη μέθοδο – του κινητού παράθυρου- όπου χρησιμοποιήθηκε μόνο η χρονοσείρα της ζήτησης του φυσικού αερίου καθημερινά ανά δύο ώρες. αποτελέσματα της προηγούμενης διπλωματικής εργασίας έδειξαν ότι η τεχνική του κινητού παραθύρου είχε καλύτερα αποτελέσματα σε σύγκριση με το στατικό δίκτυο. Συγκεκριμένα με το κινητό παράθυρο είχε επιτυχία 95,9 % , ενώ με το στατικό δίκτυο η επιτυχία ήταν 85,9 % .

1.3 Περιγραφή μεθοδολογίας

Σκοπός αυτής της διπλωματικής εργασίας είναι η κατασκευή ενός συστήματος που να προβλέπει γρήγορα και ορθά τη ζήτηση του φυσικού αερίου. Βασικά υλοποιήσαμε ένα νευρωνικό δίκτυο το οποίο στηρίζεται στα δίκτυα συναρτήσεων αξονικών βάσεων (radial basis functions - RBFs)[6][7][8]. Συγκεκριμένα χρησιμοποιήσαμε τον αλγόριθμο με μεταβλητά κέντρα έτσι ώστε να πετύχουμε όσο το δυνατό καλύτερα αποτελέσματα. Επιπλέον υλοποιήθηκε ένα MLP (Multilayer Perceptron) στο οποίο χρησιμοποιήσαμε ανατροφοδότηση της εισόδου από τη συγκεκριμένη έξοδο. Συγκεκριμένα υλοποιήσαμε ένα δίκτυο τύπου Jordan [5] στο οποίο η έξοδος δίνεται ως είσοδος για την επόμενη επανάληψη και πάνω σ' αυτό εφαρμόσαμε συγκεκριμένους αλγόριθμους μάθησης. Τα δεδομένα που χρησιμοποιήθηκαν είναι τα ίδια μ' αυτά της προηγούμενης διπλωματικής εργασίας. Όμως επειδή δεν υπήρχαν τα επεξεργασμένα δεδομένα, επεξεργαστήκαμε τα αρχικά δεδομένα που υπάρχουν και τα φέραμε στην επιθυμητή μορφή πριν τα παρουσιάσουμε ως είσοδο στο δίκτυο. Υλοποιώντας τα παραπάνω συγκρίναμε τα αποτελέσματα τόσο του RBF δικτύου όσο και του δικτύου με ανατροφοδότηση για να δούμε πιο είχε την καλύτερη απόδοση.

Τέλος στην προσπάθεια μας για βελτίωση των αποτελεσμάτων χρησιμοποιήσαμε τον αλγόριθμο ομαδοποίησης χάρτη αυτοοργάνωσης Kohonen [10] [11], SOM (self organizing map) για την εύρεση κοινών χαρακτηριστικών στα δεδομένα. Έτσι για κάθε ομάδα που δημιουργήθηκε τοποθετήσαμε στην συγκεκριμένη θέση ένα κέντρο. Ακολούθως εκπαιδεύσαμε το δίκτυο RBF με αυτά τα κέντρα και συγκρίναμε τα

αποτελέσματα. Στην ουσία χρησιμοποιήσαμε μια υβριδική μέθοδο συνδυάζοντας δίκτυα συναρτήσεων αξονικών βάσεων και χάρτες αυτό-οργάνωσης.

Κεφάλαιο 2

Μεθοδολογία Έρευνας

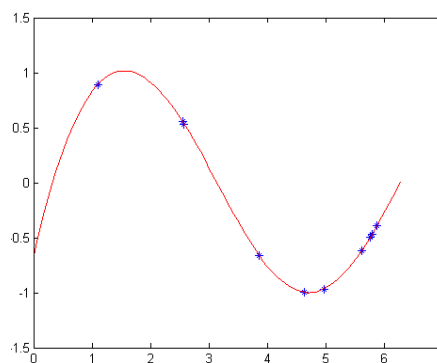
2.1 Θεωρητικό υπόβαθρο	8
2.1.1 Δίκτυα Συναρτήσεων Αξονικών Βάσεων	8
2.1.1.1 Κύρια χαρακτηριστικά Δικτύων	8
2.1.1.2 Αλγόριθμοι εκπαίδευσης	12
2.1.1.2.1 Αλγόριθμος με σταθερά κέντρα	12
2.1.1.2.2 Αλγόριθμος με μεταβλητά κέντρα	12
2.1.1.3 Μέθοδοι επιλογής των κέντρων	14
2.1.1.4 Σύγκριση RBF με MLP	15
2.1.2 Χάρτες Αυτό-οργάνωσης	16
2.1.2.1 Αρχιτεκτονική του Δικτύου	16
2.1.2.2 Λεπτομερής περιγραφή του αλγορίθμου	17
2.1.3 Δίκτυα Ανατροφοδότησης	20
2.1.3.1 Αρχιτεκτονική	21
2.1.3.2 Εκπαίδευση δικτύου με ανάδραση	25
2.2 Επεξεργασία Δεδομένων	29
2.3 Προτεινόμενη Λύση	33

2.1 Θεωρητικό υπόβαθρο

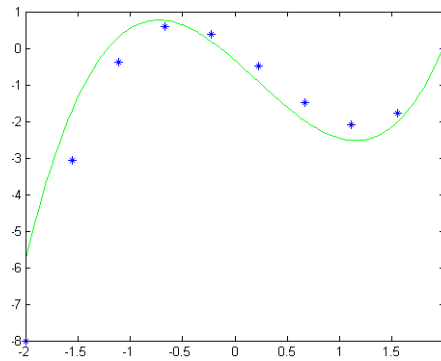
2.1.1 Δίκτυα Συναρτήσεων Αξονικών βάσεων (RBF)

2.1.1.1 Κύρια χαρακτηριστικά Δικτύων

Τα δίκτυα συναρτήσεων αξονικών βάσεων (RBF) στηρίζονται σε τεχνικές οι οποίες επιτυγχάνουν είτε ακριβή παρεμβολή είτε κατά προσέγγιση πρόβλεψη ενός αποτελέσματος από ένα σύνολο δεδομένων. Με τον όρο “παρεμβολή” το δίκτυο μετά την εκπαίδευση είναι σε θέση να μας δώσει ακριβή και ορθά αποτελέσματα για όλο το σύνολο των δεδομένων εκπαίδευσης. Αντίθετα όταν έχουμε κατά προσέγγιση εκτίμηση του αποτελέσματος το δίκτυο ταυτίζει τα δεδομένα εισόδου με τα επιθυμητά αποτελέσματα, αλλά δεν μας δίνει ακριβώς τα ορθά αποτελέσματα. Στο σχήμα 2.1.1.1 παρουσιάζεται η παρεμβολή. Όπως βλέπουμε το δίκτυο μας μαθαίνει “παπαγαλία” τα δεδομένα με αποτέλεσμα να μειώνεται η πιθανότητα να πάρουμε ορθή πρόβλεψη σε άγνωστα δεδομένα εισόδου. Στο σχήμα 2.1.1.2 βλέπουμε την κατά προσέγγιση εκτίμηση του αποτελέσματος. Όπως φαίνεται το δίκτυο μας γενικεύει τα δεδομένα με αποτέλεσμα να μπορεί να δώσει όσο το δυνατό πιο ορθή πρόβλεψη σε άγνωστα δεδομένα. Στα RBF όπως θα δούμε υπάρχουν δύο αλγόριθμοι για εκπαίδευση του δικτύου μας. Ο πρώτος αλγόριθμος επιτυγχάνει την παρεμβολή των δεδομένων και ο δεύτερος γενικεύει τα δεδομένα.



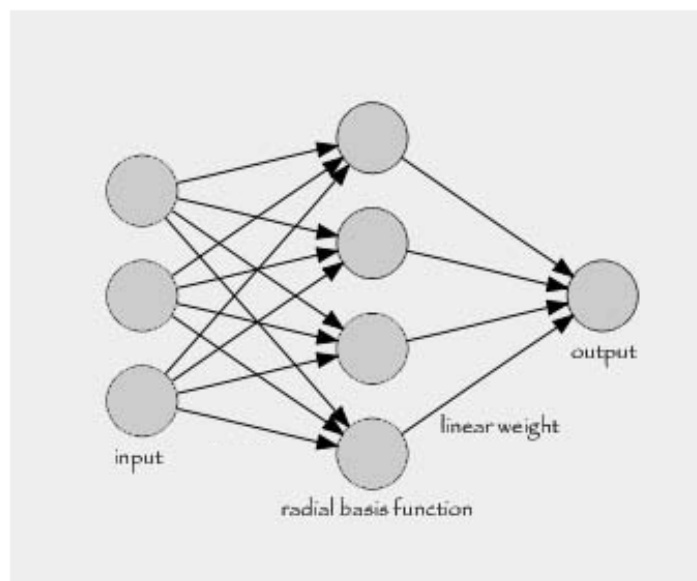
Σχήμα 2.1.1.1.1. Παρεμβολή δεδομένων



Σχήμα 2.1.1.1.2. Κατά προσέγγιση εκτίμηση

Κύρια συστατικά μέρη των δικτύων αξονικών βάσεων

Τα δίκτυα RBF αποτελούνται από μόνο τρία επίπεδα: το επίπεδο εισόδου, το επίπεδο εξόδου και το κρυφό επίπεδο. Στο σχήμα 2.1.1.1.3 παρουσιάζεται η αρχιτεκτονική ενός δικτύου RBF.



Σχήμα 2.1.1.1.3. Αρχιτεκτονική δικτύου αξονικών βάσεων

Επίπεδο εισόδου

Ο κάθε κόμβος του επιπέδου αυτού αντιστοιχεί σ' ένα δεδομένο εισόδου το οποίο συνδέεται μ' όλους τους κόμβους στο κρυφό επίπεδο.

Κρυφό Επίπεδο

Σε αντίθεση με τα δίκτυα πολλαπλών στρωμάτων MLP εδώ έχουμε **μόνο** ένα κρυφό επίπεδο. Ο κάθε κόμβος του επιπέδου αυτού αποτελείται από ένα κέντρο και μια συνάρτηση που ονομάζεται Basis Function. Το επίπεδο αυτό μετατρέπει τη μη γραμμική μορφή του διανύσματος εισόδου σε μια υψηλότερη διάσταση στην οποία τα διανύσματα γίνονται γραμμικά διαχωρίσιμα. Ο κάθε κόμβος του επιπέδου αυτού συνδέεται με το επίπεδο εξόδου με μια σύνδεση η οποία αντιστοιχεί σε κάποιο βάρος. Το βάρος αυτό ονομάζεται και συντελεστής (coefficient).

Επίπεδο εξόδου

Στο επίπεδο αυτό γίνεται ο πολλαπλασιασμός των εξόδων των κόμβων μαζί με τα αντίστοιχα βάρη. Η έξοδος προκύπτει από το άθροισμα των παραπάνω γινομένων.

Όπως αναφέραμε παραπάνω στο κρυφό επίπεδο υπάρχουν δύο κύριες παράμετροι. Η πρώτη παράμετρος είναι το κέντρο (R) το οποίο είναι της ίδιας διάστασης με το διάνυσμα εισόδου. Η δεύτερη παράμετρος είναι το βάρος (C) το οποίο δίνει κάποιο βάρος στη σύνδεση από το κρυφό επίπεδο προς το επίπεδο εξόδου.

Basis Function

Η συνάρτηση αυτή ισοδυναμεί με τη συνάρτηση ενεργοποίησης στα MLP. Αυτή που χρησιμοποιείται σε ευρεία κλίμακα και χρησιμοποιήθηκε και σ' αυτή την έρευνα είναι η γκαουσιανή συνάρτηση.

$$\Phi(r) = \exp(-r^2/2\sigma) \quad (1)$$

Στην παραπάνω εξίσωση φαίνεται η γκαουσιανή συνάρτηση που χρησιμοποιείται στα κέντρα του δικτύου. Στην εξίσωση αυτή η παράμετρος r αντιστοιχεί στην ευκλείδεια απόσταση με την είσοδο που επεξηγείται παρακάτω, ενώ η παράμετρος σ είναι το πλάτος της γκαουσιανής συνάρτησης.

Συνοψίζοντας για να βρούμε την έξοδο του κάθε κέντρου θα πρέπει να βρούμε την ευκλείδεια απόσταση του κέντρου από τη συγκεκριμένη είσοδο και να την περάσουμε

μέσα από τη γκαουσιανή συνάρτηση. Έτσι παίρνουμε την έξοδο του κάθε κέντρου στο κρυφό επίπεδο.

Υπολογισμός Ευκλείδειας απόστασης

Έστω ότι $x = [x_1, x_2, \dots, x_n]$ είναι το διάνυσμα εισόδου και $R = [R_1, R_2, \dots, R_N]$ το διάνυσμα με τα κέντρα τότε η ευκλείδεια απόσταση της εισόδου με το κάθε κέντρο υπολογίζεται με την εξίσωση 2.

$$\|x - R\| = D_{\text{eucl}} = \sqrt{\sum_{i=1}^N (x_i - R_i)^2} \quad (2)$$

Τέλος για τον υπολογισμό της εξόδου του κάθε κέντρου περνάμε την ευκλείδεια απόσταση μέσα από τη γκαουσιανή συνάρτηση.

$$O(R) = \phi(\|x - R\|) \quad (3)$$

Υπολογισμός τελικής εξόδου του δικτύου

Όπως αναφέραμε οι εξοδοί των κέντρων πολλαπλασιάζονται με τους αντίστοιχους συντελεστές (c) και ακολούθως αθροίζονται για τον υπολογισμό του τελικού αποτελέσματος του δικτύου.

$$y = \sum_{i=1}^m c_i O_i \quad (4)$$

Στην εξίσωση 4 παρουσιάζεται η μέθοδος υπολογισμού της εξόδου του δικτύου. Στην εξίσωση αυτή το m αντιπροσωπεύει τον αριθμό των κόμβων στο κρυφό επίπεδο δηλαδή των κέντρων, το O_i αντιπροσωπεύει την έξοδο του i -οστού κέντρου και το c_i είναι το βάρος μεταξύ του ισοστού κέντρου και της εξόδου.

2.1.1.2 Αλγόριθμοι εκπαίδευσης δικτύων RBF

Όπως αναφέρθηκε στην αρχή της ενότητας αυτής μπορούμε να πετύχουμε παρεμβολή ή κατά προσέγγιση εκτίμηση του αποτελέσματος. Συγκεκριμένα υπάρχουν δύο κύριοι αλγόριθμοι για την εκπαίδευση του δικτύου.

- Αλγόριθμος μάθησης με σταθερά κέντρα – Μ' αυτόν τον αλγόριθμο επιτυγχάνουμε παρεμβολή.
- Αλγόριθμος μάθησης με μεταβλητά κέντρα. Επιτυγχάνουμε κατά προσέγγιση εκτίμηση του αποτελέσματος.

2.1.1.2.1 Αλγόριθμος με σταθερά κέντρα – Βήματα αλγόριθμου

1. Επιλογή κατάλληλων κέντρων
2. Επιλογή κατάλληλων σ
3. Αρχικοποίηση βαρών c με μικρές τυχαίες τιμές
4. Υπολογισμό της εξόδου με βάση την εξίσωση 3 και 4
5. Βρίσκουμε τα βέλτιστα βάρη με τη χρήση της φόρμουλας $c = y \times A^{-1}$ (A^{-1} = αντίστροφος του πίνακα με τις εξόδους των κέντρων)

2.1.1.2.2 Αλγόριθμος με μεταβλητά κέντρα – Βήματα αλγόριθμου

Ο αλγόριθμος αυτός στην ουσία είναι ένας στοχαστικός αλγόριθμος ταχύτερας καθόδου. Στον αλγόριθμο αυτό μπορούμε να μεταβάλουμε και τα τρία σύνολα παραμέτρων : τη θέση των κέντρων (R), τα βάρη (c) και τα πλάτη των γκαουσιανών συναρτήσεων (σ).

Στην εξίσωση 5 παρουσιάζεται η μέθοδος υπολογισμού του συνολικού στιγμιαίου σφάλματος (J).

$$J[K] = \frac{1}{2}[e[k]]^2 = \frac{1}{2} [\text{desire} - \text{actual}]^2 = \frac{1}{2} \{ d[k] - \sum_{k=1}^n c[k] \varphi(\chi[k], R_k) \}^2 \quad (5)$$

Οι εξισώσεις καθόδου για το κάθε ένα από τις τρεις παραμέτρους του δικτύου δίνονται από τις εξισώσεις 6,7 και 8 αντίστοιχα (Όπου ∂ είναι η μερική παράγωγος) .

$$c[k+1] = c[k] - n_c [\partial J[k] / \partial c_k] \quad (6)$$

$$R_k[k+1] = R_k[k] - n_r [\partial J[k] / \partial R_k] \quad (7)$$

$$\sigma_k[k+1] = \sigma_k[k] - n_\sigma [\partial J[k] / \partial \sigma_k] \quad (8)$$

Χρησιμοποιώντας τα πιο πάνω καταλήγουμε στα παρακάτω βήματα του αλγορίθμου.

1. Επιλογή κατάλληλων κέντρων
2. Επιλογή κατάλληλων σ
3. Αρχικοποίηση βαρών c με μικρές τυχαίες τιμές
4. Δίνουμε στο δίκτυο ένα διανυσματικό δείγμα και βρίσκουμε την έξοδο με τις εξισώσεις 3 και 4.
5. Βρίσκουμε τις νέες τιμές των c, R και σ με βάση τις εξισώσεις 9,10 και 11. (όπου $e[k] = \text{desire}[k] - \text{actual}[k]$, n αντιστοιχεί στο ρυθμό μάθησης, το διάνυσμα χ αντιστοιχεί στο διάνυσμα εισόδου και η μεταβλητή k συμβολίζει την τρέχουσα χρονική στιγμή)

$$R_k[k+1] = R_k[k] + n_r e[k] c_k[k] \Phi \{ \chi[k], R_k[k], \sigma_k[k] \} [\chi[k] - R_k[k]] [\sigma_k[k]]^{-2}$$

$$R_{ji} = \sum_n \sum_k (y_k(x^n) - t_k^n) \frac{\overset{\text{ή}}{w_{kj} \exp\left(-\frac{\text{dist}^2}{2\sigma_j^2}\right) (x_i^n - R_{ji})}}{\sigma_j^2}$$

Όπου n είναι η διάσταση του διανύσματος εισόδου, t είναι το επιθυμητό αποτέλεσμα για την έξοδο k και το νοστό διάνυσμα εισόδου, k είναι το πλήθος των εξόδων, w είναι το βάρος από το κέντρο j στην έξοδο k , σ είναι το πλάτος της γκαουσιανής συνάρτησης, dist είναι η Ευκλείδειος απόσταση του κέντρου από την είσοδο και R_{ji} αντιστοιχεί στο ιστό στοιχείο του κέντρου R_j .

(9)

$$\sigma_k[k+1] = \sigma_k[k] + n_\sigma e[k] c_k[k] \Phi \{ \chi[k], R_k[k], \sigma_k[k] \} \|\chi[k] - R_k[k]\|^2 [\sigma_k[k]]^{-3}$$

ή

$$\sigma_j = \sum_n \sum_k (y_k(x^n) - t_k^n) \frac{w_{kj} \exp\left(-\frac{dist_j^2}{2\sigma_j^2}\right) dist_j^2}{\sigma_j^3}$$

Όπου n είναι η διάσταση του διανύσματος εισόδου, t είναι το επιθυμητό αποτέλεσμα για την έξοδο k και το νοστό διάνυσμα εισόδου, k είναι το πλήθος των εξόδων, w είναι το βάρος (c) από το κέντρο j στην έξοδο k , σ_j είναι το πλάτος της γκαουσιανής συνάρτησης του κέντρου j και $dist$.

(10)

$$c[k+1] = c[k] + n_c e[k] \psi[k]$$

(11)

$$\text{όπου } \psi[k] = [\Phi\{\chi[k], R_1[k], \sigma_1[k]\}, \dots, \Phi\{\chi[k], R_n[k], \sigma_n[k]\}]^T$$

$$c_{jk} = \sum_n \sum_k (y_k(x^n) - t_k^n) \exp\left(-\frac{dist_j^2}{2\sigma_j^2}\right)$$

Όπου n είναι η διάσταση του διανύσματος εισόδου, t είναι το επιθυμητό αποτέλεσμα για την έξοδο k και το νοστό διάνυσμα εισόδου, k είναι το πλήθος των εξόδων, σ_j είναι το πλάτος της γκαουσιανής συνάρτησης του κέντρου j και $dist_j$ είναι η Ευκλείδειος απόστασης του κέντρου j από το διάνυσμα εισόδου.

6. Εάν το δίκτυο μας έδειξε ικανοποιητική σύγκλιση τότε σταματάμε, διαφορετικά πάμε στο βήμα 4.

2.1.1.3 Μέθοδοι επιλογής των κέντρων

Τα κέντρα επιλέγονται έτσι ώστε να αντιπροσωπεύουν την κατανομή των δεδομένων εισόδου στο χώρο. Επομένως η επιλογή των κέντρων παίζει σημαντικό ρόλο στην απόδοση του δικτύου μας. Σημαντικό όμως ρόλο παίζει και το πλήθος τους. Αν έχουμε περισσότερα ή λιγότερα κέντρα απ' ότι πρέπει τότε το δίκτυο μας παρουσιάζει ανακρίβειες. Δεν υπάρχουν τεχνικές για τον υπολογισμό του βέλτιστου αριθμού των κέντρων αλλά μια γενική γραμμή είναι να επιλέγουμε τόσα κέντρα όσα τα δεδομένα εισόδου όταν θέλουμε να πετύχουμε παρεμβολή, ενώ να επιλέγουμε λιγότερα (συνήθως τα μισά) όταν θέλουμε κατά προσέγγιση εκτίμηση. Για την επιλογή των θέσεων των κέντρων μπορούμε είτε να χρησιμοποιήσουμε μια κανονική κατανομή μέσα στο χώρο

τον δεδομένων είτε να χρησιμοποιήσουμε κάποιο αλγόριθμο ομαδοποίησης, π.χ. Kohonen SOM [11] ο οποίος θα μας δώσει το πλήθος καθώς επίσης και τις αρχικές θέσεις των κέντρων.

Μέθοδοι επιλογής διασπορών

Στά Radial Basis Functions ή επιλογή του πλάτους της γκαουσιανής συνάρτησης παίζει σημαντικό ρόλο στην εκπαίδευση.

Συγκεκριμένα, σύμφωνα με εμπειρικούς κανόνες όταν θέλουμε να πετύχουμε παρεμβολή και επιλέξουμε μικρό σ τότε έχουμε απότομες διακυμάνσεις, όταν επιλέξουμε μεγάλο σ τότε η επίδραση του κάθε δείγματος αυξάνεται, ενώ όταν επιλέξουμε πολύ μεγάλο σ τότε η επίδραση του κάθε δείγματος γίνεται γενικότερη.

Όταν θέλουμε να επιτύχουμε κατά προσέγγιση εκτίμηση τότε η τιμή του πλάτους της γκαουσιανής συνάρτησης (σ) θα πρέπει να είναι μεγαλύτερη από τη μέση απόσταση μεταξύ δύο γειτονικών διανυσμάτων, αλλά να είναι μικρότερη από τη συνολική απόσταση του χώρου των εισόδων.

2.1.1.4 Σύγκριση RBF με MLP

Ομοιότητες :

1. Έχουμε μάθηση από κάποιο σύνολο δεδομένων εκπαίδευσης
2. Έχουν τη δυνατότητα γενίκευσης
3. Μπορούν να υλοποιηθούν με τη χρήση παράλληλης επεξεργασίας

Διαφορές :

1. Στα MLP έχουμε βάρη και κατώφλια σ' αντίθεση με τα RBF που έχουμε κέντρα και coefficients.
2. Τα MLP είναι αργά και έχουν αυξημένη υπολογιστική πολυπλοκότητα σ' αντίθεση με τα RBF που είναι πιο απλά και πιο γρήγορα.

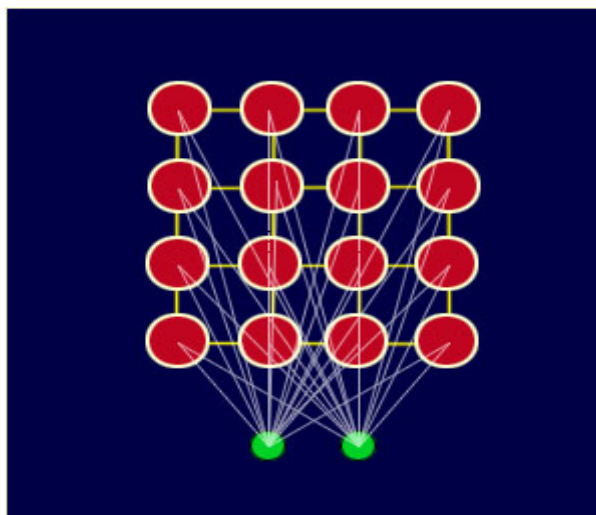
3. Τα MLP είναι πολύ καλά όμως έχουν δύσκολο μαθηματικό υπόβαθρο, ενώ τα RBF έχουν εύκολο μαθηματικό υπόβαθρο όμως υπερτερούν στην απόδοση.

2.1.2 Χάρτες Αυτό-οργάνωσης Kohonen

Αυτός ο αλγόριθμος εμπίπτει στην κατηγορία της μη επιβλεπόμενης μάθησης. Πιο συγκεκριμένα δεν είναι γνωστό το επιθυμητό αποτέλεσμα και επομένως δεν υπάρχει κάποιος ‘‘δάσκαλος’’ ή ‘‘κριτής’’ που να δείχνει στο δίκτυο αν το αποτέλεσμα εξόδου είναι ορθό ή λανθασμένο. Οι αλγόριθμοι αυτής της κατηγορίας στηρίζονται στην ομαδοποίηση των δεδομένων, με την παραδοχή ότι δεδομένα που ανήκουν στην ίδια ομάδα έχουν κάποια κοινά χαρακτηριστικά. Επίσης ονομάζονται και αλγόριθμοι συμπίεσης αφού στην ουσία συμπιέζουν τα δεδομένα. Για παράδειγμα από ένα σύνολο με 10000 δεδομένα μέσω αυτών των αλγορίθμων μπορούμε να πάρουμε 10 κατηγορίες. Έτσι οι διαστάσεις των δεδομένων μειώνονται – και τα δεδομένα συμπιέζονται – από 10000 σε μόνο 10.

2.1.2.1 Αρχιτεκτονική του Δικτύου

Το δίκτυο αποτελείται από ένα ορθογώνιο πλέγμα διαστάσεων $M \times N$ καθώς επίσης και από το διάνυσμα εισόδου. Το κάθε στοιχείο του διανύσματος εισόδου συνδέεται με όλους τους κόμβους του πλέγματος. Στην εικόνα 2.1.2.1.1 που ακολουθεί βλέπουμε ένα απλό δίκτυο Kohonen [10][11] αποτελούμενο από ένα πλέγμα 4×4 και ένα δυσδιάστατο διάνυσμα εισόδου. Ο κάθε κόμβος του πλέγματος έχει ένα διάνυσμα με βάρη το οποίο έχει την ίδια διάσταση με το διάνυσμα εισόδου.



Εικόνα 2.1.2.1.1 Αρχιτεκτονική δικτύου Kohonen [10]

2.1.2.2 Λεπτομερής περιγραφή του αλγορίθμου

Όπως αναφέραμε και πιο πάνω ο αλγόριθμος αυτός δεν απαιτεί την ύπαρξη του επιθυμητού αποτελέσματος. Το πραγματικό αποτέλεσμα προκύπτει από τον κόμβο του πλέγματος του οποίου τα βάρη βρίσκονται πιο κοντά στο διάνυσμα εισόδου. Στη συνέχεια θα αναλύσουμε τα βήματα του αλγορίθμου.

Βήμα 1. Αρχικοποίηση των βαρών

Στο βήμα αυτό γίνεται αρχικοποίηση των βαρών για κάθε κόμβο στο πλέγμα με μικρές τυχαίες τιμές μεταξύ 0 και 1.

Βήμα 2. Υπολογισμός του νευρώνα νικητή (BMU – Best Matching Unit)

Ο νευρώνας νικητής είναι αυτός με τη μικρότερη ευκλείδεια απόσταση από το διάνυσμα εισόδου. Επομένως πρέπει για κάθε κόμβο στο πλέγμα να βρούμε την ευκλείδεια απόσταση (12) και στη συνέχεια να βρούμε το νευρώνα που έχει τη μικρότερη απόσταση.

$$D_{\text{eucl}} = \sqrt{\sum_{i=1}^N (x_i - R_i)^2} \quad (12)$$

Στην παραπάνω εξίσωση η τιμή N αντιστοιχεί στη διάσταση της εισόδου, το διάνυσμα x αντιπροσωπεύει το διάνυσμα εισόδου και το διάνυσμα R αντιπροσωπεύει το διάνυσμα με τα βάρη για κάθε νευρώνα στο πλέγμα.

Βήμα 3. Υπολογισμός της ακτίνας της γειτονίας του BMU

Αφού βρούμε το νευρώνα νικητή πρέπει να βρούμε την ακτίνα της γειτονίας του. Βασική αρχή σ' αυτό το βήμα είναι ότι η ακτίνα αυτή με την πάροδο του χρόνου μειώνεται με αποτέλεσμα να έχουμε πιο συγκεκριμένες κατηγορίες. Επομένως για τον υπολογισμό της ακτίνας χρησιμοποιούμε την εξίσωση 13 (t είναι η τρέχουσα χρονική στιγμή, λ είναι μια χρονική σταθερά και σ_0 είναι η αρχική ακτίνα):

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\lambda}\right) \quad t = 1, 2, 3 \dots \quad (13)$$

Βήμα 4. Αλλαγή των βαρών

Αφού ολοκληρωθούν τα βήματα 1-3 πρέπει να αλλάξουμε τα βάρη του νευρώνα νικητή καθώς και τα βάρη των νευρώνων που βρίσκονται μέσα στην ακτίνα της γειτονίας του. Αυτό γίνεται εφικτό με τη χρήση της παρακάτω εξίσωσης.

$$W(t+1) = W(t) + \Theta(t)L(t)(V(t) - W(t)) \quad (14)$$

Στην παραπάνω εξίσωση παρουσιάζονται δύο νέοι παράγοντες. Το $L(t)$ αντιπροσωπεύει το ρυθμό μάθησης και το $\Theta(t)$ δείχνει την επίδραση που έχει στη μάθηση η απόσταση ενός νευρώνα από το νικητή. Μ' αυτό τον τρόπο επιτυγχάνεται μεγαλύτερη αλλαγή στα βάρη των νευρώνων που βρίσκονται πιο κοντά στο νικητή σ' αντίθεση με τους πιο απομακρυσμένους νευρώνες. Στις δύο παρακάτω εξισώσεις παρουσιάζονται οι εξισώσεις υπολογισμού για το ρυθμό μάθησης L (15) - t είναι η τρέχουσα χρονική στιγμή, λ είναι μια χρονική σταθερά και L_0 είναι ο αρχικός ρυθμός μάθησης - και της

επίδρασης του νευρώνα νικητή (BMU) Θ (16) - t είναι η τρέχουσα χρονική στιγμή, σ είναι η ακτίνα της γειτονιάς και $dist$ είναι η ευκλείδειος απόσταση του νευρώνα από την είσοδο -αντίστοιχα.

$$L(t) = L_0 \exp\left(-\frac{t}{\lambda}\right) \quad t = 1, 2, 3, \dots \quad (15)$$

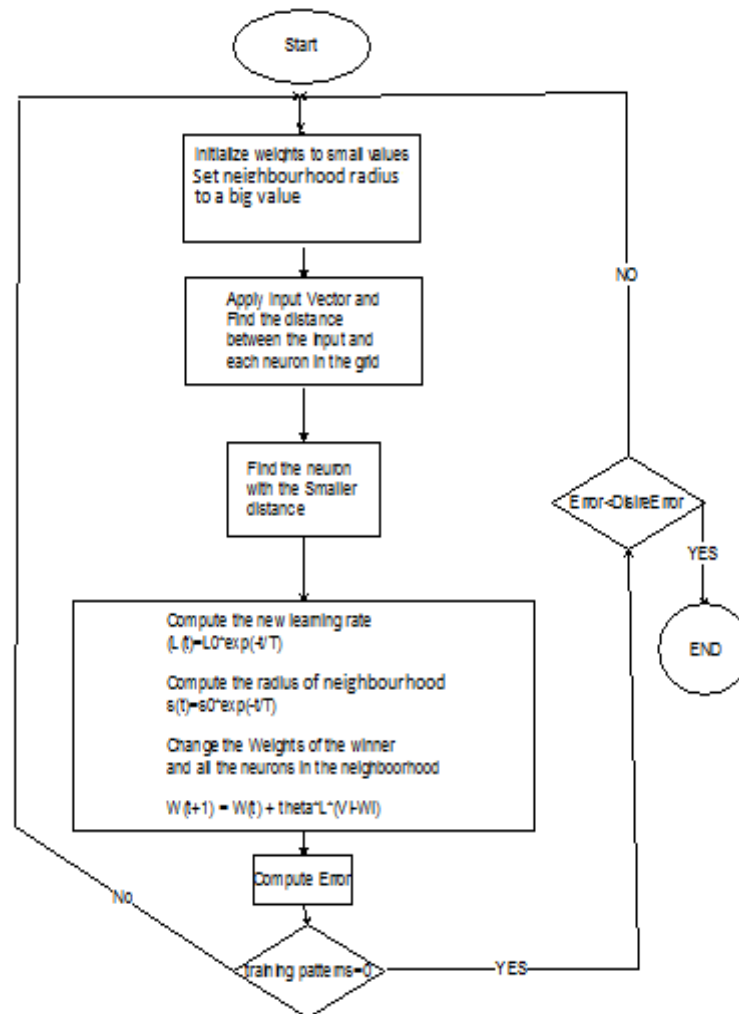
$$\Theta(t) = \exp\left(-\frac{dist^2}{2\sigma^2(t)}\right) \quad t = 1, 2, 3, \dots \quad (16)$$

Συνοψίζοντας παραθέτουμε τον αλγόριθμο SOM καθώς και το διάγραμμα ροής του (εικόνα 2.1.2.2.1).

1. Αρχικοποίησε τα βάρη των νευρώνων του δισδιάστατου πλέγματος με τυχαίες μικρές τιμές. Αρχικοποίησε το μέγεθος της γειτονιάς (r_0) και το ρυθμό μάθησης (L_0).
2. Επέλεξε τυχαία ένα διάνυσμα εισόδου από το σύνολο διανυσμάτων που έχουμε ως δεδομένα και θέλουμε να κατηγοριοποιήσουμε.
3. Σύγκρινε τα βάρη κάθε νευρώνα με το αντίστοιχο στοιχείο του διανύσματος εισόδου και υπολόγισε την ομοιότητα μεταξύ των βαρών και του διανύσματος εισόδου (ο έλεγχος γίνεται με βάση μια συνάρτηση και συνήθως γίνεται με την Ευκλείδεια απόσταση).
4. Επέλεξε τον νευρώνα του οποίου τα βάρη «μοιάζουν» περισσότερο από κάθε άλλου κόμβου με το διάνυσμα εισόδου και όρισέ τον ως τον BMU.
5. Προσάρμοσε τα βάρη του BMU και των νευρώνων που βρίσκονται στη γειτονιά του ώστε να μοιάσουν περισσότερο στο διάνυσμα εισόδου. Όσο πιο κοντά είναι κάποιος νευρώνας στον BMU, τόσο μεγαλύτερη αλλαγή υφίστανται τα βάρη του, ενώ ο BMU έχει τη μεγαλύτερη αλλαγή στα βάρη του.
6. Μείωσε το ρυθμό μάθησης $L(i)$ και το μέγεθος της γειτονιάς $r(i)$

7. Επανέλαβε από το βήμα 2

8. Τρέξε μια φορά τον αλγόριθμο περνώντας όλα τα δεδομένα με τη σειρά χωρίς να κάνεις αλλαγές στα βάρη και έχοντας ως μέγεθος της ακτίνας το 1. Αύξησε το μετρητή κάθε νευρώνα που επιλέγεται ως BMU και σημείωσε πιο δεδομένο τον ενεργοποίησε.



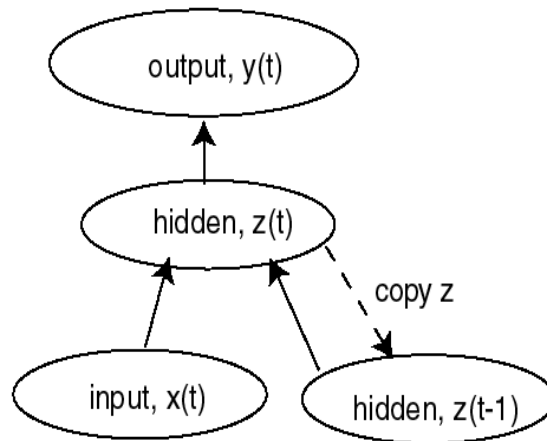
Εικόνα 2.1.2.2.1. Διάγραμμα ροής αλγορίθμου Kohonen SOM. [10][11]

2.1.3 Δίκτυα Ανατροφοδότησης

Τα δίκτυα αυτής της κατηγορίας είναι ίδια με τα δίκτυα πολλαπλών στρωμάτων perceptron με μόνη διαφορά ότι η επόμενη κατάσταση εξαρτάται από την

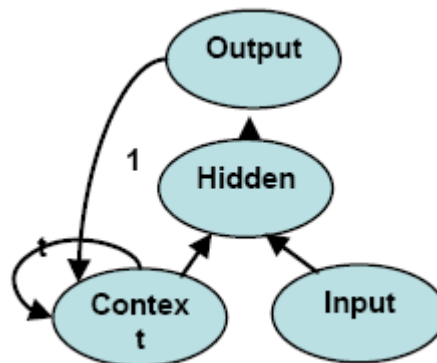
προηγούμενη. Στην πραγματικότητα σ' αυτήν την κατηγορία το δίκτυο έχει βραχυπρόθεσμη μνήμη. Συγκεκριμένα υπάρχουν state units τα οποία ανατροφοδοτούν το δίκτυο με τα προηγούμενα αποτελέσματα. Υπάρχουν δύο κύριες κατηγορίες τέτοιων δικτύων:

- Δίκτυο Elman [12] στο οποίο η ανατροφοδότηση γίνεται από το κρυφό επίπεδο



Εικόνα 2.1.3.1. Δίκτυο Elman. [12]

- Δίκτυο Jordan [13] όπου η ανατροφοδότηση γίνεται από το επίπεδο εξόδου



Εικόνα 2.1.3.2. Δίκτυο Jordan [13]

2.1.3.1 Αρχιτεκτονική

Για το σκοπό αυτής της διπλωματικής εργασίας όπως αναφέραμε και στην περιγραφή θα υλοποιήσουμε δίκτυο ανατροφοδότησης τύπου Jordan [13] το οποίο θα

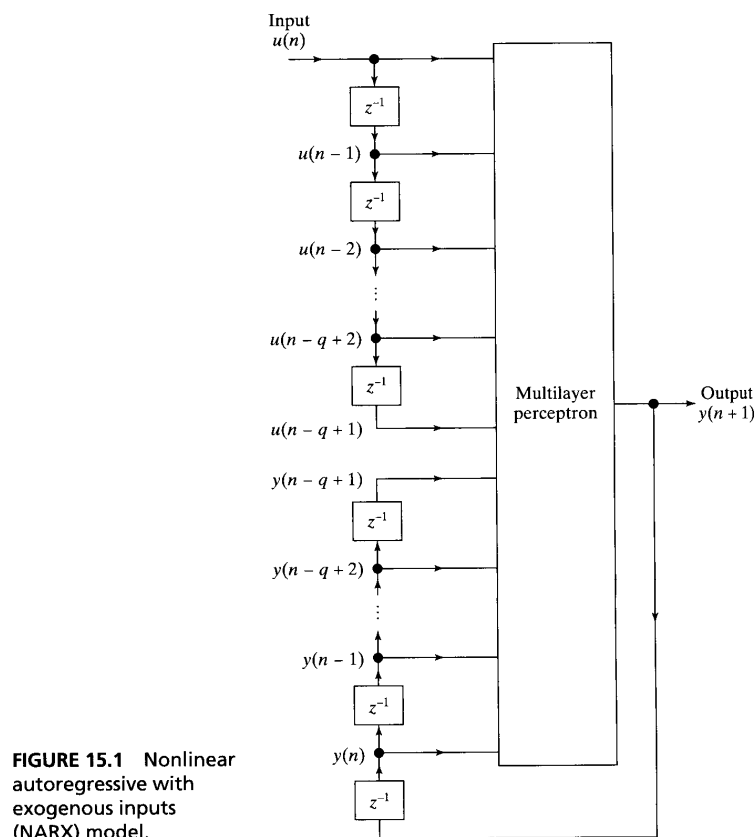
εκπαιδεύσουμε με την χρήση του αλγορίθμου ανάστροφης μετάδοσης του λάθους στο χρόνο (Back Propagation through time) [14]. Προτού αναλύσουμε την δομή του δικτύου παραθέτουμε κάποια γενικά στοιχεία για τα δίκτυα ανατροφοδότησης καθώς επίσης και για τους αλγορίθμους εκπαίδευσης τους,

Τα δίκτυα ανατροφοδότησης συναντούνται σε τέσσερις κύριες αρχιτεκτονικές:

- Μη γραμμικό με αυτοπαλιδρόμηση και εξωγενείς εισόδους (Nonlinear autoregressive with exogenous inputs - NARX) [7]
- Απλό δίκτυο ανατροφοδότησης (simple recurrent network - SRN) [7][12]
- State-Space Model [7]
- Πολυεπίπεδο Perceptron με ανάδραση (Recurrent multilayer perceptron - RMLP) [7]

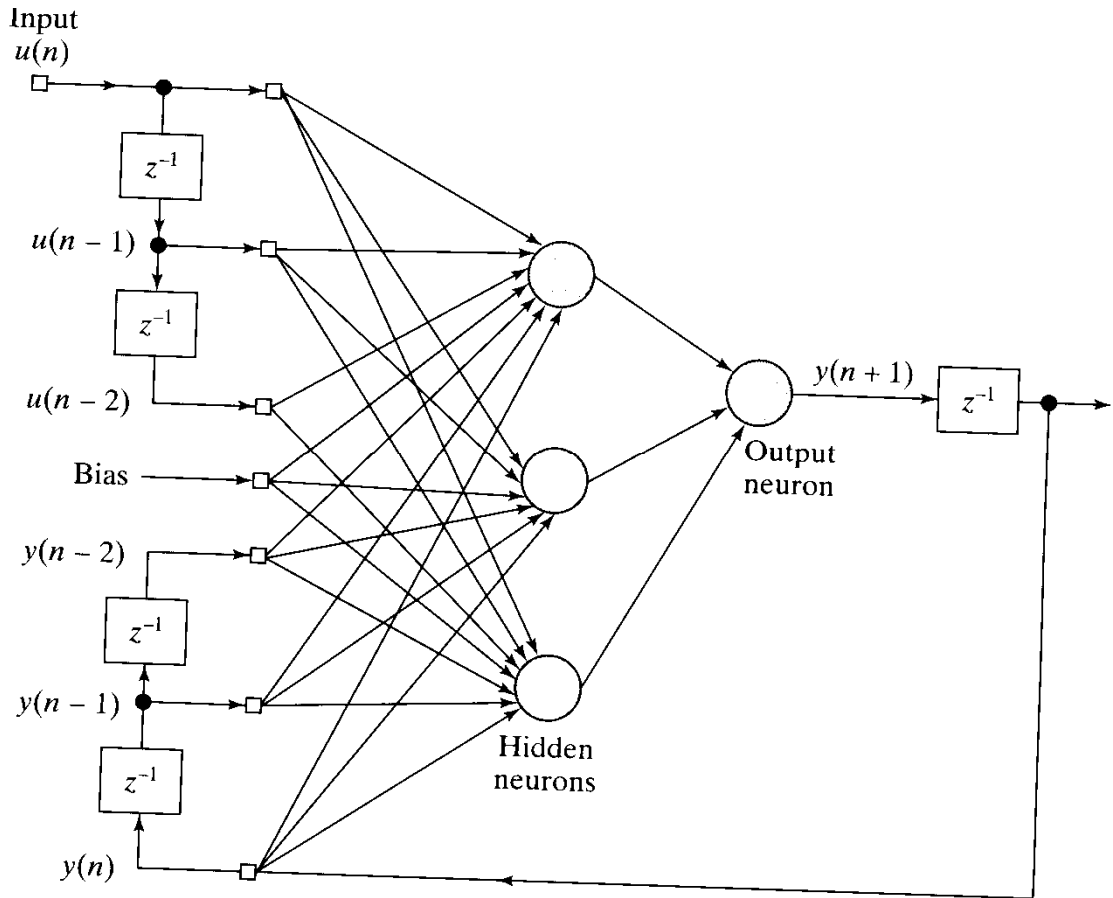
Nonlinear autoregressive with exogeneous inputs model (NARX)

Η γενικευμένη αρχιτεκτονική αυτού του μοντέλου παρουσιάζεται στην εικόνα που ακολουθεί.



Εικόνα 2.1.3.1.1. Δίκτυο NARX [7]

Στο παραπάνω δίκτυο η έξοδος την χρονική στιγμή $n+1$ εξαρτάται από κάποιο αριθμό q



προηγούμενων εξόδων καθώς επίσης και από κάποιο αριθμό q προηγούμενων εισόδων. Αυτό γίνεται πιο κατανοητό με το παρακάτω παράδειγμα όπου η τιμή του q είναι 3.

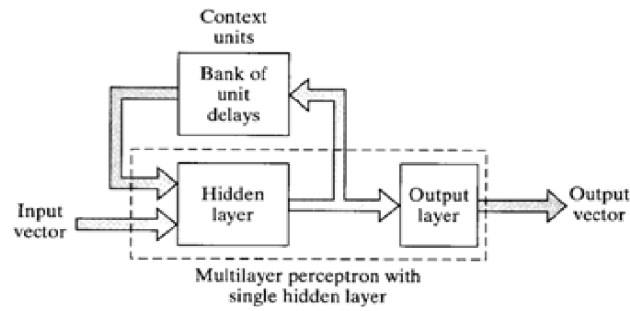
Εικόνα 2.1.3.1.2. Δίκτυο NARX με 3 states units [7]

Στο παραπάνω παράδειγμα η τιμή της εξόδου y τη χρονική στιγμή $n+1$ εξαρτάται από τις προηγούμενες 3 εισόδους u και 3 εξόδους y , επομένως μπορούμε να συνοψίσουμε τα παραπάνω με την πιο κάτω έκφραση:

$$y(n+1) = F(y(n), \dots, y(n-q+1), u(n), \dots, u(n-q+1)) \quad (17)$$

Simple Recurrent Network (SRN)

Στην εικόνα 2.1.3.1.3 που ακολουθεί βλέπουμε την αρχιτεκτονική αυτής της κατηγορίας.



Εικόνα 2.1.3.1.3. Αρχιτεκτονική SRN [7]

Σ' αυτήν την αρχιτεκτονική γίνεται αντικατάσταση του επιπέδου εξόδου με ένα επίπεδο το οποίο δεν είναι γραμμικό καθώς επίσης δεν έχουμε τα states units στην έξοδο.

State Space Model

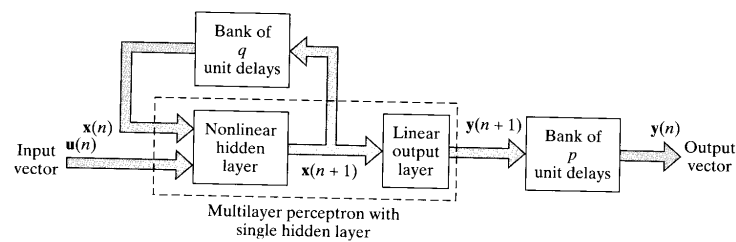


FIGURE 15.2 State-space model.

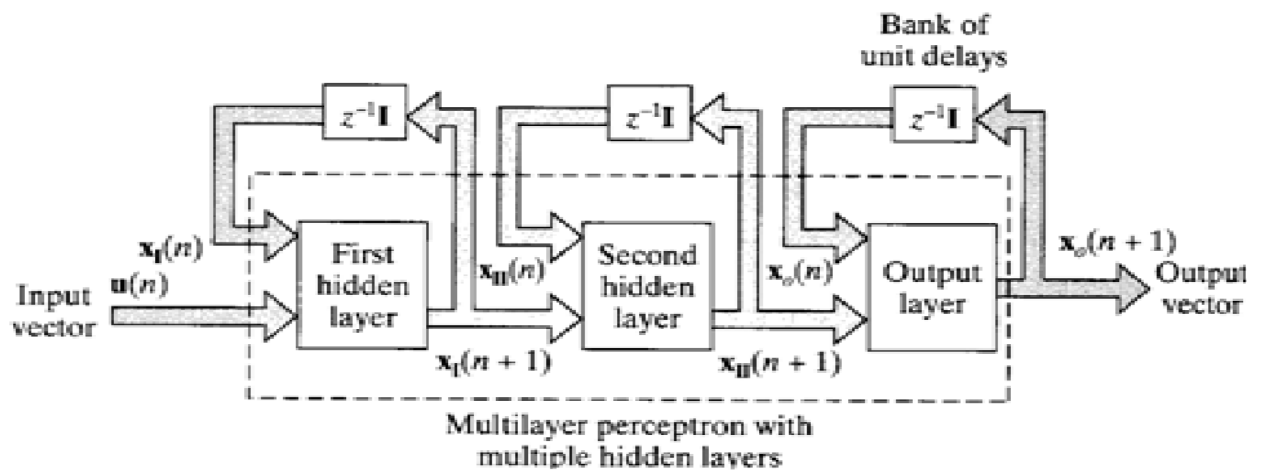
Εικόνα 2.1.3.1.4. Αρχιτεκτονική State Space [7]

Παρατηρούμε ότι στο μοντέλο αυτό σ' αντίθεση με το προηγούμενο έχουμε γραμμικό επίπεδο εξόδου καθώς επίσης υπάρχουν και states units στην έξοδο. Πιο συγκεκριμένα οι κρυφοί νευρώνες καθορίζουν την κατάσταση του δικτύου. Το επίπεδο εισόδου αποτελείται από του κόμβους εισόδου καθώς επίσης και από τους κόμβους ανατροφοδότησης οι οποίοι παίρνουν τα αποτελέσματα των νευρώνων του κρυφού επιπέδου. Ανάλογα με τον αριθμό των κόμβων ανατροφοδότησης από το κρυφό επίπεδο στο επίπεδο εισόδου εξαρτάται και η κατάταξη (order) του μοντέλου.

Recurrent multilayer perceptron (RMLP)

Αυτό το μοντέλο μπορεί να αποτελείται από ένα ή περισσότερα hidden layers και η ανατροφοδότηση γίνεται πίσω στο ίδιο επίπεδο. Για παράδειγμα θα έχουμε δυο κρυφά επίπεδα, η έξοδος του πρώτου επιπέδου θα ανατροφοδοτηθεί ως είσοδος στο πρώτο

επίπεδο ενώ η έξοδος του δευτέρου επιπέδου θα ανατροφοδοτηθεί ως είσοδος στο δεύτερο επίπεδο. Αυτά που περιγράφονται πιο πάνω φαίνονται στην εικόνα που ακολουθεί.



Εικόνα 2.1.3.1.5. Αρχιτεκτονική Recurrent multilayer perceptron

2.1.3.2 Εκπαίδευση δικτύων με ανάδραση

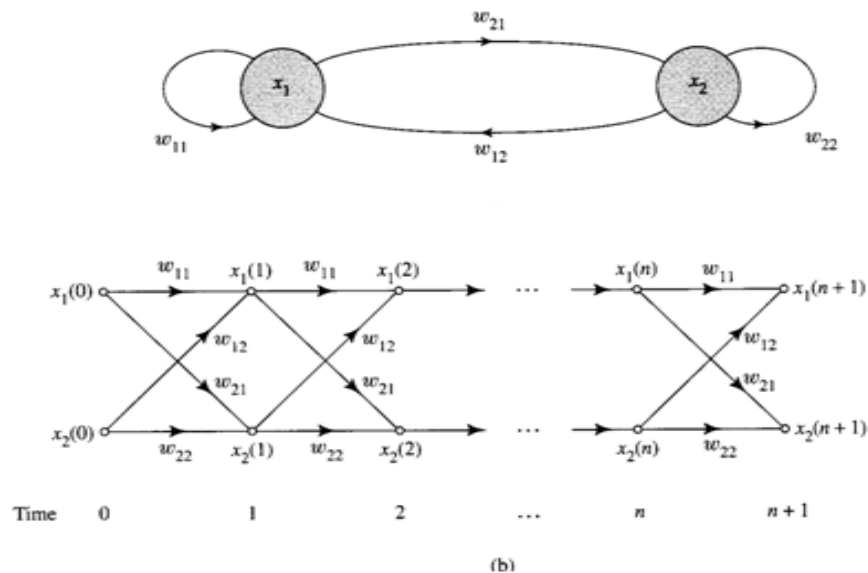
Υπάρχουν δυο βασικές μέθοδοι για εκπαίδευση των δικτύων με ανατροφοδότηση. Στην εποχιακή εκπαίδευση (Epochwise training) έχουμε μια καθορισμένη εποχή και το δίκτυο αρχίζει να τρέχει από μια αρχική κατάσταση μέχρι να φτάσει σε μια άλλη κατάσταση όπου η εκπαίδευση σταματά και το δίκτυο επαναφέρεται στην αρχική κατάσταση για την επόμενη εποχή. Στη συνεχή εκπαίδευση Continuous training το δίκτυο μαθαίνει με τη μεταφορά σημάτων μέσα στο δίκτυο. Η μέθοδος αυτή χρησιμοποιείται κυρίως στις περιπτώσεις που δεν έχουμε καταστάσεις επαναφοράς (reset states).

Στα δίκτυα μ' ανατροφοδότηση υπάρχουν δύο βασικοί αλγόριθμοι εκπαίδευσης:

- Ανάστροφη μετάδοση του λάθους στο χρόνο (back-propagation-through-time) [7][14]
- Εκπαίδευση με ανατροφοδότηση σε πραγματικό χρόνο (real-time recurrent learning) [7]

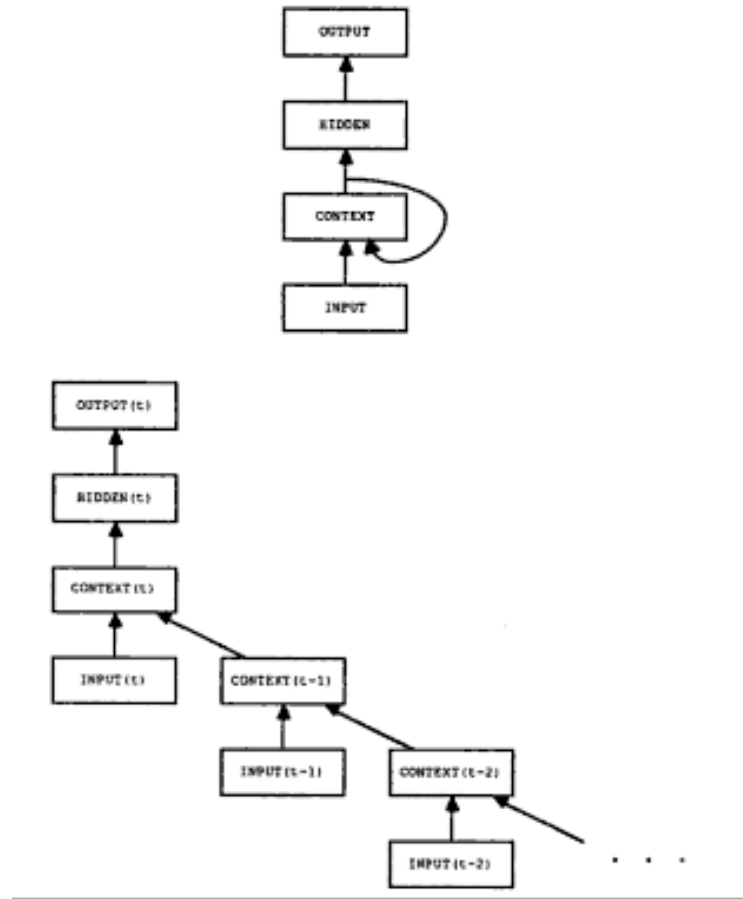
Back propagation through time

Ο αλγόριθμος αυτός στην πραγματικότητα είναι μια επέκταση του αλγορίθμου ανάστροφης μετάδοσης του σφάλματος (Back propagation). Για να μπορεί να εφαρμοστεί, πρώτα το δίκτυο πρέπει να αναδιπλωθεί σ' ένα layered feed forward network το οποίο μεγαλώνει κατά ένα επίπεδο σε κάθε χρονικό βήμα. Η πιο πάνω διαδικασία φαίνεται στην εικόνα που ακολουθεί. Όπως βλέπουμε σε κάθε επίπεδο έχουμε τον ίδιο αριθμό νευρώνων και τα βάρη μεταξύ τους παραμένουν αναλλοίωτα (π.χ το βάρος $w_{1,1}$ παραμένει αναλλοίωτο από την αρχή μέχρι το τέλος).



Εικόνα 2.1.3.2.1. Παράδειγμα αναδίπλωσης του δικτύου [7]

Στην εικόνα που ακολουθεί είναι πιο εμφανές πως γίνεται η αναδίπλωση του δικτύου στο χρόνο. Παρατηρούμε ότι σε κάθε χρονικό βήμα με εξαίρεση το τελευταίο υπάρχει μια «αντιγραφή» της κατάστασης του δικτύου και συνεπώς το δίκτυο αναδιπλώνεται.



Εικόνα 2.1.3.2.2. Παράδειγμα αναδίπλωσης του δικτύου. [15]

Epochwise Back propagation through time

Σ' αυτή την εκδοχή του αλγορίθμου υποθέτουμε ότι τα δεδομένα εκπαίδευσης έχουν χωριστεί σε n_1 patterns κάθε ένα από τα οποία αποτελεί συγκεκριμένη περιοχή ενδιαφέροντος. Επομένως για τον υπολογισμό του σφάλματος θα χρησιμοποιήσουμε την παρακάτω εξίσωση.

$$E = \sum_{n=0}^{n_1} \sum_{j \in A} e_{j,n}^2 \quad (18)$$

Όπου A είναι οι νευρώνες όπου είναι γνωστό το επιθυμητό αποτέλεσμα.

Ακολουθούν αναλυτικά τα βήματα του αλγορίθμου:

Βήμα 1

Στο βήμα αυτό γίνεται ένα απλό εμπρόσθιο πέρασμα (feed forward pass) από το δίκτυο στο διάστημα n_0 με n_1 και καταγράφονται και αποθηκεύονται όλα τα δεδομένα εισόδου, οι καταστάσεις του δικτύου καθώς επίσης και οι αναμενόμενες ανταποκρίσεις.

Βήμα 2

Στη συνέχεια εκτελείται ανάστροφη μετάδοση (Backward pass) για τον υπολογισμό των τοπικών delta ($\delta_{j,n}$). Ο υπολογισμός αυτός γίνεται με την εφαρμογή της παρακάτω εξίσωσης.

$$\delta_{j,n} = \phi'(j,n) e_{j,n} \quad \text{Av } n = n_1 \quad (19)$$

$$\delta_{j,n} = \phi'(j,n) [e_{j,n} + \sum_{k \in A} w_{jk} \delta_{k,n+1}] \quad \text{Av } n_0 < n < n_1 \quad (20)$$

Όπου $\phi'(j,n)$ είναι η παράγωγος της συνάρτησης ενεργοποίησης, $e_{j,n}$ είναι το σφάλμα στο νευρώνα j τη χρονική στιγμή n και $\delta_{j,n}$ αντιπροσωπεύει το σφάλμα που πρέπει να μεταδοθεί προς τα πίσω και το w_{jk} αντιπροσωπεύει το βάρος από τον νευρώνα j προς τον νευρώνα k . Η εξίσωση 19 χρησιμοποιείται στο τελευταίο χρονικό βήμα, ενώ σε όλα τα υπόλοιπα βήματα χρησιμοποιείται η εξίσωση 20.

Βήμα 3

Αφού φτάσουμε στη χρονική στιγμή $n_0 + 1$ αλλάζουμε τα βάρη με τη χρήση του τύπου που ακολουθεί.

$$\Delta w_{j,i} = n \sum_{n=n_0+1}^{n_1} \delta_{j,n} x_{i,n-1} \quad (21)$$

Όπου n είναι ο ρυθμός μάθησης και $x_{i,n-1}$ είναι η είσοδος που εφαρμόζεται στην i -οστή σύναψη του νευρώνα j τη στιγμή $n-1$ και $\delta_{j,n}$ αντιπροσωπεύει το σφάλμα που πρέπει να μεταδοθεί προς τα πίσω.

Truncated Back propagation through time

Σ' αυτή τη μορφή του αλγορίθμου αποθηκεύουμε τα δεδομένα εισόδου καθώς και τις καταστάσεις του δικτύου για ένα καθορισμένο αριθμό χρονικών βημάτων . Αυτός ο αριθμός ονομάζεται truncation depth. Για τον υπολογισμό του σφάλματος σ' αυτήν την περίπτωση χρησιμοποιούμε το μέσο τετραγωνικό σφάλμα.

$$E(n) = \sum_{j \in A} e_{j(n)}^2 \quad (22)$$

Στη συνέχεια όπως και στην προηγούμενη μορφή χρησιμοποιούμε τη μέθοδο κατάβασης κλίσης για τον υπολογισμό του μεταφερόμενου σφάλματος (delta).

$$\delta_{j,l} = \phi'(j,l) e_{j,l} \quad \text{Av } l = n \quad (23)$$

$$\delta_{j,l} = \phi'(j,l) [e_{j,n} + \sum_{k \in A} w_{jk,l} \delta_{k,l+1}] \quad \text{Av } n-h < l < n \quad (24)$$

Όπου $\phi'(j,l)$ είναι η παράγωγος της συνάρτησης ενεργοποίησης και h είναι το βάθος στο χρόνο (truncation depth).

Τέλος, με τη εξίσωση 25, που ακολουθεί υπολογίζουμε και αλλάζουμε τις τιμές των βαρών.

$$\Delta w_{j,i} = n \sum_{n=n-h+1}^n \delta_{j,l} x_{i,l-1} \quad (25)$$

Όπου n είναι ο ρυθμός μάθησης και $x_{i,l-1}$ είναι η είσοδος που εφαρμόζεται στην i -οστή σύναψη του νευρώνα j την στιγμή $l-1$.

2.2 Επεξεργασία Δεδομένων

Όπως αναφέραμε και στην εισαγωγή, για την έρευνα αυτή χρησιμοποιήθηκαν πραγματικά δεδομένα από μεγάλη εταιρία φυσικού αερίου της Μεγάλης Βρετανίας και καλύπτουν ένα χρονικό διάστημα 7 ετών. Συγκεκριμένα ξεκινούν από τις 2 του Μάη του 1990, μέχρι τις 30 του Σεπτεμβρίου του 1997. Τα δεδομένα αυτά περιλαμβάνουν

πληροφορίες για τη ζήτηση φυσικού αερίου τη θερμοκρασία καθώς επίσης και την ταχύτητα των ανέμων. Στον πίνακα 2.2.1 παρουσιάζονται συνοπτικά τα τρία αυτά στοιχεία καθώς επίσης και σχετικές μ' αυτά πληροφορίες.

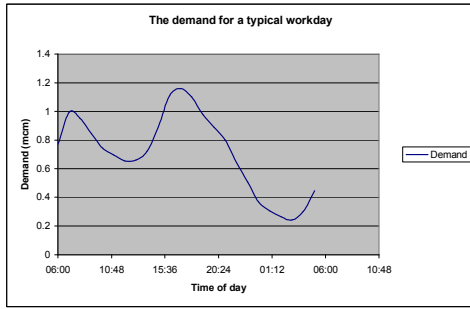
A/A	Παράγοντας	Μονάδα Μέτρησης	Συχνότητα Μέτρησης
1	Ζήτηση	εκατομμύρια κυβικά μέτρα	Καθημερινά ανά ώρα
2	Θερμοκρασία	Βαθμοί Κελσίου	Καθημερινά ανά δύο ώρες
3	Ταχύτητα των ανέμων	Κόμβους (knots)	Καθημερινά ανά τέσσερις ώρες

Πίνακας 2.2.1. Περιγραφή δεδομένων πριν την επεξεργασία

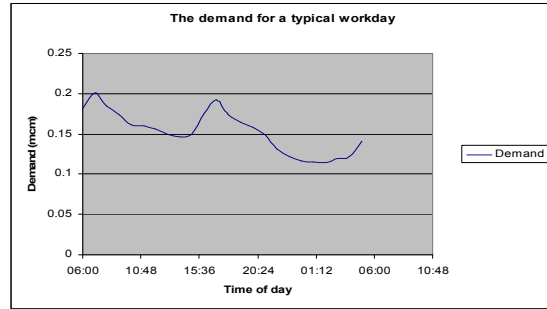
Από τη μελέτη των δεδομένων καταλήξαμε σε παρατηρήσεις οι οποίες επιβεβαιώνουν τη σημαντικότητα τόσο της θερμοκρασίας όσο και του ανέμου στον τομέα της ζήτησης του φυσικού αερίου. Οι παρατηρήσεις που καταλήξαμε είναι οι εξής:

- Στις καθημερινές μέρες παρουσιάζεται μεγαλύτερη ζήτηση τις πρωινές ώρες και απογευματινές ώρες σε σύγκριση με τις υπόλοιπες ώρες τις μέρας.
- Έχουμε μεγαλύτερη ζήτηση τους χειμερινούς μήνες σε σύγκριση με τους καλοκαιρινούς λόγω πτώσης της θερμοκρασίας.
- Τις αργίες και κυρίως τις κρύες μέρες η ζήτηση είναι μεγαλύτερη από τις καθημερινές αφού ο κόσμος παραμένει στις οικίες του.
- Οι θερμοκρασίες είναι πιο υψηλές τους καλοκαιρινούς μήνες κυρίως τις μεσημεριανές ώρες.
- Το χειμώνα έχουμε πιο δυνατούς ανέμους παρά τους καλοκαιρινούς μήνες.

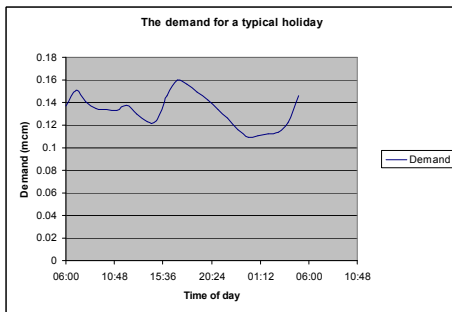
Αυτά τα στοιχεία επιβεβαιώνονται από τις γραφικές παραστάσεις που φαίνονται στην εικόνα 2.2.1.



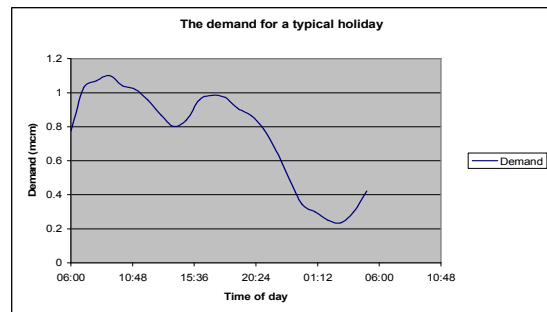
(α)



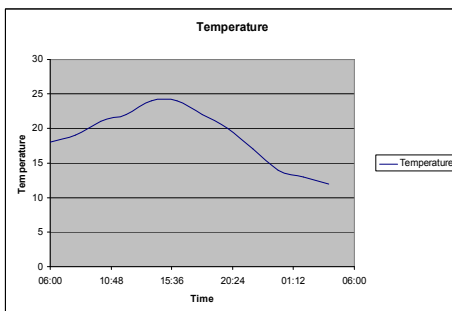
(β)



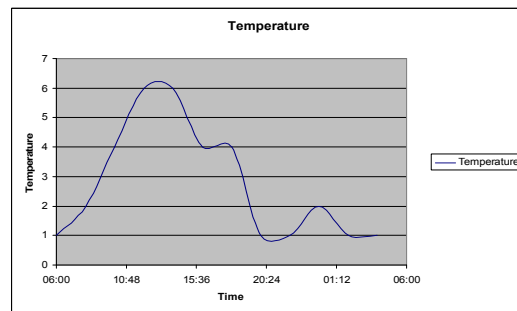
(γ)



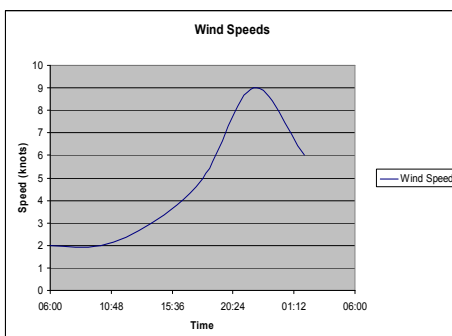
(δ)



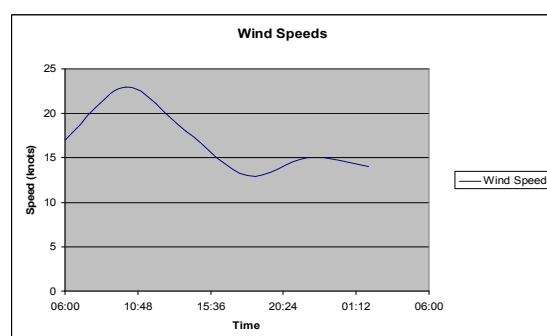
(ϵ)



(ζ)



(η)

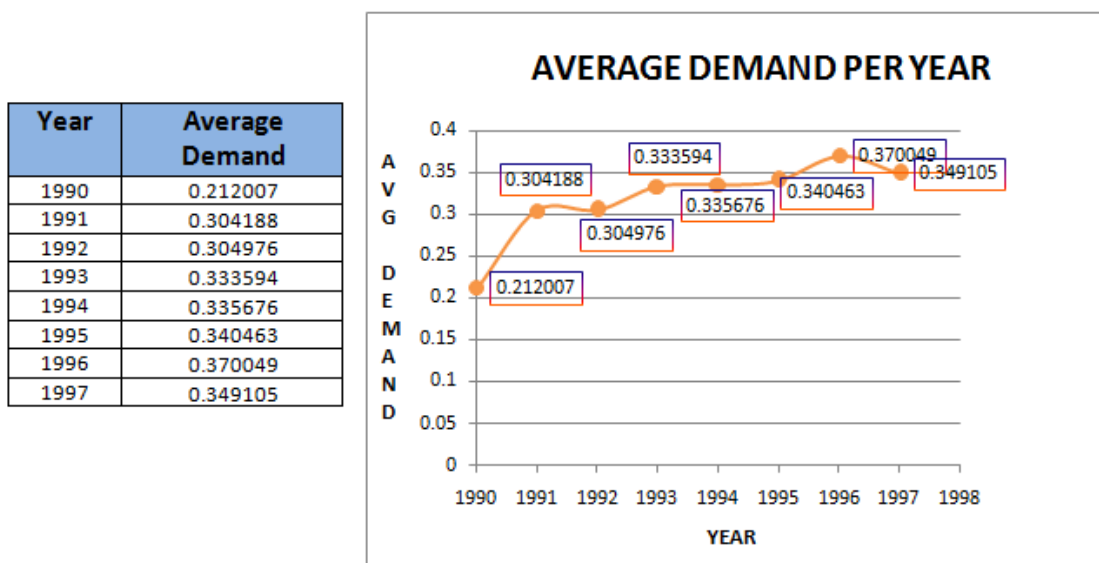


(θ)

- (α) – Ωριαία ζήτηση φυσικού αερίου για χειμερινή εργάσιμη μέρα
- (β) – Ωριαία ζήτηση φυσικού αερίου για καλοκαιρινή εργάσιμη μέρα
- (γ) – Ωριαία ζήτηση φυσικού αερίου για καλοκαιρινή μη εργάσιμη μέρα
- (δ) – Ωριαία ζήτηση φυσικού αερίου για χειμερινή μη εργάσιμη μέρα
- (ε) – Ωριαία τιμή θερμοκρασίας για καλοκαιρινή εργάσιμη μέρα
- (ζ) – Ωριαία τιμή θερμοκρασίας για χειμερινή εργάσιμη μέρα
- (η) – Ωριαία ταχύτητα ανέμων για καλοκαιρινή εργάσιμη μέρα
- (θ) – Ωριαία ταχύτητα ανέμων για χειμερινή εργάσιμη μέρα

Εικόνα 2.2.1 Γραφήματα μελέτης δεδομένων

Επιπλέον, από το διάγραμμα της ζήτησης του φυσικού αερίου ανά έτος που φαίνεται στην εικόνα 2.2.2 παρατηρούμε ότι η ζήτηση αυξάνεται με την πάροδο του χρόνου, πράγμα που δείχνει ότι το πλήθος των σπιτιών αυξάνεται και συνεπώς αυξάνεται και ο αριθμός των πελατών.



Εικόνα 2.2.2 Μέση ζήτηση φυσικού αερίου ανά έτος

Προ επεξεργασία δεδομένων

Τα δεδομένα ήταν σε μια ακατέργαστη μορφή με αποτέλεσμα να αντιμετωπίσουμε κάποια προβλήματα. Συγκεκριμένα είχαμε τιμές οι οποίες απουσίαζαν. Επίσης ο

αριθμός των κενών διαστημάτων στα αρχεία δεν ήταν σταθερός μ' αποτέλεσμα να υπάρχει δυσκολία στη σάρωση τους.

Για την επίλυση των προβλημάτων αυτών χρησιμοποιήσαμε τα δύο εργαλεία της προηγούμενης έρευνας [4] και έτσι καταφέραμε να πάρουμε τα ίδια δεδομένα μ' αυτήν. Με το πρώτο εργαλείο καταφέραμε να συμπληρώσουμε τις κενές τιμές. Για να το επιτύχουμε πήραμε δείγματα τιμών από τις ίδιες μέρες και ώρες άλλων χρόνων, πήραμε ένα μέσο όρο από τις τιμές για τις ίδιες ώρες σε άλλα έτη.

Τέλος με το δεύτερο εργαλείο καταφέραμε να δημιουργήσουμε ένα ενιαίο αρχείο με κατάλληλη δομή έτσι που να γίνεται πιο εύκολη και αποδοτική η ανάγνωση του από το πρόγραμμα.

2.3 Προτεινόμενη Λύση

Όπως είναι γνωστό μια γρήγορη και όσο το δυνατό πιο ακριβής πρόβλεψη της ζήτησης του φυσικού αερίου μπορεί να αποδειχθεί ευεργετική για τις εταιρίες παροχής του αφού μ' αυτό τον τρόπο θα μπορέσουν να αυξήσουν τα έσοδα τους μειώνοντας παράλληλα τα έξοδα τους. Στην προηγούμενη διπλωματική εργασία [4] μελετήθηκαν δύο μέθοδοι για πρόβλεψη της ζήτησης, αυτή του κινητού παραθύρου και ένα στατικό δίκτυο εκπαιδευμένο με τη χρήση του αλγορίθμου ανάστροφης μετάδοσης του σφάλματος back propagation. Όπως είπαμε και στην εισαγωγή η έρευνα καταπιάνεται με τη χρήση άλλων τεχνικών για πρόβλεψη. Συγκεκριμένα μελετήσαμε τρεις διαφορετικές μεθόδους:

- Πρόβλεψη με τη χρήση δικτύου αξονικών συναρτήσεων αξονικών βάσεων (RNF) [6][7][8].
- Δημιουργία υβριδικού δικτύου με συνδυασμό του αλγορίθμου SOM Kohonen [11] –για εύρεση του πλήθους και των θέσεων των κέντρων - και δικτύου συναρτήσεων αξονικών βάσεων [6][7][8].
- Μελέτη με χρήση δικτύου ανατροφοδότησης τύπου Jordan [13] εκπαιδευμένο με τον αλγόριθμο ανάστροφης μετάδοσης του λάθους στο χρόνο (back propagation through the time) [14][15].

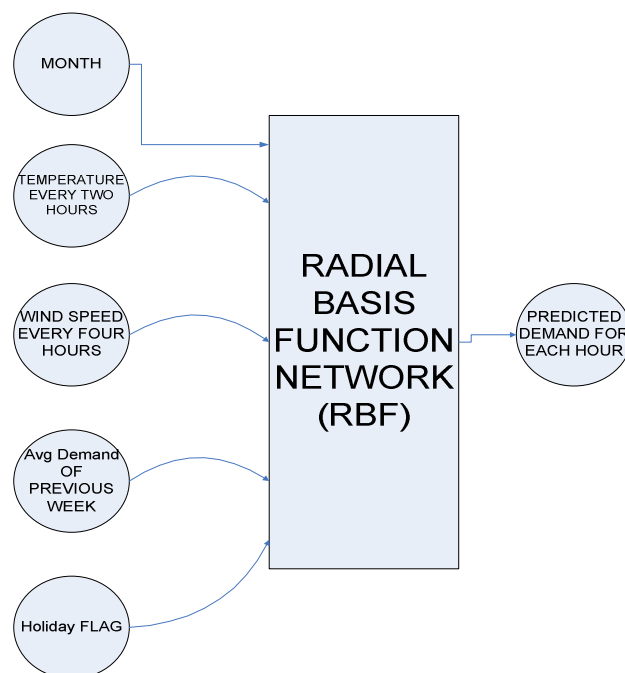
Σ' αυτήν την υπό ενότητα θα αναλύσουμε την αρχιτεκτονική για κάθε ένα από τα τρία πιο πάνω δίκτυα.

Πρόβλεψη με τη χρήση δικτύου συναρτήσεων αξονικών βάσεων

Σ' αυτό το μέρος της έρευνας αναπτύχθηκε δίκτυο συναρτήσεων αξονικών βάσεων (RBF) [6][7][8] για την πρόβλεψη της ζήτησης. Από τη μελέτη των δεδομένων καταλήξαμε στις ακόλουθες εισόδους για το δίκτυο μας:

- Μήνας, κωδικοποιημένος σε δυαδική μορφή (π.χ., Ιανουάριος → 0001)
- Θερμοκρασία για κάθε δύο ώρες
- Ταχύτητα ανέμων για κάθε τέσσερις ώρες
- Μέση ζήτηση φυσικού αερίου για την προηγούμενη εβδομάδα
- Είσοδος που υποδηλώνει αν μια μέρα είναι αργία ή όχι (1 για αργία ,0 για καθημερινή)

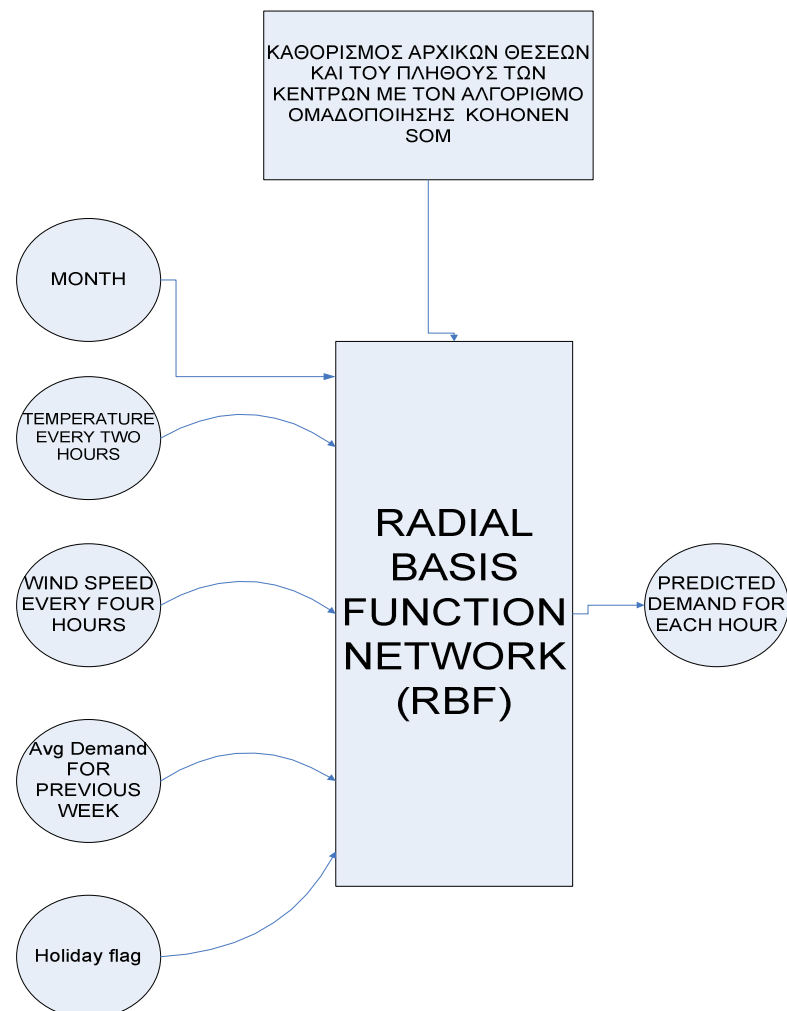
Ως έξοδος για το δίκτυο θα είναι η προβλεπόμενη **ωριαία** τιμή ζήτησης φυσικού αερίου. Στο σχήμα 2.3.1 παρουσιάζεται διαγραμματικά η αρχιτεκτονική του δικτύου συναρτήσεων αξονικών βάσεων.



Σχήμα 2.3.1 Αρχιτεκτονική δικτύου RBF

Δημιουργία υβριδικού δικτύου με συνδυασμό του Kohonen SOM και δικτύου αξονικών βάσεων

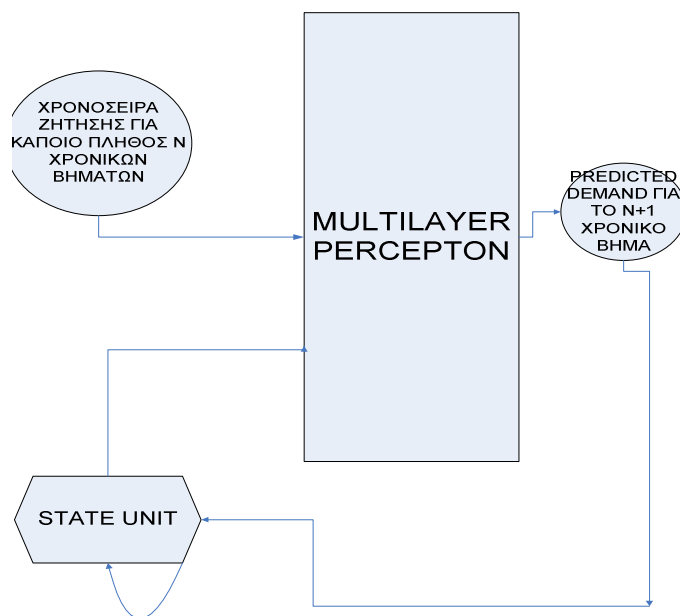
Εδώ πειραματιστήκαμε όσον αφορά τον τρόπο επιλογής του πλήθους και θέσεων των κέντρων. Στο προηγούμενο δίκτυο τα κέντρα επιλέγονται τυχαία καθώς επίσης το πλήθος τους καθορίζεται από το χρήστη. Εδώ χρησιμοποιούμε τον αλγόριθμο ομαδοποίησης Kohonen SOM [11] για να καθορίσουμε τις αρχικές θέσεις των κέντρων. Αφού γίνει αυτό χρησιμοποιούμε το δίκτυο του σχήματος 2.3.1 με τις θέσεις των κέντρων να είναι αυτές που καθορίστηκαν μέσω του αλγορίθμου ομαδοποίησης Kohonen SOM [11].



Σχήμα 2.3.2 Αρχιτεκτονική υβριδικού δικτύου Kohonen SOM - RBF

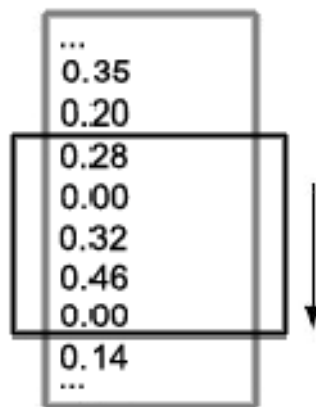
Δίκτυο ανατροφοδότησης τύπου Jordan εκπαιδευμένο με τον αλγόριθμό αναστροφής μετάδοσης του σφάλματος στο χρόνο (back propagation through time)

Τέλος, πειραματιστήκαμε με τη δημιουργία ενός δικτύου με ανατροφοδότηση (recurrent network) τύπου Jordan [11] το οποίο και εκπαιδεύτηκε με τη χρήση του αλγορίθμου back αναστροφής μετάδοσης του σφάλματος στο χρόνο (back propagation through time - BPTT) [13][14]. Στο δίκτυο αυτό η προηγούμενη πρόβλεψη είναι είσοδος στην επόμενη επανάληψη, δηλαδή μπορούμε να πούμε ότι η επόμενη έξοδος εξαρτάται από την προηγούμενη. Στο δίκτυο αυτό θα χρησιμοποιήσουμε μόνο τα δεδομένα της χρονοσειράς της ζήτησης αφού όπως είναι γνωστό τα δίκτυα ανατροφοδότησης δεν είναι καλά με τη χρήση στατικών δεδομένων. Στο σχήμα 2.3.3 που ακολουθεί βλέπουμε την αρχιτεκτονική αυτού του δικτύου.



Σχήμα 2.3.3 Αρχιτεκτονική δικτύου με ανατροφοδότηση (Jordan Recurrent Network)

Στο δίκτυο αυτό υπάρχει ένα πλήθος χρονικών βημάτων στα δεδομένα εισόδου, το οποίο κινείται μέσα στα δεδομένα καθώς περνάει ο χρόνος και ουσιαστικά καθορίζει το πλήθος των εισόδων. Σε αυτή την μέθοδο θα χρησιμοποιήσουμε μόνο τα δεδομένα της χρονοσειράς της ζήτησης του φυσικού αερίου. Με αυτό το τρόπο θα εκμεταλλευτούμε την σχέση της ζήτησης του φυσικού αερίου με τον χρόνο. Τα δεδομένα μέσα στο παράθυρο παρουσιάζονται στο δίκτυο ως είσοδοι και η επιθυμητή έξοδος είναι η αμέσως επόμενη τιμή μετά το τέλος του παραθύρου. Μεταβάλλοντας το πλήθος των χρονικών βημάτων καθορίζουμε το πόσο πίσω στον χρόνο θα κυμαίνονται τα δεδομένα εισόδου. Στην εικόνα 2.3.1 παρουσιάζεται ένα παράδειγμα των δεδομένων εισόδου για 5 χρονικά βήματα. Συγκεκριμένα οι είσοδοι του δικτύου θα είναι οι τιμές που φαίνονται στο πλαίσιο και στόχος μας είναι η πρόβλεψη της αμέσως επόμενης τιμής της ζήτησης, δηλαδή 0.14 για το συγκεκριμένο παράδειγμα.



Εικόνα 2.3.1 Παράδειγμα δεδομένων χρονοσειράς εισόδου για το δίκτυο με ανατροφοδότηση (Jordan Recurrent Network) για 5 χρονικά βήματα - εισόδους

Επιπλέον, στο σχήμα 2.3.3 παρατηρούμε ότι στη κατάσταση ανατροφοδότησης (state unit), υπάρχει ανάδραση στην ίδια την κατάσταση. Αυτό συνεπάγεται τη χρήση μιας **χρονικής σταθεράς (T)**, από την οποία εξαρτάται η τιμή της εκθετικής συνάρτησης. Η σταθερά αυτή πρέπει να είναι θετική ($T > 0$) και ουσιαστικά καθορίζει τη διάρκεια της μνήμης (memory depth), δηλαδή για πόσο μια τιμή στο state unit θα «απομνημονεύεται». Συγκεκριμένα, η αλλαγή που γίνεται στη τιμή των καταστάσεων ανατροφοδότησης (state units), εξαρτάται από την εκθετική συνάρτηση και παρουσιάζεται στην εξίσωση (26) που ακολουθεί.

$$SU(t) = SU_0 e^{-t/T} \quad (26)$$

- $SU(t)$ αντιστοιχεί στη νέα τιμή του context unit
- SU_0 είναι η τρέχουσα τιμή του context unit
- t αντιστοιχεί στη τρέχουσα χρονική στιγμή
- T είναι η χρονική σταθερά (time constant)

Τέλος, για μεγάλες τιμές της σταθεράς T το δίκτυο θα έχει μεγαλύτερο εύρος μνήμης σε αντίθεση με μικρότερες τιμές όπου το δίκτυο θα έχει βραχυπρόθεσμη μνήμη.

Κεφάλαιο 3

Αναλυτικός Σχεδιασμός – Υλοποίηση

3.1 Περιγραφή	40
3.2 Σχεδιασμός	40
3.2.1 Διαγράμματα UML	41
3.2.2 Επεκτασιμότητα	44
3.3 Υλοποίηση	45
3.3.1 Επεξήγηση κλάσεων	45

3.1 Περιγραφή

Στο κεφάλαιο αυτό θα αναφερθούμε στον τρόπο που σχεδιάσαμε και υλοποιήσαμε το λογισμικό για πρόβλεψη της ζήτησης του φυσικού αερίου. Αρχικά, μετά από μελέτη των θετικών και αρνητικών στοιχείων των διαφόρων γλώσσών προγραμματισμού καταλήξαμε στην επιλογή μια αντικειμενοστρεφούς γλώσσας, της Java. Έτσι μ' αυτόν τον τρόπο το πρόγραμμα μας θα είναι πιο κατανοητό, η επαναχρησιμοποίηση θα είναι πιο εύκολη καθώς και θα διευκολύνεται σε μεγάλο βαθμό η συντήρηση.

Έτσι αφού στηριχτήκαμε στα πιο πάνω υλοποιήσαμε το GPS (Gas Prediction Software). Στη συνέχεια αυτού του κεφαλαίου θα παραθέσουμε τα διαγράμματα UML του λογισμικού αυτού και επίσης θα περιγράψουμε πώς το πρόγραμμα μπορεί πολύ εύκολα να επεκταθεί.

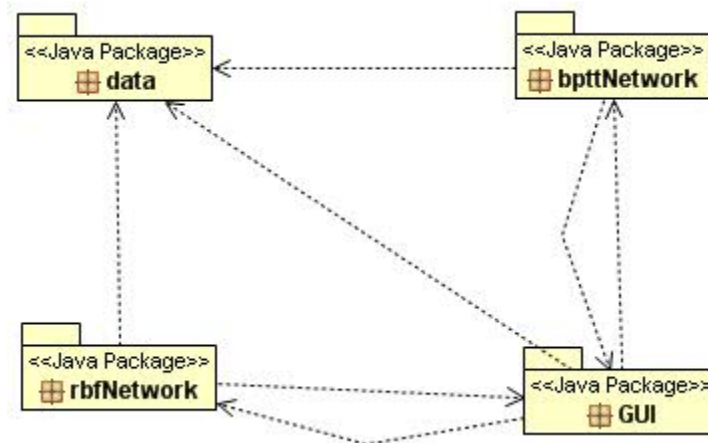
Στο δεύτερο μέρος του κεφαλαίου θα επικεντρωθούμε στη υλοποίηση του λογισμικού για πρόβλεψη της ζήτησης του φυσικού αερίου περιγράφοντας τις κλάσεις καθώς και τις μεθόδους τους. Έτσι με τη χρήση των πιο αποδοτικών τεχνικών σχεδιασμού και υλοποίησης προσπαθήσαμε να φτιάξουμε ένα λογισμικό το οποίο να είναι φιλικό προς το χρήστη, επαναχρησιμοποιήσιμο και πάνω απ' όλα εύχρηστο.

3.2 Σχεδιασμός

Όπως αναφέραμε και στην περιγραφή του κεφαλαίου σ' αυτό το μέρος θα επικεντρωθούμε στον αντικειμενοστρεφή σχεδιασμό του λογισμικού (3.2.1) καθώς επίσης και πώς αυτό βοηθάει στην εύκολη επεκτασιμότητα (3.2.2) η οποία είναι μια από τις πιο σημαντικότερες προϋποθέσεις που πρέπει να τηρούνται για τη σωστή ανάπτυξη ενός λογισμικού.

3.2.1 Διαγράμματα UML

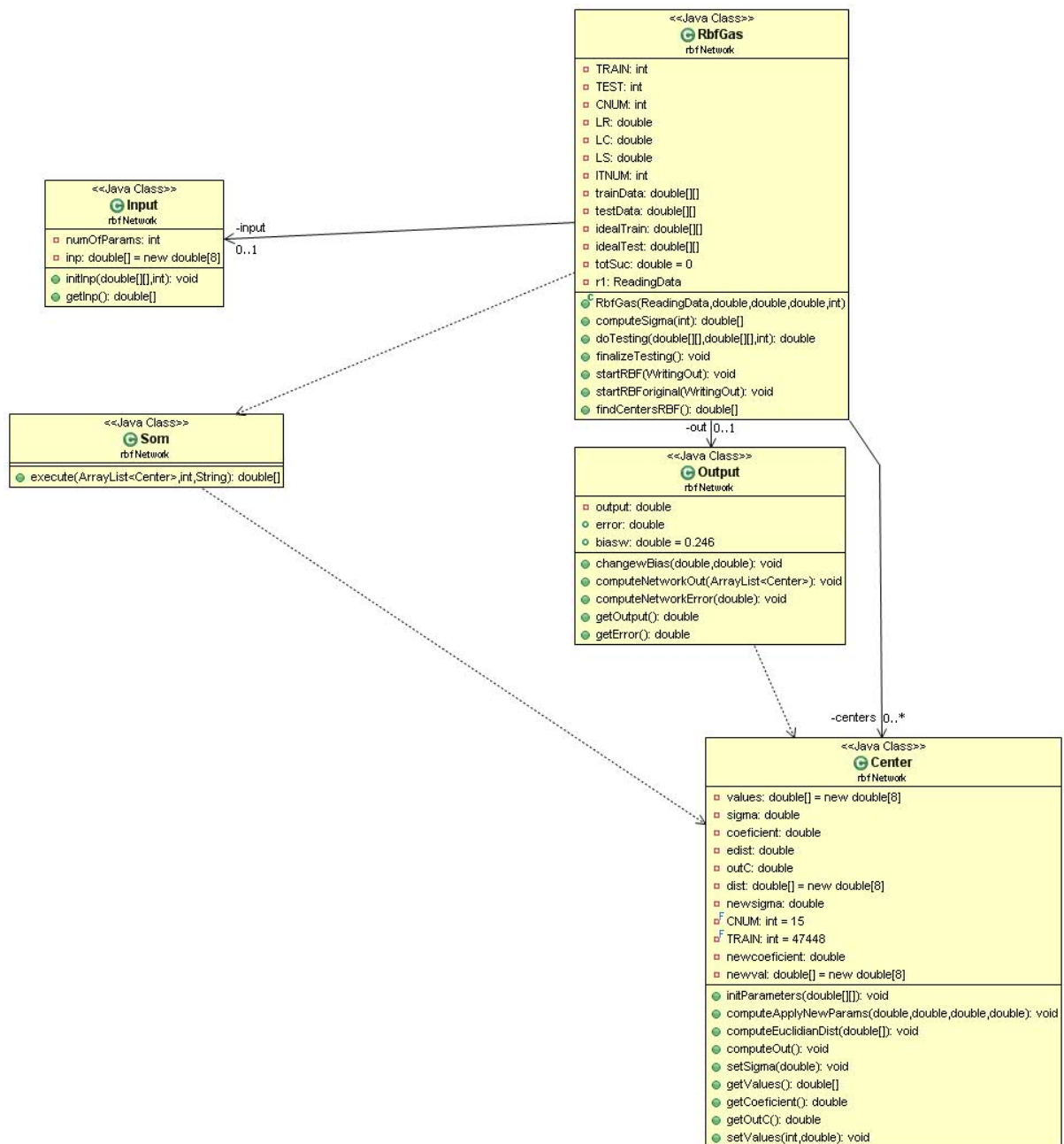
Αρχίζοντας αυτήν την υποενότητα θα παραθέσουμε το γενικό διάγραμμα UML που δείχνει τον τρόπο επικοινωνίας μεταξύ των υποομάδων του λογισμικού.



Διάγραμμα 3.2.1.1 Γενικό διάγραμμα UML του λογισμικού

Παρατηρούμε ότι το λογισμικό αποτελείται από τέσσερις κύριες υπό μονάδες – πακέτα. Το πρώτο πακέτο (data) περιέχει όλες τις κλάσεις και μεθόδους οι οποίες σχετίζονται με την ανάγνωση δεδομένων καθώς επίσης και εγγραφή δεδομένων σε αρχείο. Το πακέτο bpttNetwork περιέχει όλες τις κλάσεις και μεθόδους οι οποίες σχετίζονται με τη δημιουργία του δικτύου με ανάδραση καθώς επίσης και την υλοποίηση του αλγορίθμου back propagation through time. Το πακέτο rbfNetwork περιέχει όλες τις κλάσεις και μεθόδους οι οποίες σχετίζονται με την δημιουργία του δικτύου RBF καθώς επίσης και την υλοποίηση του αλγορίθμου με μεταβλητά κέντρα στο δίκτυο αυτό. Τέλος το πακέτο GUI περιέχει τις κλάσεις και τις μεθόδους οι οποίες καθορίζουν τη διεπαφή του λογισμικού με το χρήστη. Στη συνέχεια θα παρουσιάσουμε αναλυτικότερα διαγράμματα UML για τα πιο πάνω πακέτα. Το πακέτο data περιέχει τις κλάσεις ReadingData και writeOut (περιγράφονται στην υλοποίηση) δεν συσχετίζονται μεταξύ τους απλώς αλληλεπιδρούν με τ' άλλα πακέτα όπως φαίνεται πιο πάνω. Το πακέτο bpttNetwork περιέχει μόνο την κλάση RNNetwork η οποία περιγράφεται στη συνέχεια.

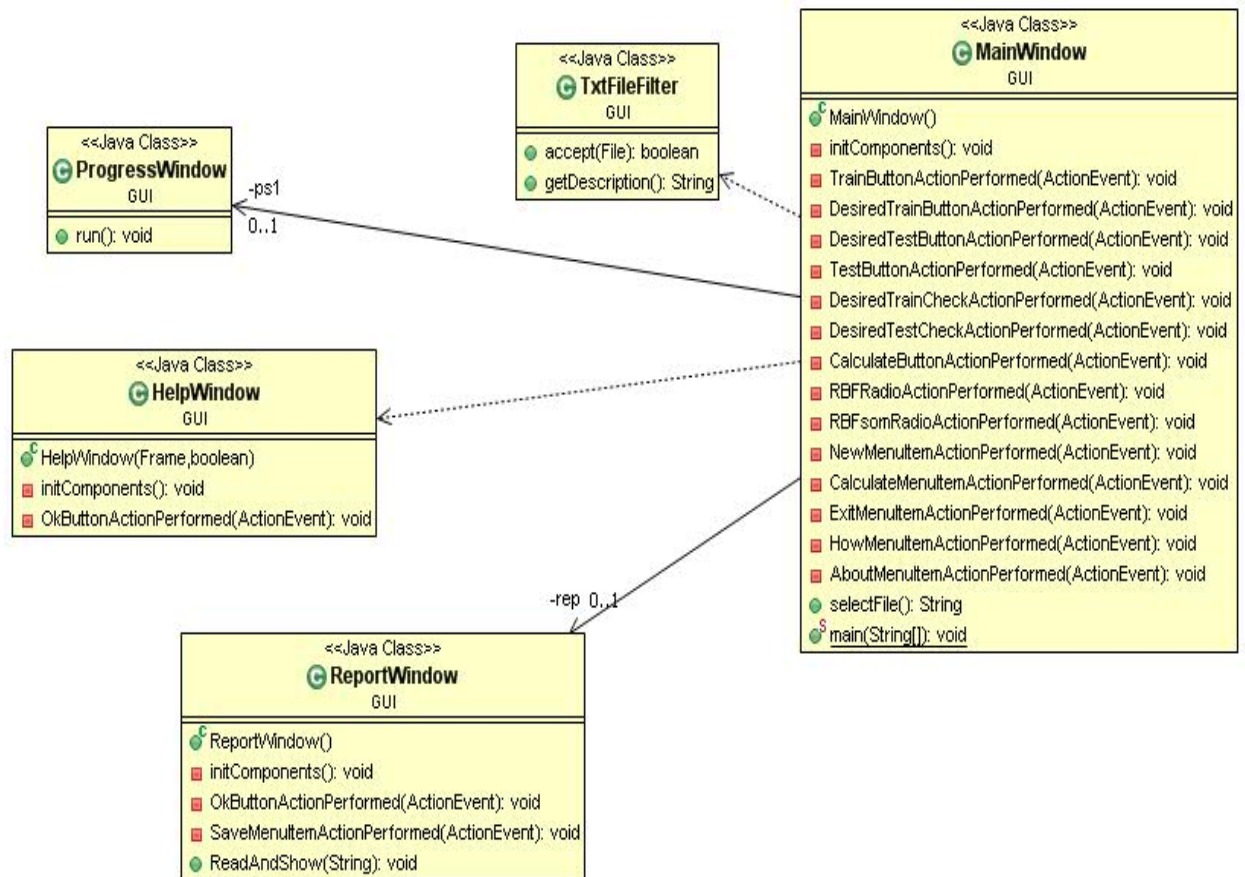
Στο διάγραμμα που ακολουθεί φαίνεται ο τρόπος επικοινωνίας των κλάσεων που βρίσκονται στο πακέτο rbfNetwork.



Διάγραμμα 3.2.1.2 Διάγραμμα UML του δικτύου RBF

Στο πιο πάνω διάγραμμα παρουσιάζεται πως επικοινωνούν οι κλάσεις για το δίκτυο RBF. Παρατηρούμε ότι έχουμε μια κεντρική κλάση η οποία στην ουσία αντιπροσωπεύει το δίκτυο RBF. Συγκεκριμένα κάθε στιγμιότυπο αυτής της κλάσης

πρέπει να έχει μέχρι ένα διάνυσμα εισόδου (κλάση Input) και εξόδου (κλάση Output) καθώς επίσης από μηδέν μέχρι πολλά κέντρα (κλάση Center).



Διάγραμμα 3.2.1.3 Διάγραμμα UML της διεπαφής (GUI) με το χρήστη

Στο διάγραμμα 3.2.1.3 παρουσιάζεται πως επικοινωνούν οι κλάσεις, τον έλεγχο της διεπαφής (GUI) του χρήστη με το λογισμικό. Παρατηρούμε ότι και σ' αυτήν την περίπτωση έχουμε μια κεντρική κλάση η οποία στην ουσία ελέγχει τη διεπαφή. Η κλάση αυτή ονομάζεται MainWindow και περιέχει τις μεθόδους για έλεγχο των κουμπιών της διεπαφής. Επίσης η κλάση αυτή επικοινωνεί με τις τέσσερις πιο κάτω κλάσεις:

- TxtFileFilter – υπεύθυνη για φόρτωση των αρχείων
- ProgressWindow – υπεύθυνη για την εμφάνιση και έλεγχο της μπάρας προόδου

- HelpWindow – υπεύθυνη για την εμφάνιση των πληροφοριών χρήσης του λογισμικού
- ReportWindow – υπεύθυνη για την εμφάνιση και έλεγχο της τελικής αναφοράς της εκπαίδευσης.

3.2.2 Επεκτασιμότητα

Η γλώσσα η οποία επιλέχτηκε είναι η java. Ο λόγος είναι γιατί είναι αντικειμενοστρεφής και προσφέρει όλα τα πλεονεκτήματα του αντικειμενοστρεφούς προγραμματισμού. Ένα από αυτά είναι η επεκτασιμότητα, η επαναχρησιμοποίηση του κώδικα κτλ.

Πιο συγκεκριμένα χρησιμοποιήθηκαν όλες οι αρχές της αντικειμενοστρεφούς σχεδίασης οι οποίες είναι βασικές στον κλάδο της Τεχνολογίας Λογισμικού. Αυτό βοηθά στο ότι μπορεί να χρησιμοποιηθεί ο κώδικας για επέκταση ή τροποποίηση απο άλλα άτομα. Οι αρχές είναι πολλές γι αυτό και επιλέχθηκαν οι πιο σημαντικές. Αυτές αναφέρονται τεκμηριωμένα παρακάτω:

1. Αρχή της Ανοικτής – Κλειστής Σχεδίασης: Σ αυτή εδώ την αρχή οι οντότητες λογισμικού θα πρέπει να είναι ανοικτές για επέκταση και κλειστές για τροποποίηση. Το λογισμικό GPS πληρεί αυτή την αρχή εφόσον για περεταίρω αλλαγές στο λογισμικό αυτές επιτυγχάνονται με την προσθήκη νέου κώδικα και όχι με την τροποποίηση υπάρχοντος κώδικα που ήδη λειτουργεί. Η τεκμηρίωση για αυτή την αρχή είναι οι αφαιρέσεις που χρησιμοποιούν οι μονάδες λογισμικού του GPS. Για παράδειγμα οι κλάσεις ReadData και WriteData είναι λογικό να χρησιμοποιηθούν και από την επέκταση του GPS σε νέους αλγορίθμους. Με την τήρηση αυτής της αρχής εξασφαλίζουμε ότι οι αλλαγές θα πραγματοποιηθούν με τη μικρότερη δυνατή προσπάθεια και χρόνο.
2. Αρχή της Ενσωμάτωσης: Σ αυτήν εδώ την αρχή θα πρέπει η εσωτερική κατάσταση ενός αντικειμένου να είναι τροποποιήσιμη μόνο μέσω της δημόσιας διασύνδεσης του. Σύμφωνα με την ανωτέρω αρχή, οι ιδιότητες πρέπει να μην είναι προσπελάσιμες εκτός της κλάσης, δηλαδή η τροποποίηση των τιμών των ιδιοτήτων θα πρέπει να επιτρέπεται μόνο μέσω της πρόσβασης που περιέχουν οι δημόσια διαθέσιμες μέθοδοι.

Η αρχή της ενσωμάτωσης υλοποιείται απλά θέτοντας την ορατότητα όλων των ιδιοτήτων ως ιδιωτική. Έτσι τα ιδιωτικά μέλη δεδομένων αποκρύπτονται κατά συνέπεια από τους προγραμματιστές άλλων κλάσεων ή άλλων συστημάτων (παραμένει ωστόσο η δυνατότητα προσπέλασης τους από άλλα αντικείμενα της ίδιας κλάσης).

Το πλεονέκτημα που προκύπτει από την εφαρμογή της παραπάνω αρχής σε όλες της μονάδες του GPS είναι ότι διατηρείται η εγκυρότητα και συνέπεια της κατάστασης ενός αντικειμένου. Και αυτό βοηθά στο ότι η εγκυρότητα αυτή μπορεί να διατηρηθεί σε όλες τις χρονικές στιγμές που μπορεί να τρέχει το GPS.

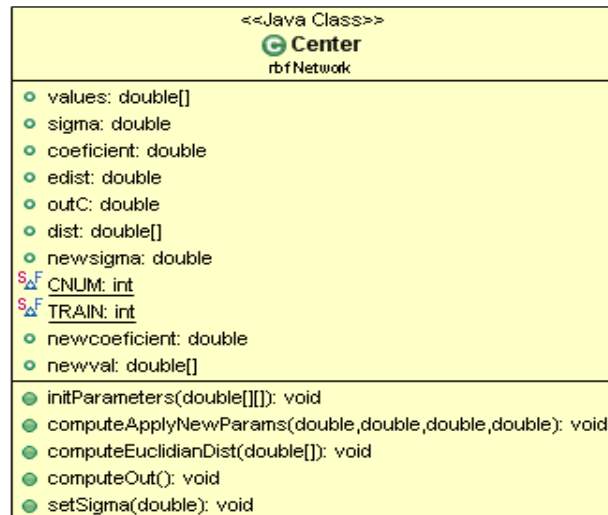
3. Αρχή της Χαμηλής Σύζευξης: Σ αυτή την αρχή θα πρέπει σε ένα σχέδιο λογισμικού να επιδιώκεται η επίτευξη της μικρότερης δυνατής σύζευξης μεταξύ των συστατικών του.

3.3 Υλοποίηση

Σ' αυτήν την υποενότητα θα επικεντρωθούμε σε κάθε μια κλάση ξεχωριστά περιγράφοντας παράλληλα τα χαρακτηριστικά τους στοιχεία καθώς επίσης και τις διάφορες μεθόδους.

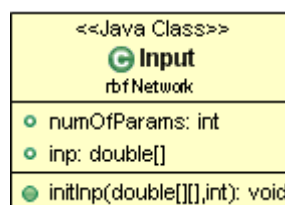
3.3.1 Επεξήγηση κλάσεων

Δίκτυο RBF: Για σκοπό υλοποίησης του δικτύου RBF υλοποιήθηκαν οι κλάσεις Center, Input, Output, ReadingData και WritingOut. Στις σελίδες που ακολουθούν θα περιγράψουμε τις παραπάνω κλάσεις.



Εικόνα 3.3.1.1 Η κλάση Center

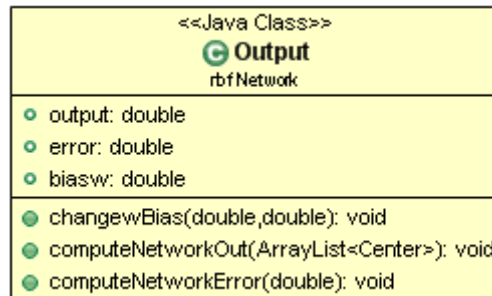
Στην εικόνα 3.3.1.1 παρουσιάζεται η κλάση Center. Τα χαρακτηριστικά αυτής της κλάσης (με τη σειρά που παρουσιάζονται στην εικόνα) είναι : το διάνυσμα με τις θέσεις του κέντρου, η διασπορά του κέντρου, το βάρος που συνδέει το κέντρο με την έξοδο, την ευκλείδεια απόσταση, την έξοδο του κέντρου, η νέα τιμή της διασποράς , η νέα τιμή για τα βάρη καθώς και η νέα τιμή για τις θέσεις των κέντρων. Οι σταθερές CNUM και TRAIN είναι στο πλήθος των κέντρων και των δεδομένων εκπαίδευσης αντίστοιχα. Η κλάση αυτή έχει τέσσερις μεθόδους. Η πρώτη μέθοδος η `initParameters` αρχικοποιεί τα βάρη με μικρές τυχαίες τιμές. Η `computeApplyNewParams` υπολογίζει βάσει των εξισώσεων που περιγράφηκαν στο κεφάλαιο 2, τις νέες τιμές των παραμέτρων και ανανεώνει τις παλιές τους τιμές. Η μέθοδοι `computeEuclidianDist` και `computeOut` υπολογίζουν την ευκλείδεια απόσταση και την έξοδο του κέντρου αντίστοιχα. Τέλος, η μέθοδος `setSigma` αντικαθιστά την τιμή της διασποράς με μια νέα τιμή που παίρνει ως παράμετρο.



Εικόνα 3.3.1.2 Η κλάση Input

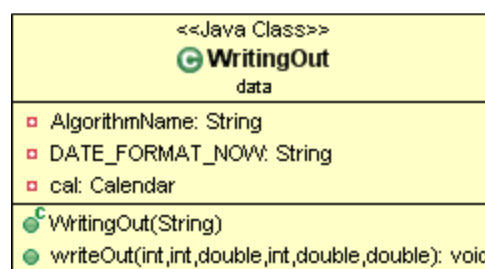
Στην εικόνα 3.3.1.2 παρουσιάζεται η κλάση Input. Τα χαρακτηριστικά αυτής της κλάσης (με τη σειρά που παρουσιάζονται στην εικόνα) είναι : το πλήθος των εισόδων

και το διάνυσμα με τις τιμές εισόδου. Η μέθοδος `initInp` αρχικοποιεί το διάνυσμα των τιμών εισόδου με τις αντίστοιχες τιμές από τα δεδομένα εκπαίδευσης ή επαλήθευσης. Τα δεδομένα αυτά αντιστοιχούν στην πρώτη παράμετρο ενώ η θέση μέσα στο πλήθος των δεδομένων εκφράζεται με τη δεύτερη παράμετρο της μεθόδου.



Εικόνα 3.3.1.3 Η κλάση Output

Στην εικόνα 3.3.1.3 παρουσιάζεται η κλάση Output. Τα χαρακτηριστικά αυτής της κλάσης (με τη σειρά που παρουσιάζονται στην εικόνα) είναι : η τελική έξοδος , το σφάλμα του δικτύου και το βάρος Bias. Η μέθοδος `changeBias` αλλάζει την τιμή του βάρους προς το συγκεκριμένο νευρώνα εξόδου. Οι μέθοδοι `computeNetworkError` και `computeNetworkOutput` υπολογίζουν το σφάλμα και την τελική έξοδο του δικτύου αντίστοιχα.



Εικόνα 3.3.1.4 Η κλάση WritingOut

Στην εικόνα 3.3.1.4 παρουσιάζεται η κλάση WritingOut. Τα χαρακτηριστικά αυτής της κλάσης (με τη σειρά που παρουσιάζονται στην εικόνα) είναι : το όνομα του αλγορίθμου, η τρέχουσα ημερομηνία καθώς και ώρα. Η κλάση αυτή χρησιμοποιείται για την εγγραφή των αποτελεσμάτων κάθε εποχής στο αρχείο εξόδου. Ο

κατασκευαστής της κλάσης αυτής στην ουσία αρχικοποιεί τα χαρακτηριστικά της ενώ η μέθοδος writeOut γράφει στο αρχείο τ' αποτελέσματα της εκπαίδευσης.



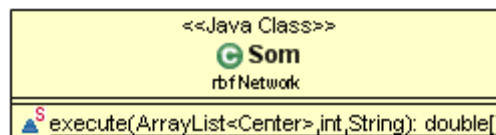
Εικόνα 3.3.1.5 Η κλάση ReadingData

Στην εικόνα 3.3.1.5 παρουσιάζεται η κλάση ReadingData. Τα χαρακτηριστικά αυτής της κλάσης (με τη σειρά που παρουσιάζονται στην εικόνα) είναι : το πλήθος των δεδομένων εκπαίδευσης και επαλήθευσης , το πλήθος των επιθυμητών αποτελεσμάτων

για εκπαίδευση και επαλήθευση, το πλήθος των εξόδων, τους πίνακες με τα δεδομένα εκπαίδευσης και επαλήθευσης καθώς επίσης και τα ονόματα των αρχείων με τα δεδομένα. Η κλάση αυτή χρησιμοποιείται για την ανάγνωση των δεδομένων εκπαίδευσης και επαλήθευσης μέσα από τ' αρχεία. Όπως φαίνεται και στην εικόνα υπάρχουν δυο κατασκευαστές. Ο πρώτος κατασκευαστής της κλάσης αυτής στην ουσία αρχικοποιεί τα χαρακτηριστικά της και τις παραμέτρους για το δίκτυο RBF ενώ ο δεύτερος αρχικοποιεί τις ιδιότητες για το δίκτυο με ανάδραση. Η μέθοδος InitData αρχικοποιεί τους πίνακες με τα δεδομένα εκπαίδευσης και επαλήθευσης από τα καθορισμένα αρχεία. Η μέθοδος Read διαβάζει μέσα από τη διεπαφή τα ονόματα των αρχείων που καθόρισε ο χρήστης. Τέλος η μέθοδος check κάνει έλεγχο αν οι γραμμές των αρχείων με τα δεδομένα και τα επιθυμητά αποτελέσματα ταυτίζονται τόσο για τα δεδομένα εκπαίδευσης όσο και για τα δεδομένα επαλήθευσης.

Υβριδικό δίκτυο SOM/RBF:

Στο δίκτυο αυτό γίνεται χρήση όλων των πιο πάνω κλάσεων με διαφορά ότι χρησιμοποιείται μια επιπλέον κλάση που παρουσιάζεται πιο κάτω.

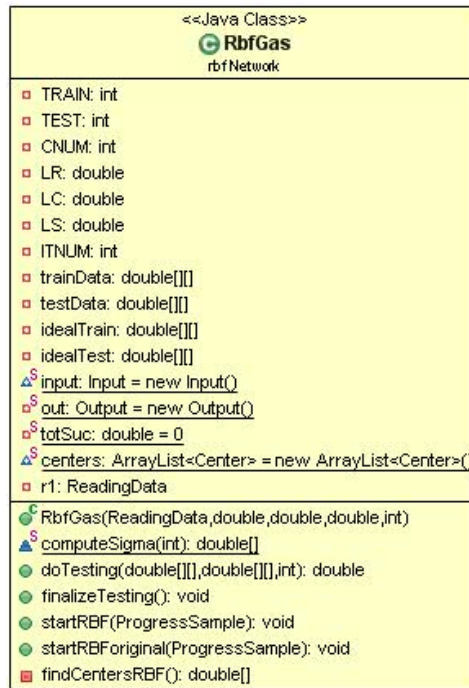


Εικόνα 3.3.1.6 Η κλάση SOM

Στην εικόνα 3.3.1.6 παρουσιάζεται η κλάση SOM. Η κλάση αυτή έχει μια μόνο μέθοδο η οποία εκτελεί τον αλγόριθμο ομαδοποίησης SOM που περιγράφεται στο κεφαλαίο 2 και με βάσει τα αποτελέσματα του αλγορίθμου αυτού κάνει αρχικοποίηση των κέντρων.

Κεντρική κλάση δικτύου RBF και υβριδικού δικτύου SOM-RBF

Στην εικόνα που ακολουθεί παρουσιάζεται η κεντρική κλάση η οποία συντονίζει και υλοποιεί τον αλγόριθμο εκπαίδευσης στα δίκτυα RBF.

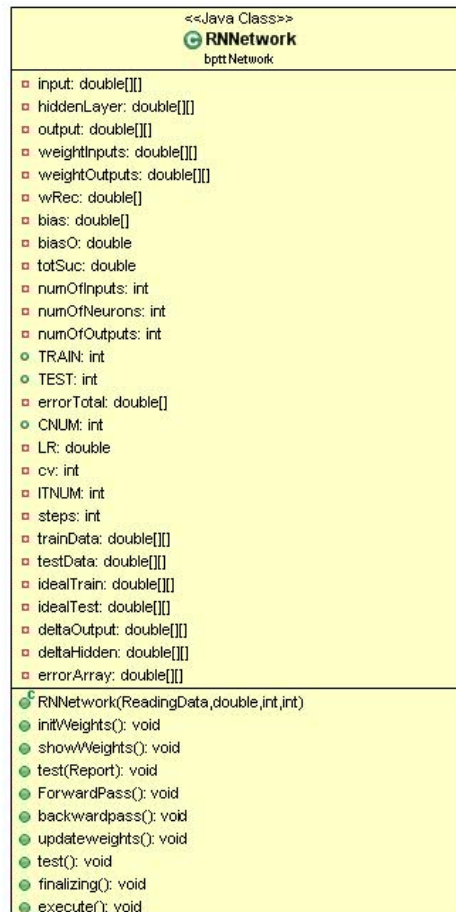


Εικόνα 3.3.1.7 Η κλάση RbfGas

Στην εικόνα 3.3.1.7 παρουσιάζεται η κλάση RbfGas. Τα χαρακτηριστικά αυτής της κλάσης (με τη σειρά που παρουσιάζονται στην εικόνα) είναι : το πλήθος των δεδομένων εκπαίδευσης, το πλήθος των δεδομένων επαλήθευσης, το πλήθος των κέντρων, οι ρυθμοί μάθησης για τα κέντρα τις διασπορές και τα βάρη, ο αριθμός των επαναλήψεων (εποχές), οι πίνακες με τα δεδομένα εκπαίδευσης και επαλήθευσης καθώς επίσης και οι πίνακες με τα επιθυμητά αποτελέσματα, οι νευρώνες εισόδου, οι νευρώνες εξόδου, το ποσοστό επιτυχίας και το στρώμα από τα κέντρα. Ο κατασκευαστής της κλάσης αρχικοποιεί όλες τις πιο πάνω ιδιότητες με τις κατάλληλες τιμές. Η μέθοδος computeSigma υπολογίζει την τιμή της διασποράς του κάθε κέντρου έτσι ώστε να είναι η καλύτερη. Η μέθοδος findCentersRBF αρχικοποιεί τις θέσεις των κέντρων τυχαία μέσα από το σύνολο των δεδομένων. Η συνάρτηση doTesting περνά τα δεδομένα επαλήθευσης μέσα από το δίκτυο και υπολογίζει το σφάλμα επαλήθευσης και το ποσοστό επιτυχίας. Οι μέθοδοι startRBF και StartRBForiginal υλοποιούν τον αλγόριθμο με μεταβλητά κέντρα(κεφάλαιο 2.1.1.2.1). Η πρώτη μέθοδος κάνει τυχαία αρχικοποίηση των κέντρων με τη χρήση της μεθόδου findCentersRBF ενώ η δεύτερη αρχικοποιεί τα κέντρα ανάλογα με τις κατηγορίες όπως αυτές προκύπτουν από τον αλγόριθμο κατηγοριοποίησης SOM. Τέλος, η μέθοδος finalizeTesting περνά από το

δίκτυο μια φορά τα δεδομένα επαλήθευσης και εκτυπώνει στο αρχείο την πραγματική και την προβλεπόμενη τιμή της ζήτησης.

Δίκτυο με ανατροφοδότηση από την έξοδο – Back Propagation Through Time (BPTT)



Εικόνα 3.3.1.8 Η κλάση RNNetwork

Στην εικόνα 3.3.1.8 παρουσιάζεται η κλάση RNNetwork. Τα χαρακτηριστικά αυτής της κλάσης (με τη σειρά που παρουσιάζονται στην εικόνα) είναι: ο πίνακας με τις εισόδους, ο πίνακας με τις εξόδους των νευρώνων των κρυφών επιπέδων για κάθε χρονικό βήμα, ο πίνακας με τις εξόδους για κάθε χρονικό βήμα, το στρώμα με βάρη από την είσοδο προς τους κρυφούς νευρώνες, το στρώμα με βάρη από τους κρυφούς προς την έξοδο, το στρώμα με βάρη από την προηγούμενη έξοδο με το κρυφό επίπεδο, οι τιμές του bias για τους κρυφούς νευρώνες, η τιμή του bias για την έξοδο, το τελικό ποσοστό επιτυχίας

του δικτύου, ο αριθμός των νευρώνων εισόδου, κρυφών και εξόδου, το πλήθος των δεδομένων εκπαίδευσης και επαλήθευσης, το συνολικό σφάλμα για κάθε χρονικό βήμα, ο ρυθμός μάθησης, μια μεταβλητή για μέτρημα των δεδομένων, το πλήθος των εποχών, το πλήθος το χρονικών βημάτων, οι πίνακες με τα δεδομένα εκπαίδευσης, επαλήθευσης καθώς και τα επιθυμητά αποτελέσματα, οι πίνακες με τα τοπικά σφάλματα (δ) για τους κρυφούς νευρώνες και νευρώνες εξόδου και το σφάλμα για κάθε χρονικό βήμα. Ο κατασκευαστής της κλάσης αρχικοποιεί όλες τις πιο πάνω ιδιότητες με τις κατάλληλες τιμές. Η μέθοδος `initWeights` αρχικοποιεί όλα τα στρώματα από βάρη με μικρές τυχαίες τιμές. Η μέθοδος `test` περνάει μέσα από το δίκτυο τα δεδομένα επαλήθευσης και υπολογίζει το ποσοστό επιτυχίας και το σφάλμα επαλήθευσης. Η `forwardPass` κάνει ένα μπροστινό πέρασμα μέσα από το δίκτυο και υπολογίζει τις τιμές εξόδου των κρυφών νευρώνων καθώς και τις τελικές εξόδου. Αντίθετα η μέθοδος `backwardpass` ξεκινάει από το τελευταίο χρονικό βήμα και υπολογίζει τα τοπικά σφάλματα (δ) τόσο για τους κρυφούς όσο και για τους νευρώνες εξόδου. Με βάσει τ' αποτελέσματα της μεθόδου αυτής και με τη μέθοδο `updateWeights` αλλάζουν οι τρέχουσες τιμές των βαρών. Ο αλγόριθμος στην πραγματικότητα υλοποιείται από τη μέθοδο `execute` η οποία καλεί τις μεθόδους που περιγράφονται πιο πάνω. Τέλος, η μέθοδος `finalizing` περνά από το δίκτυο μια φορά τα δεδομένα επαλήθευσης και εκτυπώνει στο αρχείο την πραγματική και την προβλεπόμενη τιμή της ζήτησης.

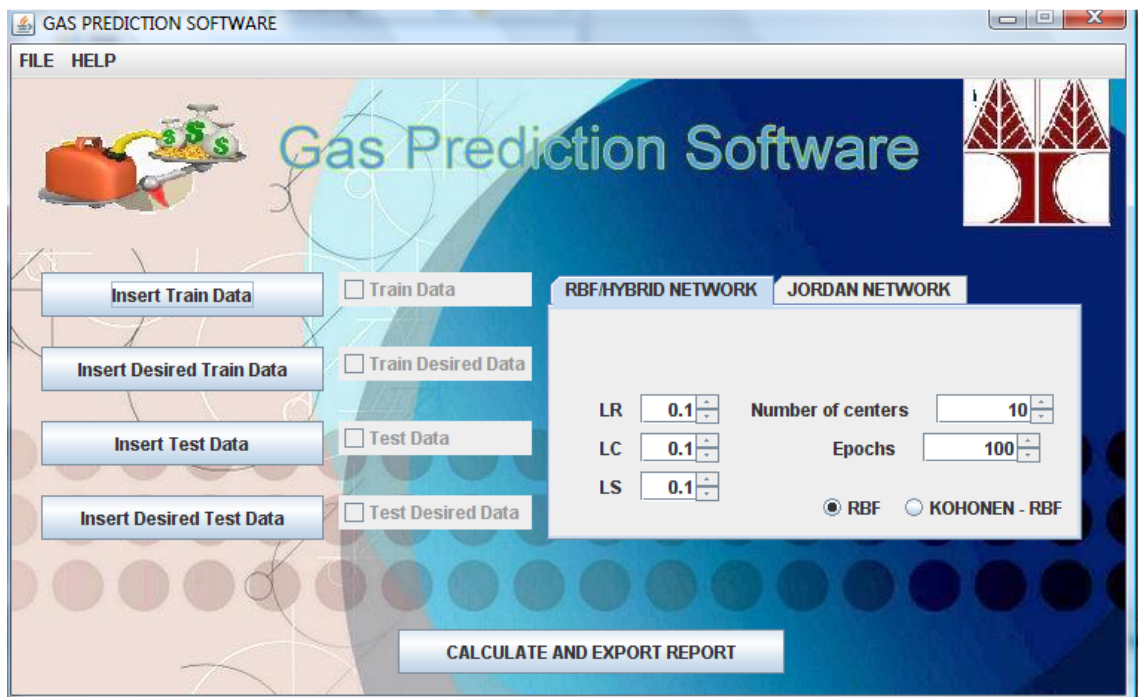
Κεφάλαιο 4

Επίδειξη Χρήσης Εργαλείου Εκτίμησης Ζήτησης

4.1 Γενική περιγραφή	54
4.2 Εισαγωγή δεδομένων εκπαίδευσης – ελέγχου	54
4.3 Αλγόριθμοι εκπαίδευσης	56
4.3.1 Αλγόριθμος μεταβλητών κέντρων σε δίκτυο RBF	56
4.3.2 Αλγόριθμος back propagation through time	58
4.4 Λοιπές Λειτουργίες	59

4.1 Γενική περιγραφή

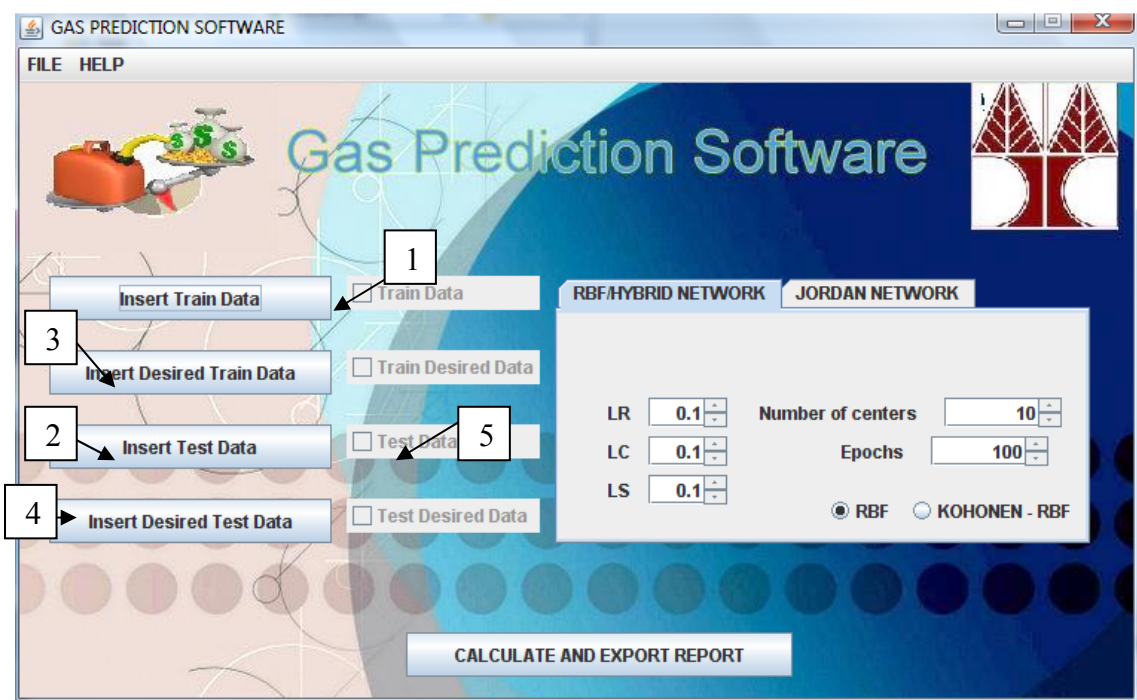
Στο κεφάλαιο αυτό θ' ασχοληθούμε με την αναλυτική περιγραφή του εργαλείου το οποίο σχεδιάστηκε και υλοποιήθηκε όπως περιγράφεται στο κεφάλαιο 3. Το εργαλείο αυτό (GPS –Gas Prediction Software) έχει φιλικό προς το χρήστη περιβάλλον καθώς επίσης είναι πολύ εύχρηστο αφού παρέχει τη δυνατότητα χρήσης του για οποιοδήποτε πρόβλημα το οποίο μπορεί να λυθεί είτε με δίκτυο RBF είτε με δίκτυο με ανάδραση τύπου Jordan. Στην εικόνα 4.1.1 βλέπουμε την κύρια οθόνη του προγράμματος.



Εικόνα 4.1.1 Κεντρική οθόνη GPS

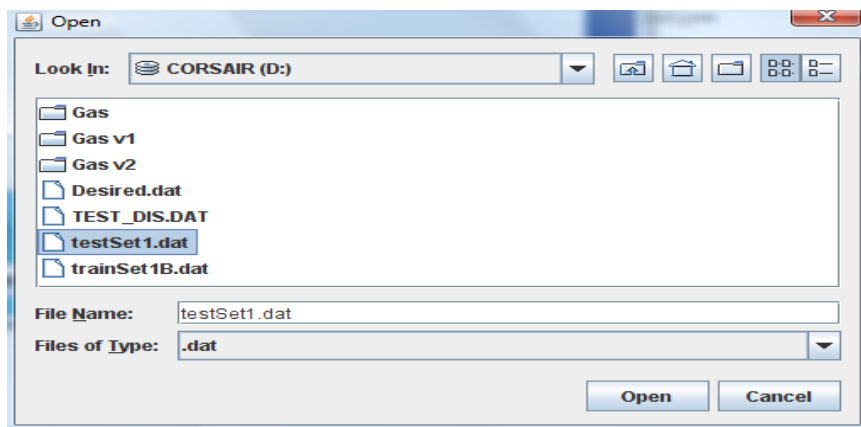
4.2 Εισαγωγή δεδομένων εκπαίδευσης - ελέγχου

Ο χρήστης μέσω του προγράμματος μπορεί να εισάγει δεδομένα για εκπαίδευση και έλεγχο. Αυτό γίνεται με το πάτημα του κουμπιού Insert Train Data και Insert Test Data (βέλη 1 και 2 στην εικόνα 4.2.1). Στη συνέχεια πρέπει να εισάγει τα επιθυμητά αποτελέσματα για τα δυο σύνολα δεδομένων πράγμα που γίνεται με το πάτημα των κουμπιών Insert Desired Train και Insert Desired Test βέλη 3 και 4 στην εικόνα 4.2.1).



Εικόνα 4.2.1 Εισαγωγή δεδομένων εκπαίδευσης - ελέγχου

Με το πάτημα των πιο πάνω πλήκτρων εμφανίζεται το παράθυρο πλοήγησης που φαίνεται στην εικόνα 4.2.2 μέσω του οποίου ο χρήστης θα πρέπει να βρει και να επιλέξει το αντίστοιχο αρχείο.



Εικόνα 4.2.2 Παράθυρο πλοήγησης για εισαγωγή δεδομένων εκπαίδευσης - ελέγχου

Αφού επιλεγεί το κατάλληλο αρχείο θα ενεργοποιηθεί το κουμπί ελέγχου που φαίνεται στο βέλος 5 στην εικόνα 4.2.1. Τα αρχεία εκπαίδευσης θα πρέπει να αποτελούνται από ένα πλήθος διανυσμάτων εισόδου κάθε ένα από τα οποία θα βρίσκεται σε διαφορετική γραμμή. Κάθε στοιχείο του διανύσματος θα αντιστοιχεί σε μια είσοδο. Όσον αφορά τα

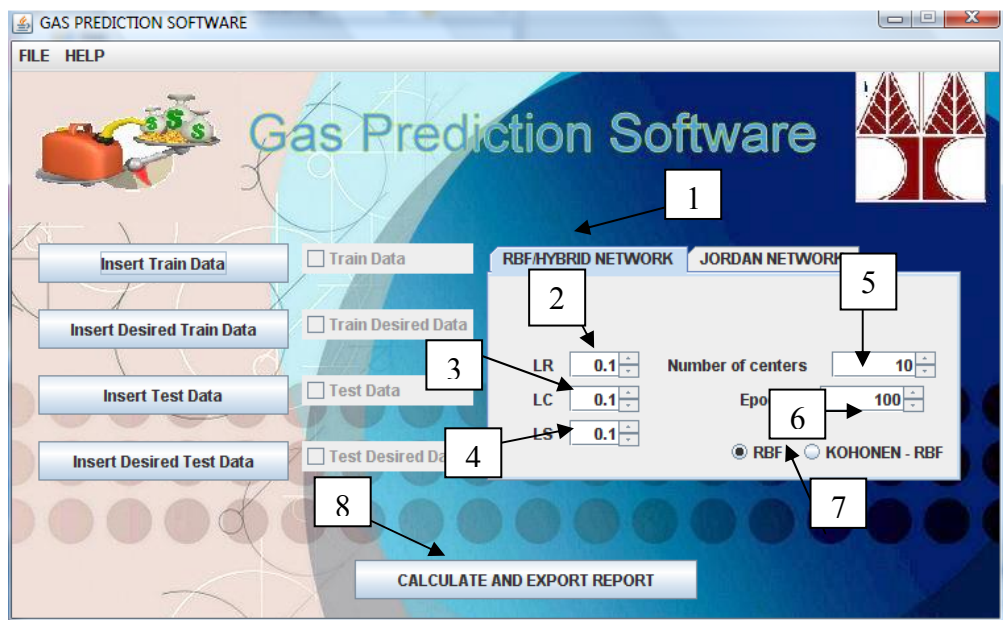
αρχεία με τα επιθυμητά αποτελέσματα θα πρέπει σε κάθε γραμμή να έχουμε το επιθυμητό αποτέλεσμα που αντιστοιχεί στην αντίστοιχη γραμμή του αρχείου εκπαίδευσης. Σε κάθε γραμμή θα έχουμε τόσες τιμές όσοι είναι και οι νευρώνες εξόδου. Σε περίπτωση που δεν ταυτίζεται ο αριθμός μεταξύ δεδομένων εισόδου και επιθυμητών αποτελεσμάτων τότε θα εμφανίζεται στο χρήστη αντίστοιχο μήνυμα λάθους.

4.3 Αλγόριθμοι εκπαίδευσης

Το σύστημα (GPS) παρέχει τη δυνατότητα εκπαίδευσης δύο κατηγοριών δικτύων. Συγκεκριμένα υποστηρίζει εκπαίδευση δικτύων RBF και SOM – RBF με τη χρήση του αλγορίθμου μεταβλητών κέντρων. Επίσης υποστηρίζει την εκπαίδευση δικτύου με ανάδραση τύπου Jordan με τη χρήση του αλγορίθμου back propagation through time.

4.3.1 Αλγόριθμός μεταβλητών κέντρων σε δίκτυο RBF και σε SOM – RBF

Για την εκπαίδευση αυτού του δικτύου θα πρέπει να επιλέξουμε το tab που φαίνεται με το βέλος 1 στην εικόνα 4.3.1.1 που ακολουθεί.



Εικόνα 4.3.1.1 Δίκτυο RBF και SOM - RBF

Στη συνέχεια ο χρήστης θα πρέπει να εισάγει τις διάφορες παραμέτρους (βέλη 2-6 στην εικόνα 4.3.1.1) για το συγκεκριμένο αλγόριθμο:

- **LR** – Learning rate for centres
- **LC** – Learning rate for coefficients
- **LS** – Learning rate for sigma
- **Number of centres** – Αυτή η παράμετρος αφορά μόνο το δίκτυο RBF
- **Epochs** – Number of iterations

Για την αλλαγή αυτών των τιμών θα πρέπει να γίνει κλικ στα up and down keys δίπλα σε κάθε παράμετρο. Για τη χρήση του υβριδικού δικτύου SOM – RBF θα πρέπει να επιλέγει το κουτί που φαίνεται με το βέλος 7 στην εικόνα 4.3.1.1.

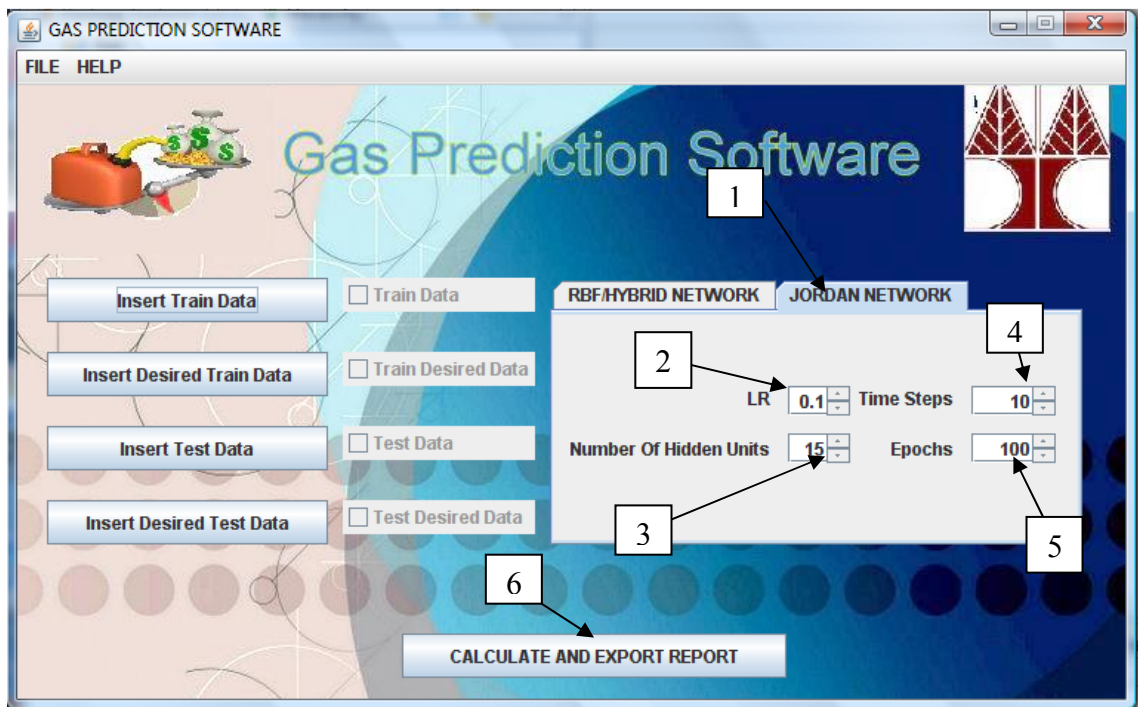
Τέλος για την εξαγωγή των αποτελεσμάτων σε αρχείο και την εμφάνιση του σχετικού report θα πρέπει να επιλέξει το κουμπί με το βέλος 8. Το αρχείο εξόδου θα έχει τη μορφή που περιγράφεται πιο κάτω:

EPOCH	TRAINING ERROR	TESTING ERROR	SUCCESS RATE %
1	0.5	0.6	55
2	0.41	0.55	58
3	0.40	0.53	63
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Εικόνα 4.3.1.2 Μορφή αρχείου εξόδου

4.3.2 Αλγόριθμος back propagation through time

Για την εκπαίδευση αυτού του δικτύου θα πρέπει να επιλέξουμε το tab που φαίνεται με το βέλος 1 στην εικόνα 4.3.2.1 που ακολουθεί.



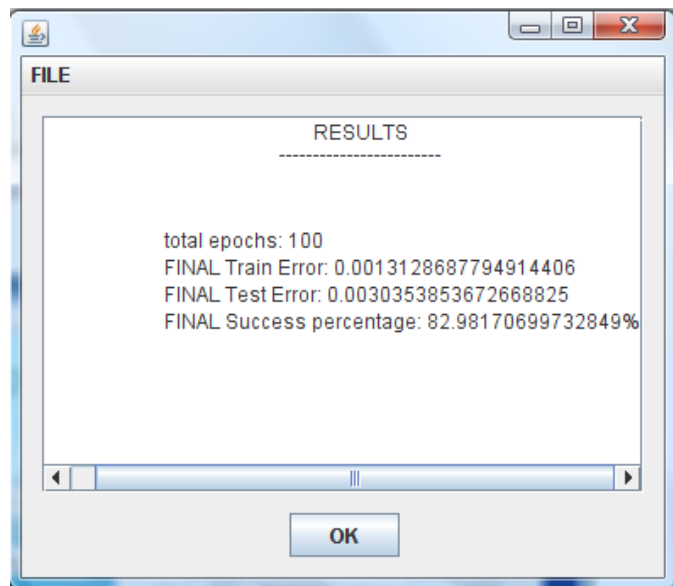
Εικόνα 4.3.2.1 Δίκτυο ανατροφοδότησης τύπου Jordan -Back Propagation Trough time-

Στη συνέχεια ο χρήστης θα πρέπει να εισάγει τις διάφορες παραμέτρους (βέλη 2-5 στην εικόνα 4.3.2.1) για το συγκεκριμένο αλγόριθμο:

- LR – Learning rate
- Number of hidden units – it supports up to 2 hidden layers -
- Number of time steps
- Epochs – Number of iterations

Για την αλλαγή αυτών των τιμών θα πρέπει να γίνει κλικ στα up and down keys δίπλα σε κάθε παράμετρο.

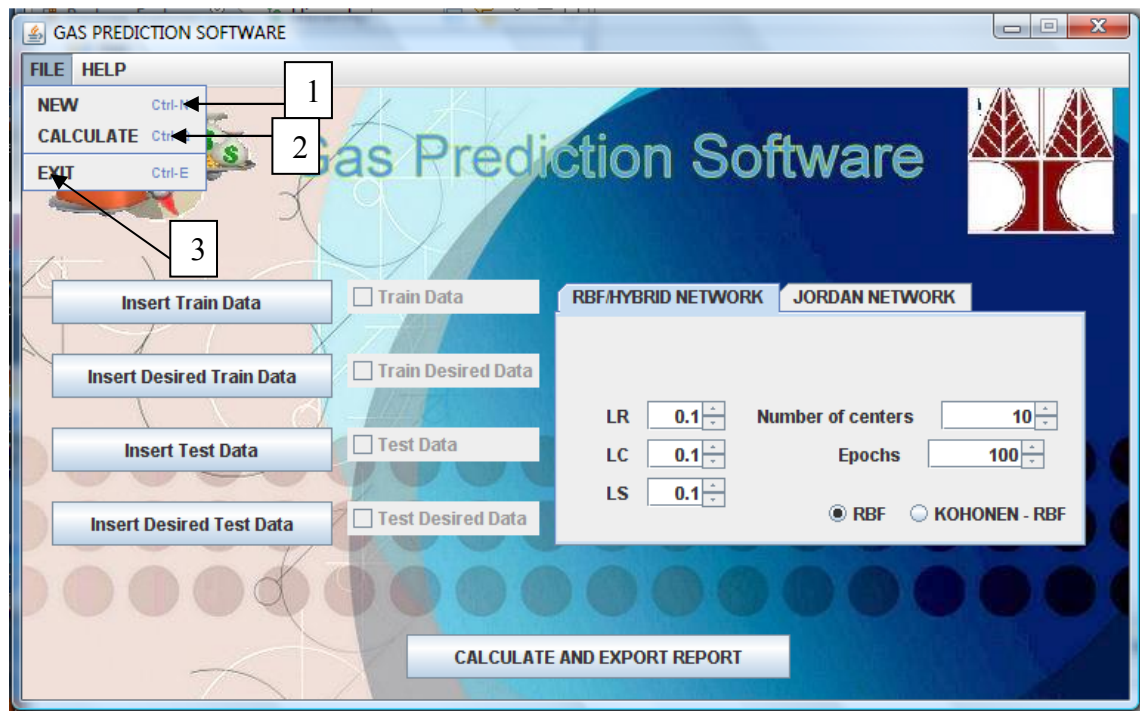
Τέλος για την εξαγωγή των αποτελεσμάτων σε αρχείο και την εμφάνιση του σχετικού report θα πρέπει να επιλέξει το κουμπί με το βέλος 6. Το αρχείο εξόδου θα έχει τη μορφή που φαίνεται στην εικόνα 4.3.1.2. Στην εικόνα 4.3.2.2 που ακολουθεί παρουσιάζεται το παράθυρο αποτελεσμάτων.



Εικόνα 4.3.2.2 Παράθυρο αποτελεσμάτων

4.4 Λοιπές Λειτουργίες

Τέλος το πρόγραμμα θα έχει και κάποιες επιπλέον λειτουργίες οι οποίες φαίνονται και επεξηγούνται στην παρακάτω εικόνα



Εικόνα 4.4.1 Λοιπές Λειτουργίες Προγράμματος

Λειτουργία New – βέλος 1

Μ' αυτή τη λειτουργία ο χρήστης μπορεί να κάνει reset το πρόγραμμα.

Λειτουργία Calculate – βέλος 2

Μ' αυτή τη λειτουργία ο χρήστης μπορεί τρέξει τον επιλεγόμενο αλγόριθμο.

Λειτουργία Exit – βέλος 2

Μ' αυτή τη λειτουργία ο χρήστης κλείνει το πρόγραμμα.

Με την επιλογή του tab help ο χρήστης θα έχει τη δυνατότητα να δει τον οδηγό χρήσης καθώς επίσης και σχετικές πληροφορίες για το GPS.

Κεφάλαιο 5

Πειράματα - Αποτελέσματα

5.1 Εισαγωγή	62
5.2 Αλλαγές στις παραμέτρους του δικτύου RBF	63
5.2.1 Αλλαγές στους ρυθμούς μάθησης	63
5.2.2 Αλλαγές στον αριθμό των κέντρων	68
5.3 Αλλαγές στις παραμέτρους του υβριδικού δικτύου SOM-RBF	73
5.3.1 Αλλαγές στους ρυθμούς μάθησης	74
5.4 Αλλαγές στα σύνολα δεδομένων εκπαίδευσης και επαλήθευσης	79
5.5 Αλλαγές στις διαστάσεις του πλέγματος στον αλγόριθμο Kohonen SOM	86
5.6 Αλλαγές στις παραμέτρους του δικτύου με ανάδραση-Back Propagation Through Time	88
5.6.1 Αλλαγές στο ρυθμό μάθησης	89
5.6.2 Αλλαγές στον αριθμό των κρυφών νευρώνων	93
5.6.3 Αλλαγές στον αριθμό των χρονικών βημάτων	98
5.6.4 Αλλαγή των τιμών της χρονικής σταθεράς (time constant)	102
5.6.5 Τροποποίηση στον τρόπο υπολογισμού της επόμενης τιμής της ζήτησης	104
5.7 Σύγκριση επιθυμητού - πραγματικού αποτελέσματος	106
5.8 Χρήση του Μέσου Απόλυτου Σχετικού Σφάλματος (MARE) για τον υπολογισμό του σφάλματος	118

5.1 Εισαγωγή

Στο κεφάλαιο αυτό παραθέτουμε τα πειράματα μας καθώς και τ' αποτελέσματα που προέκυψαν από την εκπαίδευση των τριών δικτύων. Πιο συγκεκριμένα θα παρατηρήσουμε τις αλλαγές που προέκυψαν με την αλλαγή δομικών παραμέτρων όπως ο ρυθμός μάθησης και το πλήθος των κέντρων στα δύο δίκτυα RBF καθώς επίσης το ρυθμό μάθησης, τον αριθμό των κρυφών νευρώνων και τον αριθμό των χρονικών βημάτων στο δίκτυο με ανατροφοδότηση.

Όπως αναφέραμε και στο κεφάλαιο 2 στην προεπεξεργασία των δεδομένων, τα δεδομένα καλύπτουν ένα χρονικό περιθώριο 7 ετών. Συγκεκριμένα ξεκινούν από τις 2 του Μάη του 1990, μέχρι τις 30 του Σεπτεμβρίου του 1997. Επομένως για το σκοπό αυτής της διπλωματικής εργασίας έγινε διαχωρισμός τους σε δεδομένα εκπαίδευσης και ελέγχου. Έτσι τα δεδομένα από τις 03/05/1992 μέχρι 30/09/1997 χρησιμοποιήθηκαν για εκπαίδευση ενώ τα δεδομένα από τις 02/05/1990 μέχρι 02/05/1992 για έλεγχο των δικτύων RBF και Kohonen-Som - RBF. Αντίθετα, στο δίκτυο με ανατροφοδότηση τύπου Jordan [11] χρησιμοποιήσαμε τα δεδομένα της ζήτησης ως χρονοσειρά τόσο για έλεγχο όσο και για εκπαίδευση του δικτύου και επιπλέον το πλήθος των δεδομένων εισόδου καθορίζεται από την τιμή της παραμέτρου των χρονικών βημάτων, για παράδειγμα για δέκα χρονικά βήματα τα δεδομένα εισόδου ήταν κατά δεκάδες χωρισμένα και η πρόβλεψη αντιστοιχούσε στην αμέσως επόμενη τιμή ζήτησης για τη κάθε δεκάδα. Η αλλαγή αυτή οφείλεται στο ότι ασχολούμαστε με χρονοσειρές και τα δεδομένα πρέπει να είναι χρονικά τοποθετημένα. Επιπλέον για την εξαγωγή των τιμών των σφαλμάτων της εκπαίδευσης και επαλήθευσης των δικτύων μας χρησιμοποιήσαμε το **μέσο τετραγωνικό λάθος** (mean square error MSE – κεφάλαιο 2).

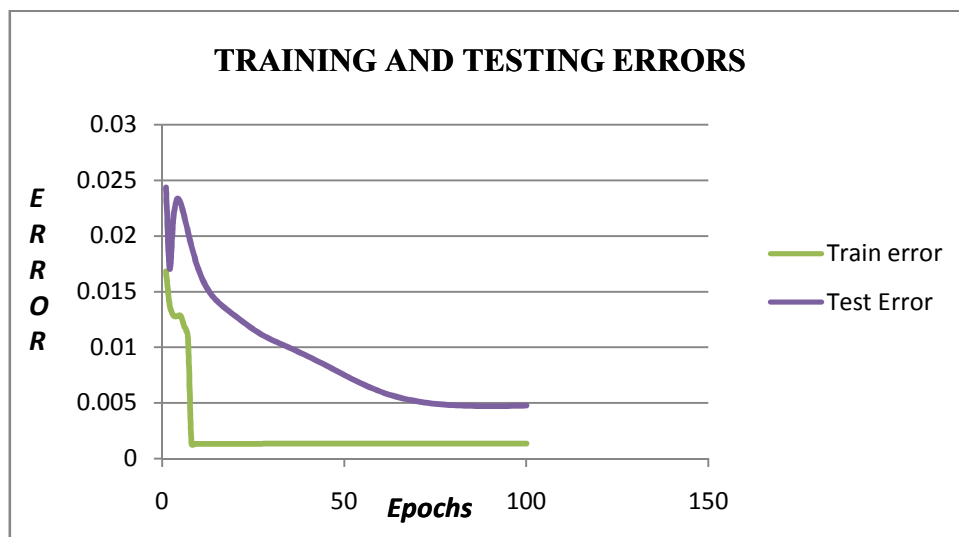
Τέλος, αξίζει να σημειωθεί ότι τόσο η δομή των δικτύων όσο και οι αλγόριθμοι εκπαίδευσης έχουν αυξημένη χρονική πολυπλοκότητα με αποτέλεσμα η εκπαίδευση των δικτύων να γίνεται μια χρονοβόρα και πολύπλοκη διαδικασία. Στο κεφάλαιο αυτό θα παρουσιάσουμε επιλεκτικά ένα υποσύνολο αποτελεσμάτων άλλα με θετική και άλλα με αρνητική σημασία.

5.2 Αλλαγές στις παραμέτρους του δικτύου RBF

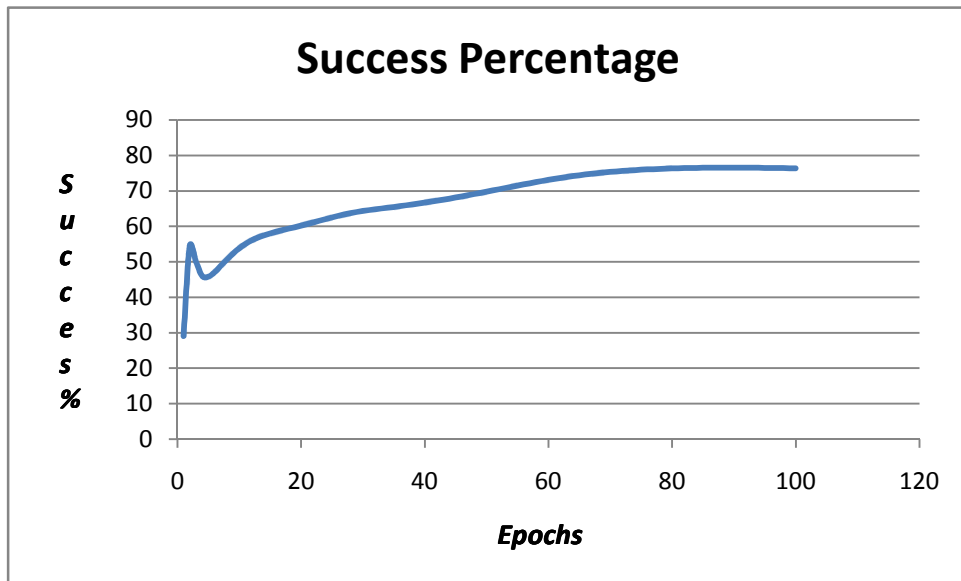
Σ' αυτή την υποενότητα θα επικεντρωθούμε στο δίκτυο RBF εκπαιδευμένο με τον αλγόριθμο με μεταβλητά κέντρα. Όπως αναφέραμε και στο σχεδιασμό σ' αυτό το δίκτυο η αρχική θέση των κέντρων καθορίζεται τυχαία μέσα από τα δεδομένα εκπαίδευσης. Στις υποενότητες που ακολουθούν θα παρατηρήσουμε την αλλαγή στη συμπεριφορά του δικτύου όταν αλλάζουμε το ρυθμό μάθησης (5.2.1) και το πλήθος των κέντρων (5.2.2).

5.2.1 Αλλαγή στους ρυθμούς μάθησης

Στην προσπάθεια μας να βρούμε την τιμή του ρυθμού μάθησης ο οποίος μας δίνει το μεγαλύτερο ποσοστό επιτυχίας κρατήσαμε σταθερά τον αριθμό των επαναλήψεων και τον αριθμό των κέντρων. Πιο συγκεκριμένα είχαμε 15 κέντρα και εκπαιδεύσαμε το δίκτυο για 100 εποχές. Στη συνέχεια εκπαιδεύσαμε το δίκτυο με 3 διαφορετικούς ρυθμούς μάθησης (0.1, 0.3 και 0.5) οι οποίοι ήταν κοινοί και στα κέντρα και στις διασπορές καθώς επίσης και στα βάρη. Τα αποτελέσματα που προέκυψαν παρουσιάζονται στα γραφήματα 5.2.1.1 – 5.2.1.6 που ακολουθούν.



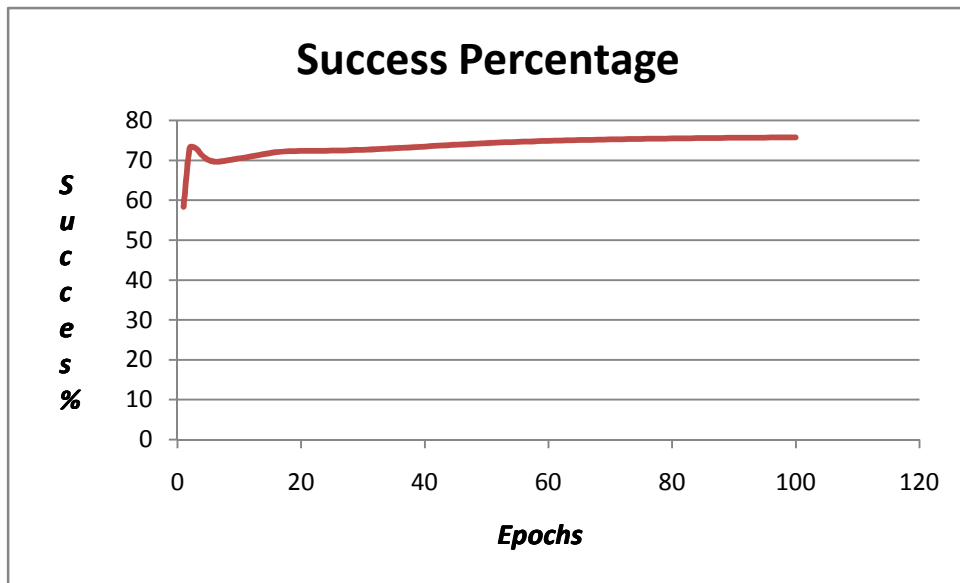
Γράφημα 5.2.1.1 Λάθη ελέγχου και επαλήθευσης στο δίκτυο RBF για ρυθμό μάθησης 0.1



Γράφημα 5.2.1.2 Ποσοστό επιτυχίας στο δίκτυο RBF για ρυθμό μάθησης 0.1



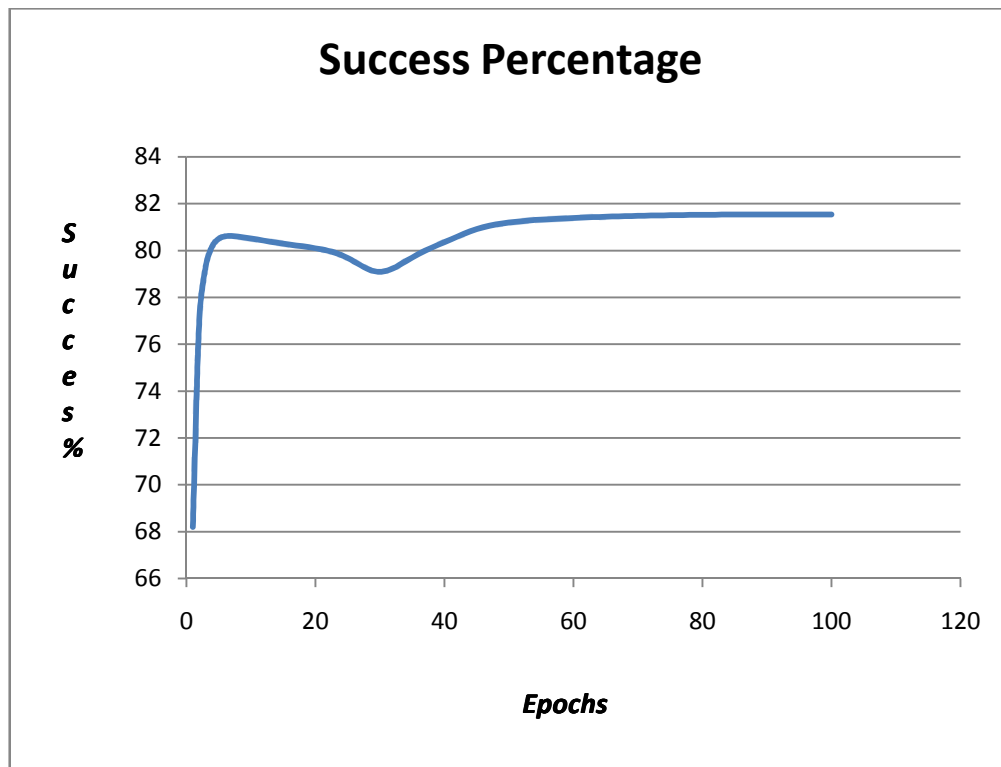
Γράφημα 5.2.1.3 Λάθη ελέγχου και επαλήθευσης στο δίκτυο RBF για ρυθμό μάθησης 0.3



Γράφημα 5.2.1.4 Ποσοστό επιτυχίας στο δίκτυο RBF για ρυθμό μάθησης 0.3

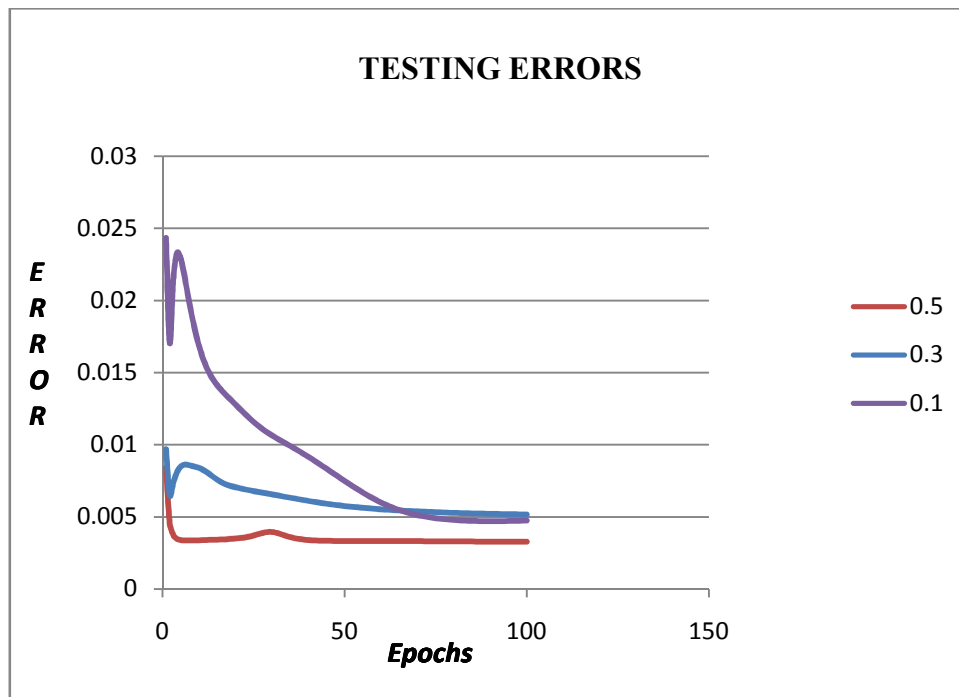


Γράφημα 5.2.1.5 Λάθη ελέγχου και επαλήθευσης στο δίκτυο RBF για ρυθμό μάθησης 0.5



Γράφημα 5.2.1.6 Ποσοστό επιτυχίας στο δίκτυο RBF για ρυθμό μάθησης 0.5

Στις πιο πάνω γραφικές παραστάσεις παρατηρούμε ότι τόσο το σφάλμα εκπαίδευσης όσο και το σφάλμα επαλήθευσης μειώνονται με την πάροδο του χρόνου σε αντίθεση με το ποσοστό επιτυχίας πρόβλεψης της ζήτησης του φυσικού αερίου το οποίο αυξάνεται. Επομένως στις επιτυχείς εκπαιδεύσεις το λάθος τείνει να μειώνεται όσο περνά ο χρόνος και το δίκτυο μαθαίνει με αποτέλεσμα τα ποσοστά επιτυχίας του στις προβλέψεις να αυξάνονται. Στη συνέχεια παραθέτουμε το συγκριτικό γράφημα (5.2.1.7) μεταξύ των σφαλμάτων επαλήθευσης για τους τρεις ρυθμούς μάθησης.

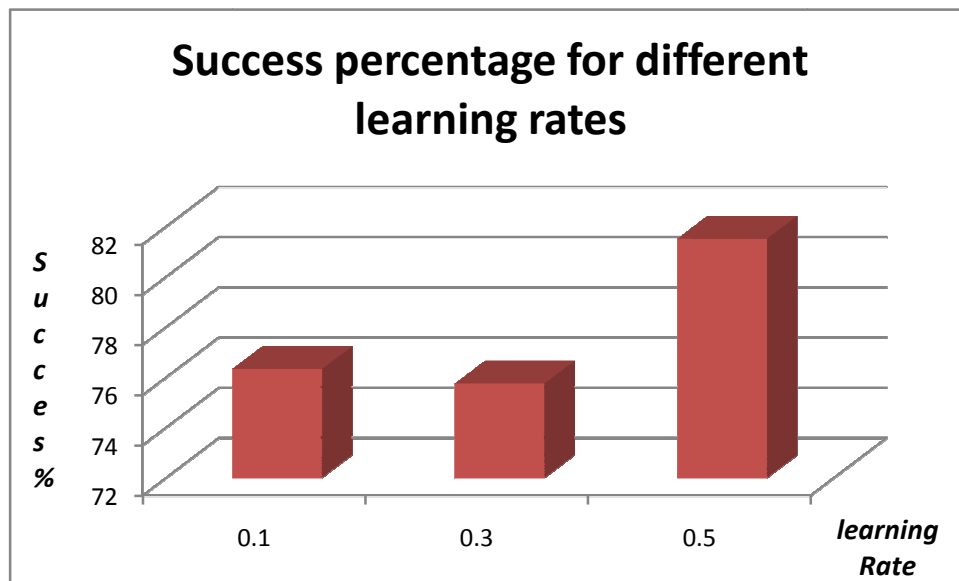


Γράφημα 5.2.1.7 Σφάλματα επαλήθευσης για ρυθμούς μάθησης 0.1,0.3 και 0.5

Στην παραπάνω γραφική παράσταση παρατηρούμε ότι η μικρότερη τιμή του σφάλματος παρουσιάζεται όταν ο ρυθμός μάθησης είναι 0.5. Γι' αυτό το λόγο παρατηρείται το μεγαλύτερο ποσοστό επιτυχίας μ' αυτήν την τιμή του ρυθμού μάθησης. Στον πίνακα που ακολουθεί παρουσιάζονται τα ποσοστά επιτυχίας στην τελευταία εποχή για κάθε ένα από τους τρεις ρυθμούς μάθησης.

<i>Ρυθμός Μάθησης</i>	0.1	0.3	0.5
<i>Ποσοστό επιτυχίας</i>	76.34%	75.76%	81.53%

Πίνακας 5.2.1.1 Ποσοστά επιτυχίας (%) στα δεδομένα επαλήθευσης στο δίκτυο RBF για ρυθμούς μάθησης 0.1, 0.3 και 0.5

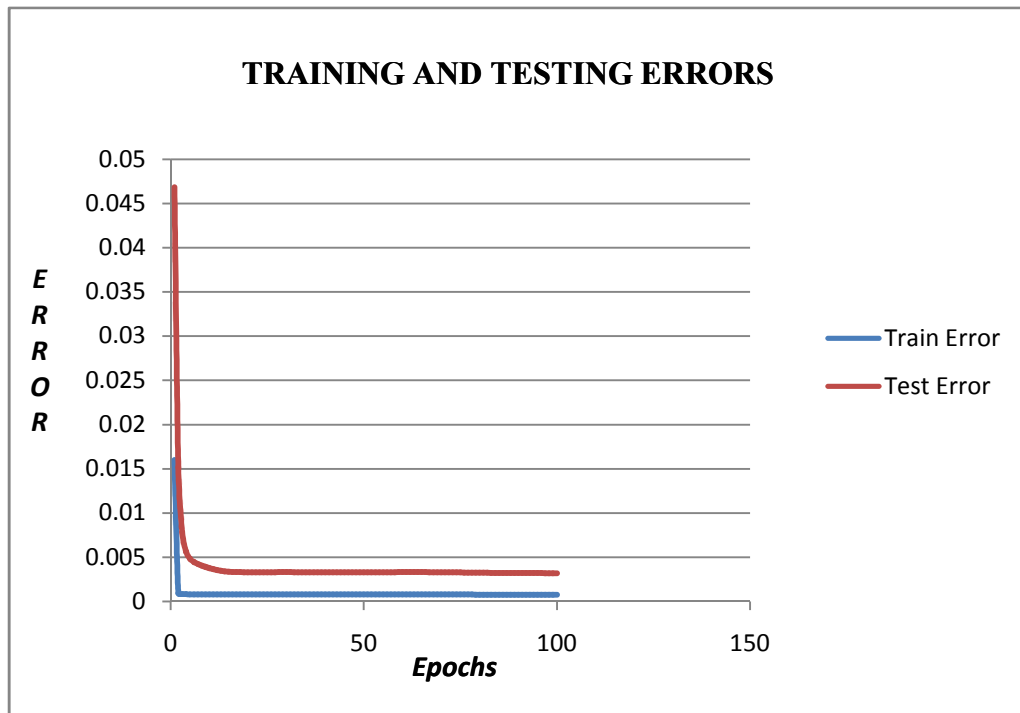


Γράφημα 5.2.1.8 Τελικά ποσοστά επιτυχίας στο δίκτυο RBF για ρυθμούς μάθησης 0.1,0.3 και 0.5

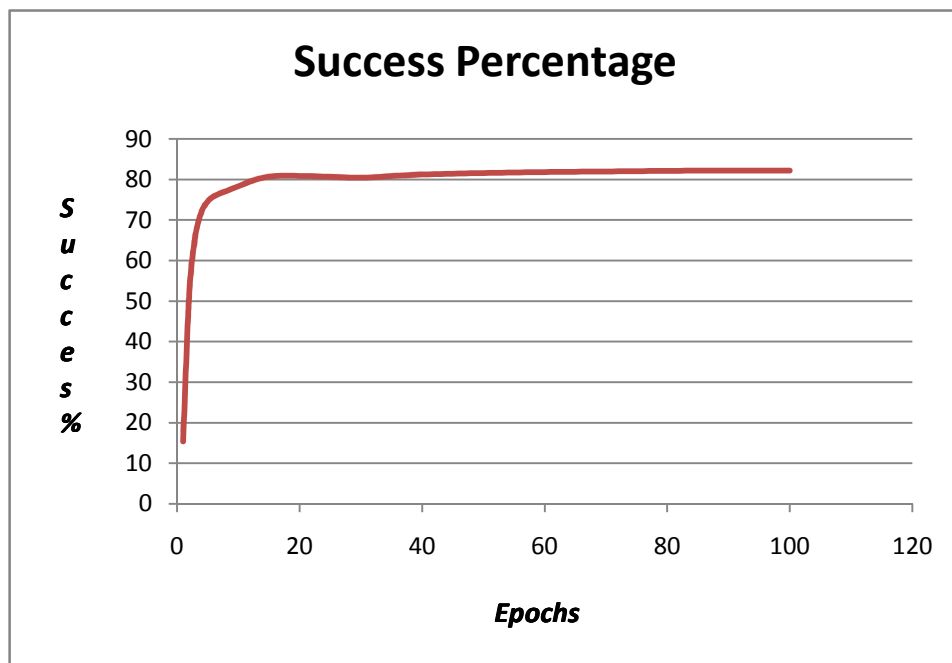
Από το πιο πάνω γράφημα (5.2.1.8) φαίνεται ότι το μεγαλύτερο ποσοστό επιτυχίας παρουσιάζεται με ρυθμό μάθησης 0.5 και 15 κέντρα και φτάνει το 81.53%. Στη συνέχεια θα προσπαθήσουμε να βελτιώσουμε το ποσοστό αυτό διατηρώντας σταθερό το ρυθμό μάθησης εκπαιδεύοντας το δίκτυο με διαφορετικό πλήθος κέντρων.

5.2.2 Αλλαγή στον αριθμό των κέντρων

Αφού αποφανθήκαμε ότι με ρυθμό μάθησης 0.5 έχουμε το μεγαλύτερο ποσοστό επιτυχίας προσπαθήσαμε να το βελτιώσουμε κρατώντας σταθερό το ρυθμό μάθησης σε 0.5 εκπαιδεύοντας το δίκτυο με διαφορετικό αριθμό κέντρων. Συγκεκριμένα εκπαιδεύσαμε το δίκτυο για 100 εποχές τρεις φορές. Στην πρώτη εκπαίδευση χρησιμοποιήσαμε 10 κέντρα, στη δεύτερη 20 και στην τρίτη 25. Τα αποτελέσματα που προέκυψαν παρουσιάζονται στα γραφήματα 5.2.2.1 – 5.2.2.6 που ακολουθούν.



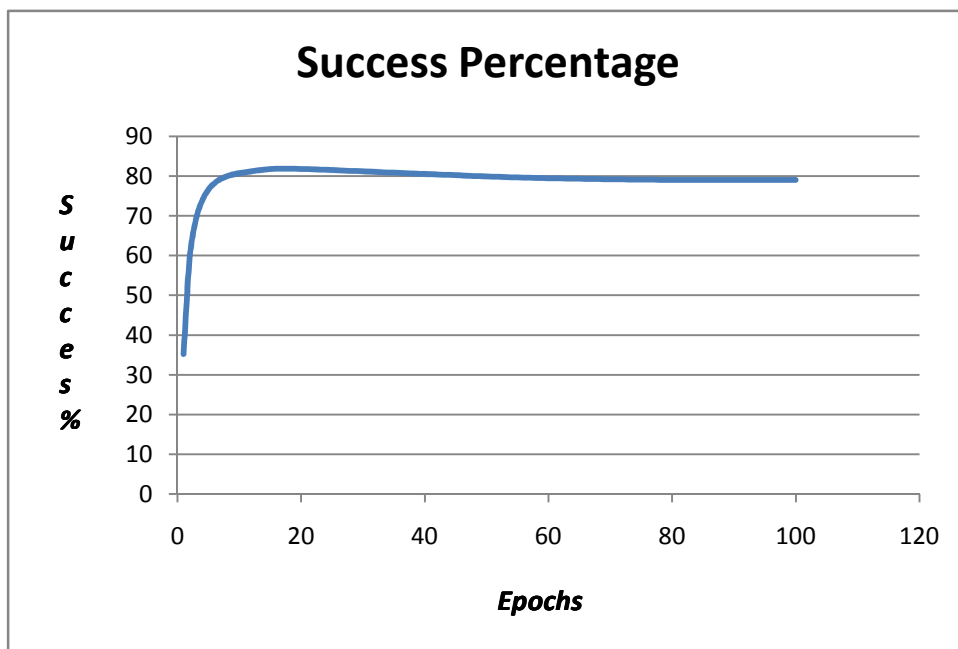
Γράφημα 5.2.2.1 Λάθη ελέγχου και επαλήθευσης στο δίκτυο RBF με 10 κέντρα



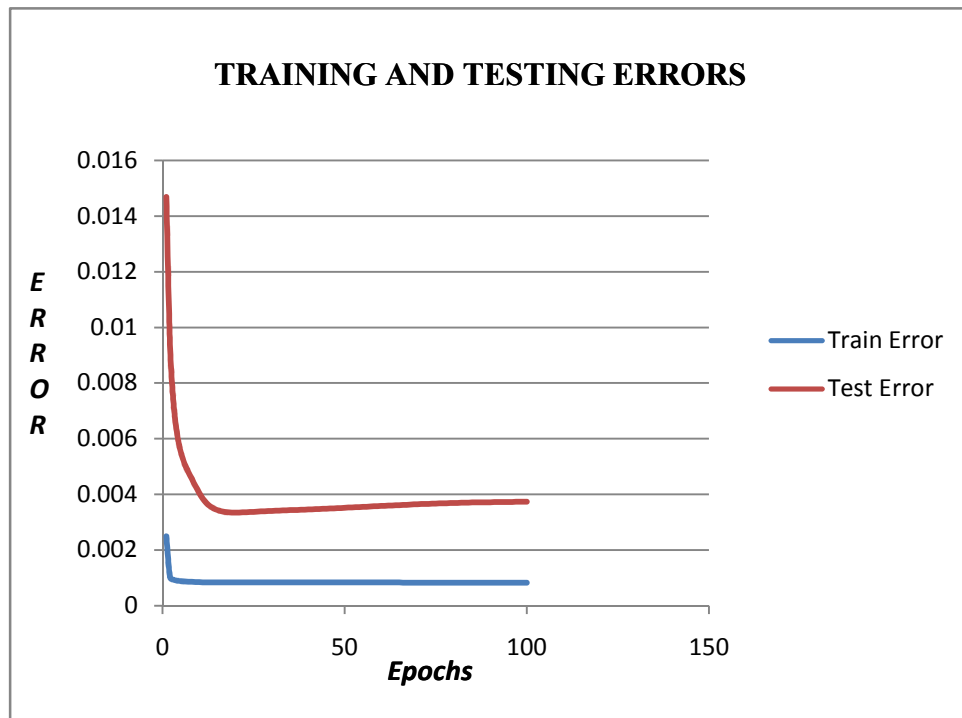
Γράφημα 5.2.2.2 Ποσοστό επιτυχίας στο δίκτυο RBF με 10 κέντρα



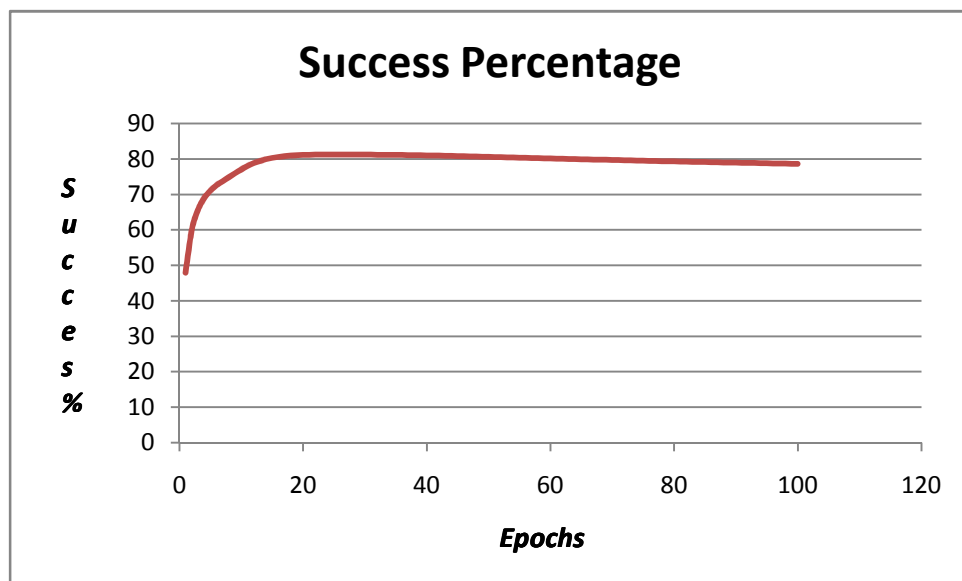
Γράφημα 5.2.2.3 Λάθη ελέγχου και επαλήθευσης στο δίκτυο RBF με 20 κέντρα



Γράφημα 5.2.2.4 Ποσοστό επιτυχίας στο δίκτυο RBF με 20 κέντρα

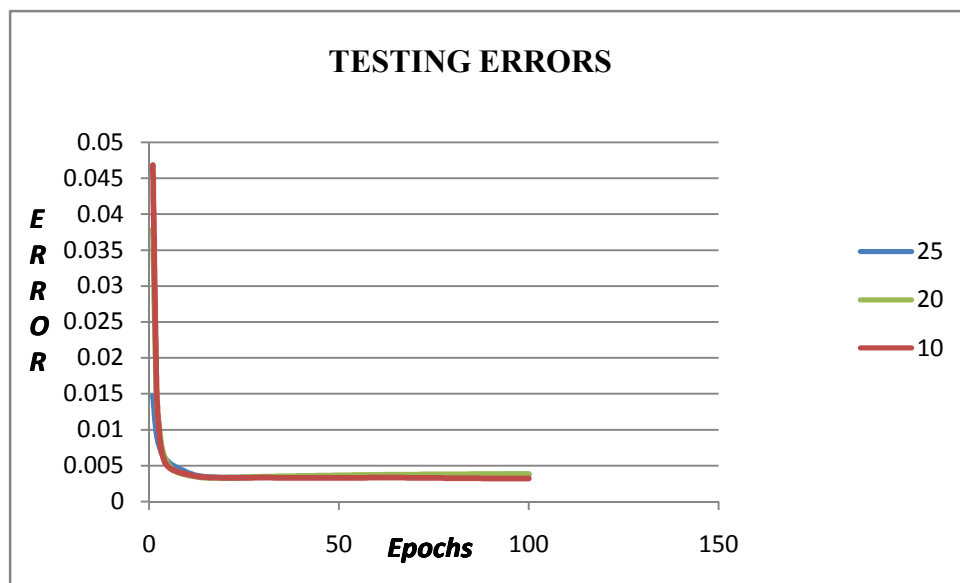


Γράφημα 5.2.2.5 Λάθη ελέγχου και επαλήθευσης στο δίκτυο RBF με 25 κέντρα



Γράφημα 5.2.2.6 Ποσοστό επιτυχίας στο δίκτυο RBF με 25 κέντρα

Στις πιο πάνω γραφικές παραστάσεις όπως και στις αντίστοιχες με ρυθμούς μάθησης παρατηρούμε ότι τόσο το σφάλμα εκπαίδευσης όσο και το σφάλμα επαλήθευσης μειώνονται με την πάροδο του χρόνου σε αντίθεση με το ποσοστό επιτυχίας πρόβλεψης της ζήτησης του φυσικού αερίου το οποίο αυξάνεται. Επομένως στις επιτυχείς εκπαιδεύσεις το λάθος τείνει να μειώνεται όσο περνά ο χρόνος και το δίκτυο μαθαίνει με αποτέλεσμα τα ποσοστά επιτυχίας του στις προβλέψεις να αυξάνονται. Στη συνέχεια παραθέτουμε το συγκριτικό γράφημα μεταξύ των σφαλμάτων επαλήθευσης.

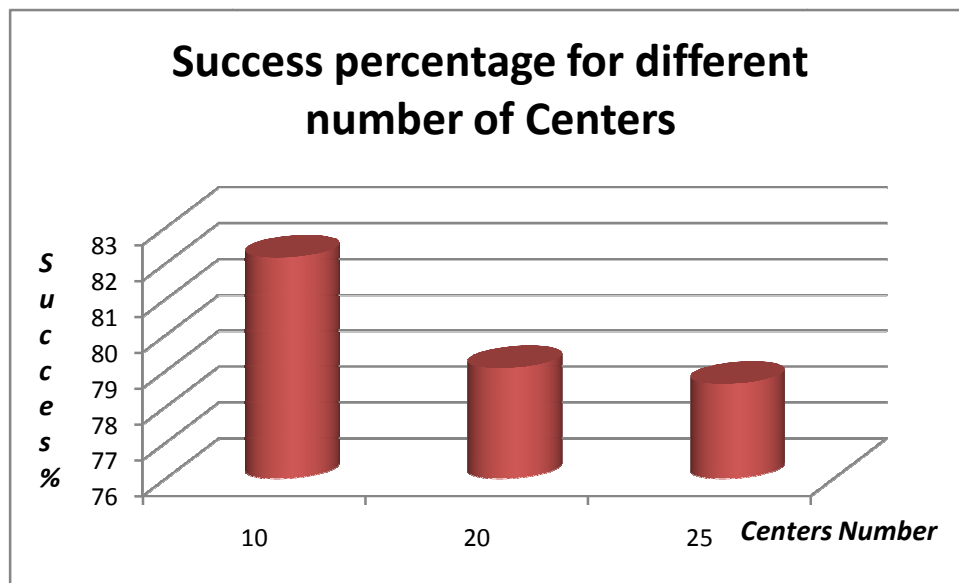


Γράφημα 5.2.2.7 Σφάλματα επαλήθευσης στο δίκτυο RBF με 10,20 και 25 κέντρα

Στην γραφική παράσταση 5.2.2.7 παρατηρούμε ότι τα σφάλματα επαλήθευσης για 20 , 25 και 10 κέντρα είναι πολύ κοντά πράγμα που δεν μας επιτρέπει να πούμε πιο από τα τρία μας δίνει το καλύτερο αποτέλεσμα. Όμως παρατηρώντας τον πίνακα 5.2.2.1 καθώς και το γράφημα 5.2.2.8 βλέπουμε ότι με 10 κέντρα έχουμε το μεγαλύτερο ποσοστό επιτυχίας στην τελευταία εποχή.

<i>Αριθμός Κέντρων</i>	10	20	25
<i>Ποσοστό επιτυχίας</i>	82.152%	79.07%	78.63%

Πίνακας 5.2.2.1 Ποσοστά επιτυχίας (%) στα δεδομένα επαλήθευσης στο δίκτυο RBF με 10, 20 και 25 κέντρα



Γράφημα 5.2.2.8 Τελικά ποσοστά επιτυχίας στο δίκτυο RBF με 10,20 και 25 κέντρα

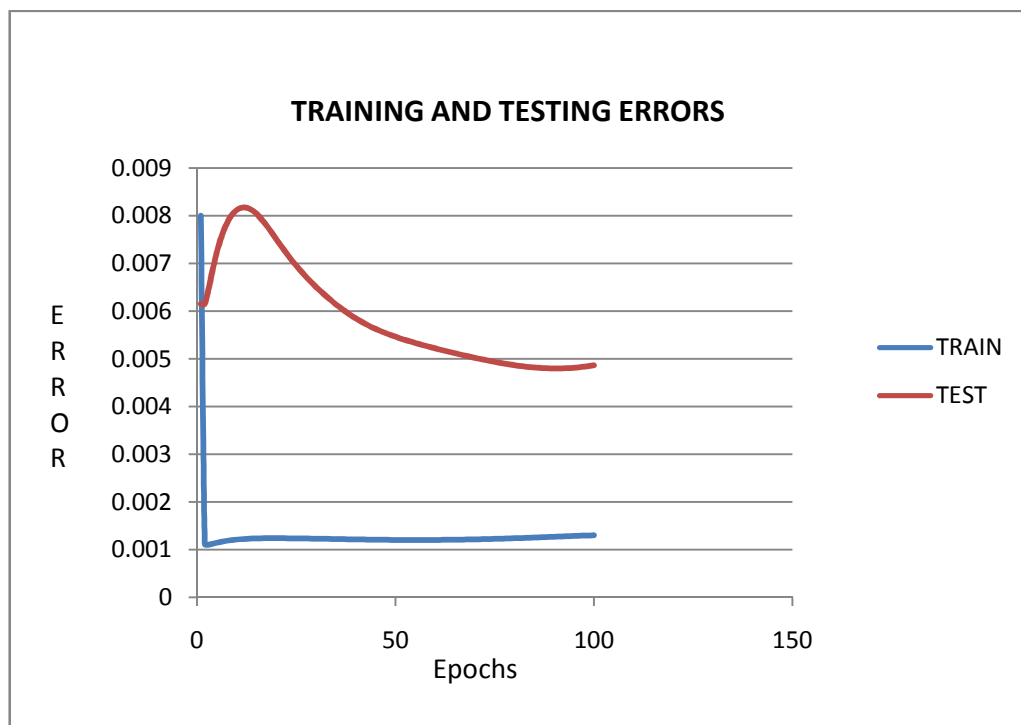
Από το πιο πάνω γράφημα φαίνεται ότι το μεγαλύτερο ποσοστό επιτυχίας παρουσιάζεται με ρυθμό μάθησης 0.5 και 10 κέντρα και φτάνει το 82.152%. Το ποσοστό αυτό είναι υψηλότερο από το προηγούμενο (με ρυθμό μάθησης 0.5 και 15 κέντρα) το οποίο ανέρχεται στο 81.53%. Στη συνέχεια θα προσπαθήσουμε να βελτιώσουμε το ποσοστό κάνοντας τα ίδια πειράματα σε ένα υβριδικό δίκτυο SOM-RBF.

5.3 Αλλαγές στις παραμέτρους του υβριδικού δικτύου Kohonen SOM-RBF

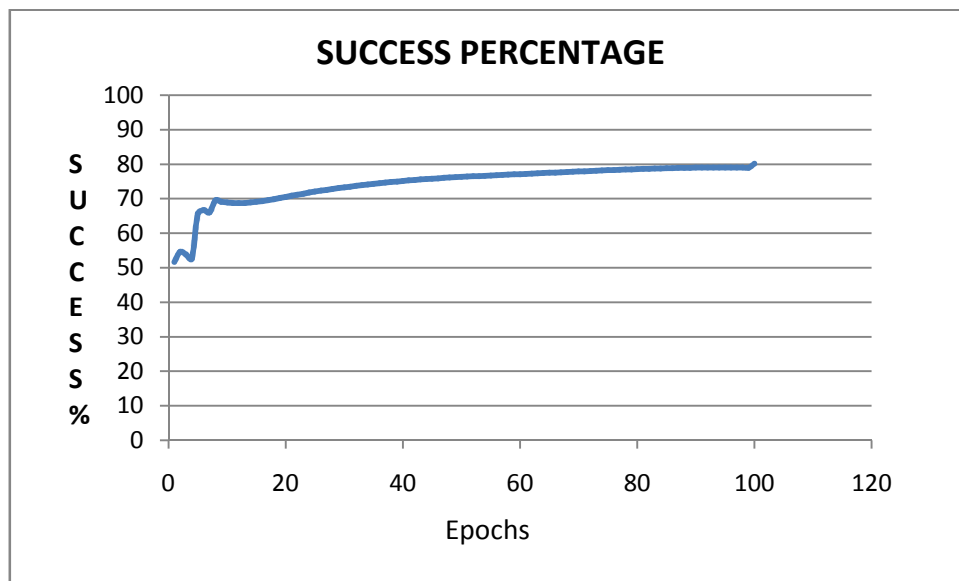
Σ' αυτή την υποενότητα θα επικεντρωθούμε στο δίκτυο RBF εκπαιδευμένο με τον αλγόριθμο με μεταβλητά κέντρα. Όπως αναφέραμε και στο σχεδιασμό σ' αυτό το δίκτυο η **αρχική θέση και το πλήθος των κέντρων** καθορίζεται με τον αλγόριθμο ομαδοποίησης Kohonen SOM [11] σε αντίθεση με το δίκτυο RBF του 5.2 όπου γίνεται τυχαία η αρχικοποίηση της θέσης των κέντρων και ο αριθμός τους καθορίζεται από το χρήστη. Στις υποενότητες που ακολουθούν θα παρατηρήσουμε την αλλαγή στη συμπεριφορά του υβριδικού αυτού δικτύου όταν αλλάζουμε το ρυθμό μάθησης (5.3.1).

5.3.1 Αλλαγή στους ρυθμούς μάθησης

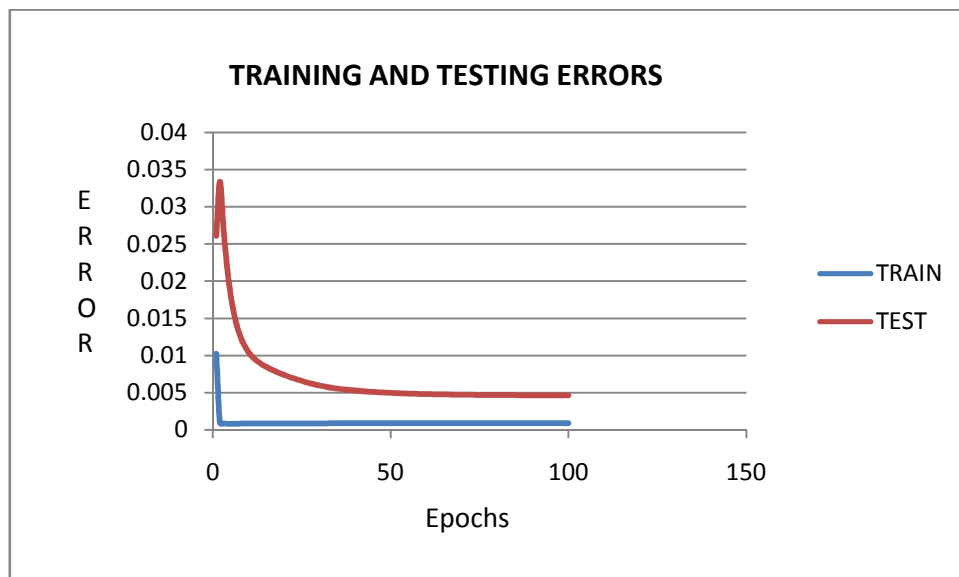
Στην προσπάθεια μας να βρούμε την τιμή του ρυθμού μάθησης ο οποίος μας δίνει το μεγαλύτερο ποσοστό επιτυχίας κρατήσαμε σταθερά τον αριθμό των επαναλήψεων και ο αριθμός των κέντρων καθοριζόταν από τον αλγόριθμο ομαδοποίησης kohonen SOM [11], του οποίου το μέγεθος του πλέγματος –ενότητα 2.1.2- αρχικοποιήθηκε με διαστάσεις 20x20. Στη συνέχεια εκπαιδεύσαμε το δίκτυο με 3 διαφορετικούς ρυθμούς μάθησης (0.1, 0.3 και 0.5) οι οποίοι ήταν οι ίδιοι και στα κέντρα και στις διασπορές καθώς επίσης και στα βάρη. Το πλήθος των κέντρων όπως αυτό καθορίστηκε από τον αλγόριθμο ομαδοποίησης Kohonen SOM ήταν 37, 33 και 35 για του τρεις ρυθμούς μάθησης αντίστοιχα. Τα αποτελέσματα που προέκυψαν παρουσιάζονται στα γραφήματα 5.3.1.1 – 5.3.1.6 που ακολουθούν.



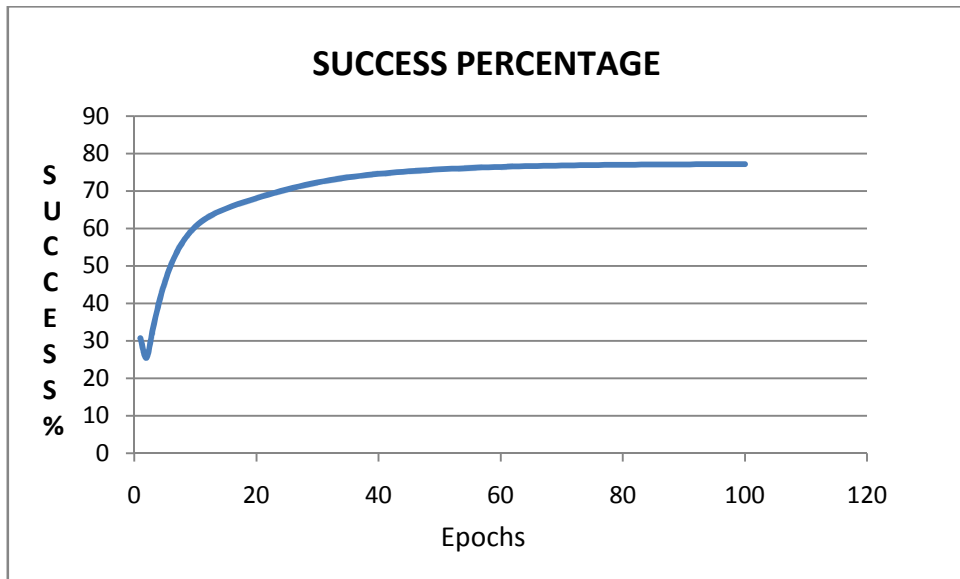
Γράφημα 5.3.1.1 Λάθη ελέγχου και επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμό μάθησης 0.1



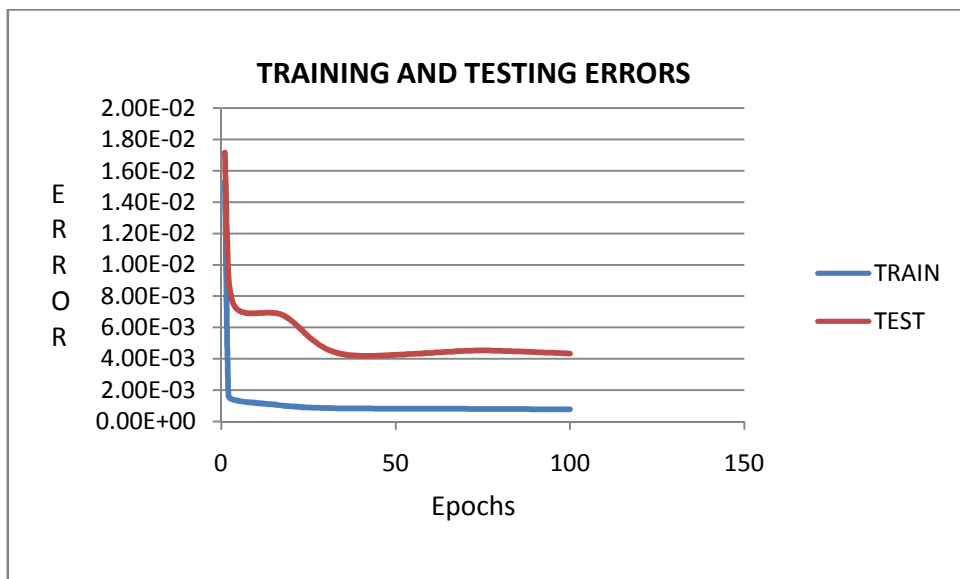
Γράφημα 5.3.1.2 Ποσοστό επιτυχίας στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμό μάθησης 0.1



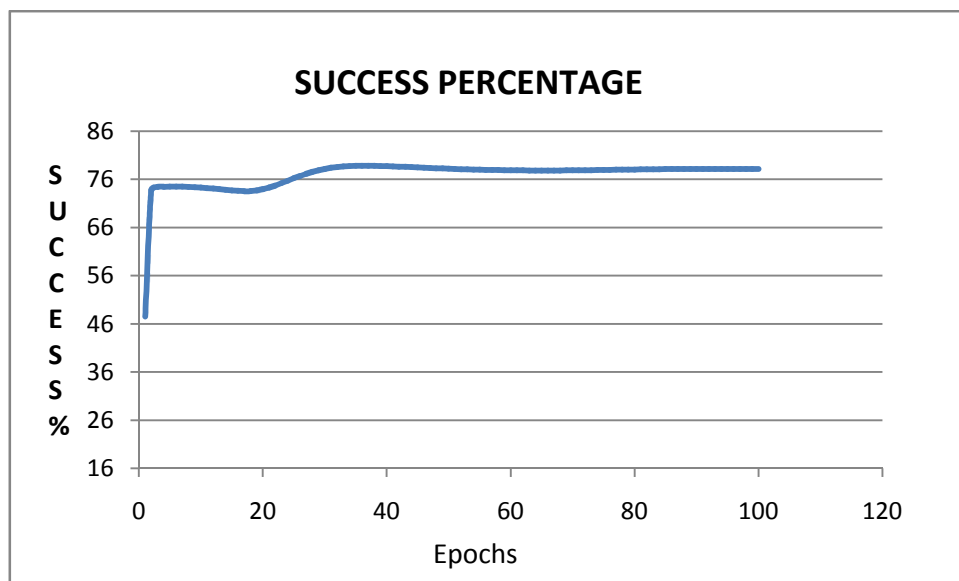
Γράφημα 5.3.1.3 Λάθη ελέγχου και επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμό μάθησης 0.3



Γράφημα 5.3.1.4 Ποσοστό επιτυχίας στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμό μάθησης 0.3



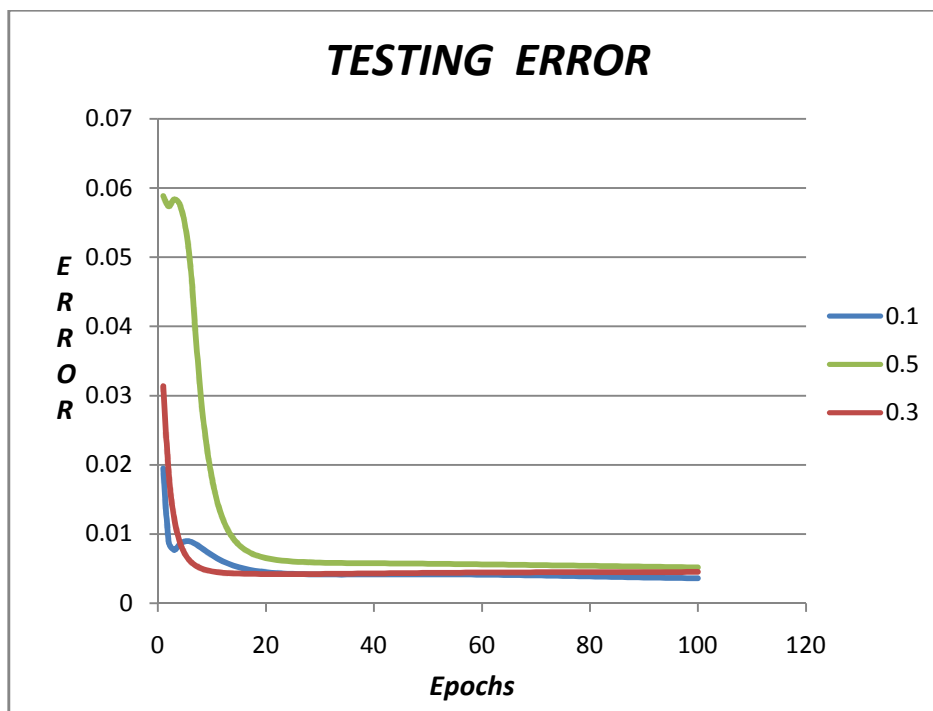
Γράφημα 5.3.1.5 Λάθη ελέγχου και επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμό μάθησης 0.5



Γράφημα 5.3.1.6 Ποσοστό επιτυχίας στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμό μάθησης 0.5

Στις πιο πάνω γραφικές παραστάσεις παρατηρούμε ότι τόσο το σφάλμα εκπαίδευσης όσο και το σφάλμα επαλήθευσης μειώνονται με την πάροδο του χρόνου σε αντίθεση με το ποσοστό επιτυχίας πρόβλεψης της ζήτησης του φυσικού αερίου το οποίο αυξάνεται. Επομένως στις επιτυχείς εκπαιδεύσεις το λάθος τείνει να μειώνεται όσο περνά ο χρόνος και το δίκτυο μαθαίνει με αποτέλεσμα τα ποσοστά επιτυχίας του στις προβλέψεις να αυξάνονται.

Στη συνέχεια παραθέτουμε το συγκριτικό γράφημα (5.3.2.7) μεταξύ των σφαλμάτων επαλήθευσης για τους τρεις ρυθμούς μάθησης.

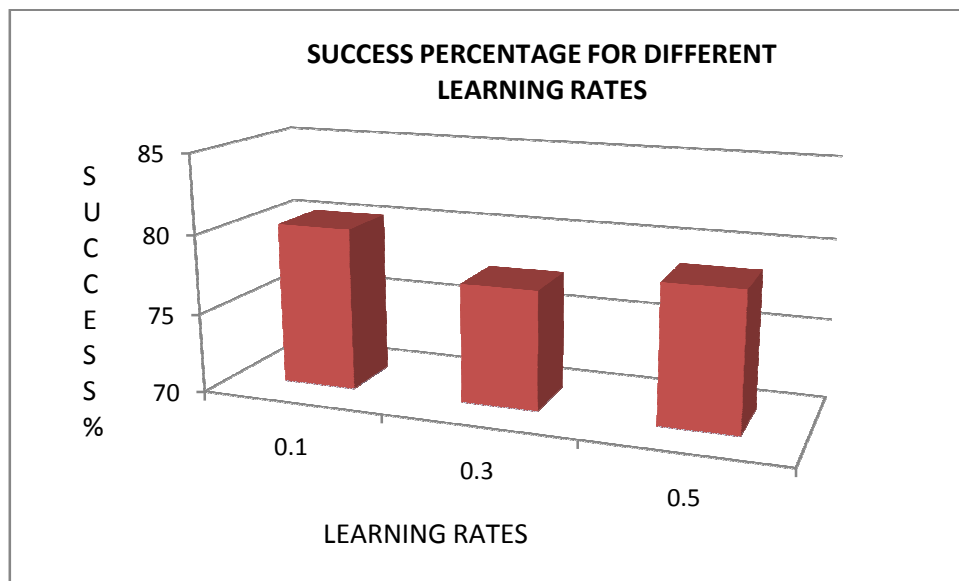


Γράφημα 5.3.1.7 Σφάλματα επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμούς μάθησης 0.1,0.3 και 0.5

Στη παραπάνω γραφική παράσταση παρατηρούμε ότι η μικρότερη τιμή του σφάλματος παρουσιάζεται όταν ο ρυθμός μάθησης είναι 0.1. Γι' αυτό το λόγο παρατηρείται το μεγαλύτερο ποσοστό επιτυχίας μ' αυτήν την τιμή του ρυθμού μάθησης. Στον πίνακα που ακολουθεί παρουσιάζονται τα ποσοστά επιτυχίας στην τελευταία εποχή για κάθε ένα από τους τρεις ρυθμούς μάθησης.

<i>Ρυθμός Μάθησης</i>	0.1	0.3	0.5
<i>Αριθμός Κέντρων</i>			
-	37	33	35
<i>Kohonen SOM</i>			
<i>Ποσοστό επιτυχίας</i>	80.12%	77.15%	78.13%

Πίνακας 5. 3.1.1 Ποσοστά επιτυχίας(%) στα δεδομένα επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμούς μάθησης 0.1, 0.3 και 0.5



Γράφημα 5.2.1.8 Τελικά ποσοστά επιτυχίας στα δεδομένα επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM – RBF για ρυθμούς μάθησης 0.1, 0.3 και 0.5

Από το πιο πάνω γράφημα φαίνεται ότι το μεγαλύτερο ποσοστό επιτυχίας παρουσιάζεται με ρυθμό μάθησης 0.1 και φτάνει το 80.12% το οποίο είναι χαμηλότερο με αυτό του δικτύου RBF του 5.2. Το γεγονός αυτό υποδηλώνει ότι τα δεδομένα μας **δεν είναι ομαδοποιημένα με αποτέλεσμα ο αλγόριθμός Kohonen SOM [11] να αδυνατεί να μας δώσει ομάδες δεδομένων**. Στη συνέχεια θα προσπαθήσουμε να βελτιώσουμε το ποσοστό αυτό όπως και του αλγορίθμου στο δίκτυο RBF κάνοντας χρήση διαφορετικών συνόλων δεδομένων για εκπαίδευση και έλεγχο των δικτύων.

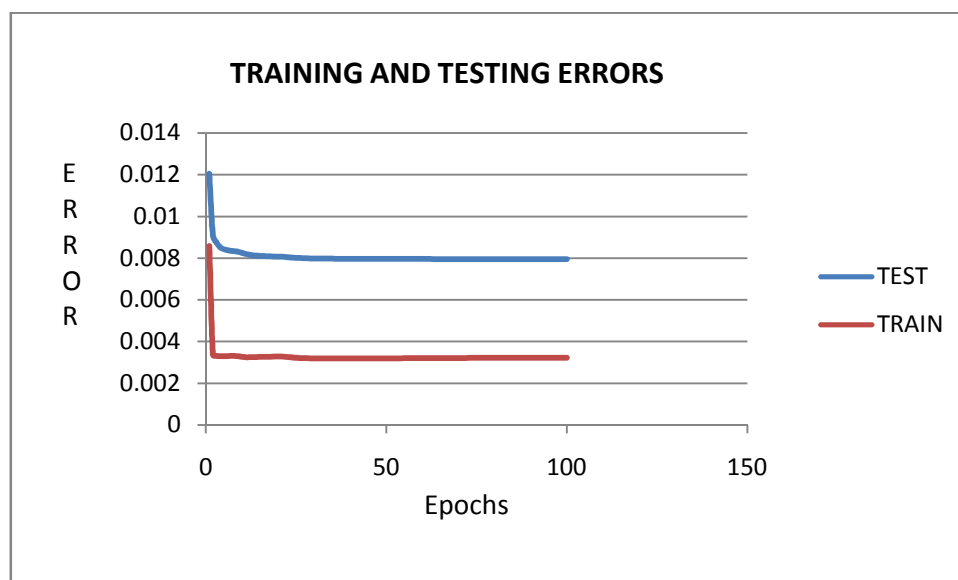
5.4 Αλλαγές στα σύνολα δεδομένων εκπαίδευσης και επαλήθευσης

Στα πειράματα που περιγράφονται στις πιο πάνω ενότητες (5.3 και 5.4), χρησιμοποιήθηκαν τα δεδομένα από τις 03/05/1992 μέχρι 30/09/1997 για εκπαίδευση ενώ τα δεδομένα από τις 02/05/1990 μέχρι 02/05/1992 για έλεγχο των δύο δικτύων (RBF και υβριδικού δικτύου Kohonen SOM –RBF). Στην προσπάθεια μας να βελτιώσουμε τα αποτελέσματα των δικτύων πήραμε **τυχαία δεδομένα** -από το σύνολο δεδομένων των 7 ετών- για εκπαίδευση και έλεγχο. Συγκεκριμένα δημιουργήσαμε δυο ζεύγη δεδομένων εκπαίδευσης και ελέγχου –πίνακας 5.4.1- . Στο πρώτο ζεύγος χρησιμοποιήσαμε το 70% των

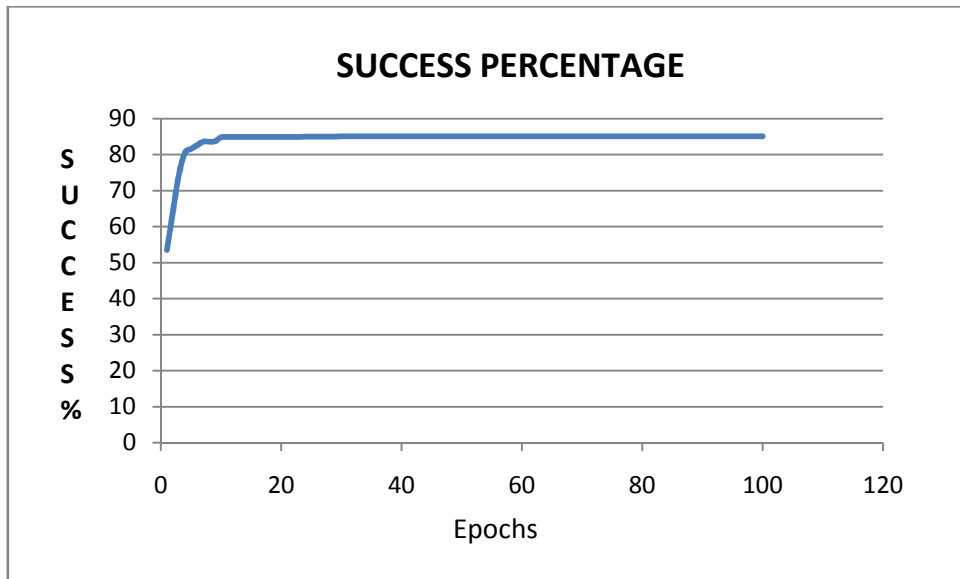
δεδομένων για εκπαίδευση ενώ το υπόλοιπο 30% για έλεγχο, ενώ στο δεύτερο ζεύγος χρησιμοποιήσαμε το 55% των δεδομένων για εκπαίδευση ενώ το υπόλοιπο 45% για έλεγχο. Τα αποτελέσματα που προέκυψαν από τις εκπαιδεύσεις των δικτύων παρουσιάζονται στα γραφήματα που ακολουθούν.

Σύνολο Δεδομένων	A	B	Γ
Δεδομένα	03/05/1992 μέχρι 30/09/1997 για εκπαίδευση ενώ τα δεδομένα από τις 02/05/1990 μέχρι 02/05/1992 για έλεγχο	Τυχαία επιλογή 70% εκπαίδευση 30% επαλήθευση	Τυχαία επιλογή 55% εκπαίδευση 45% επαλήθευση

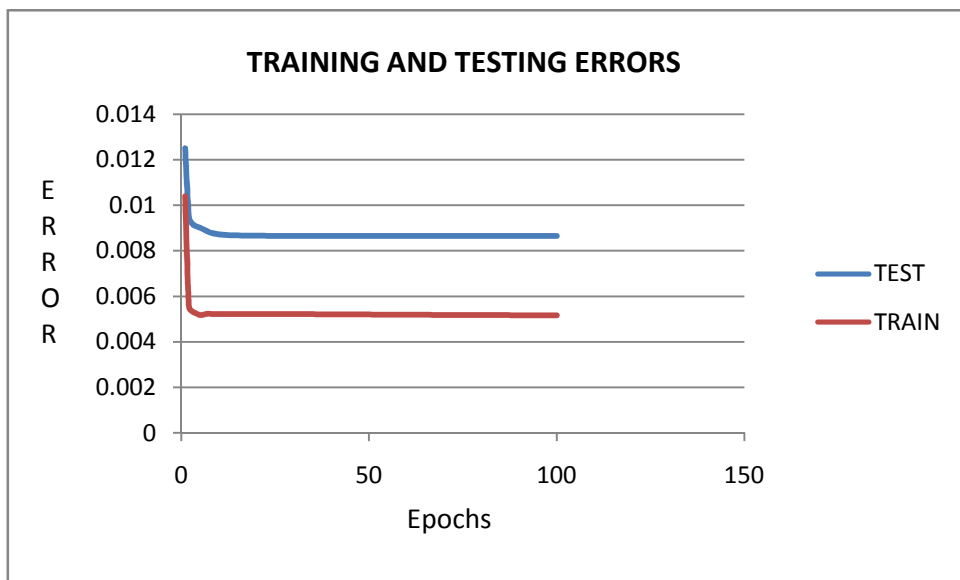
Πίνακας 5.4.1 Τα τρία σύνολα δεδομένων που χρησιμοποιήθηκαν για εκπαίδευση του δικτύου RBF και του υβριδικού δικτύου Kohonen SOM – RBF



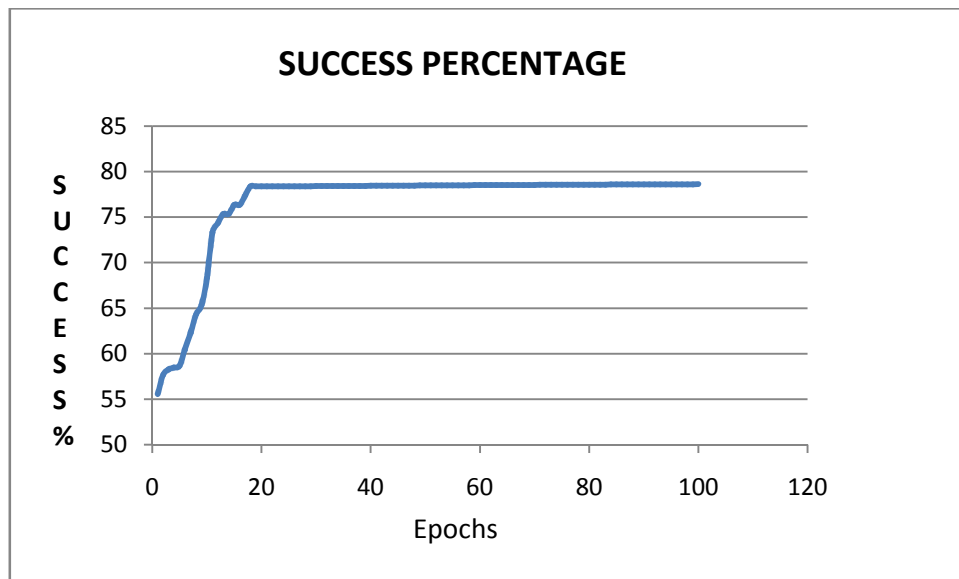
Γράφημα 5.4.1 Λάθη ελέγχου και επαλήθευσης δικτύου RBF για το σύνολο δεδομένων B



Γράφημα 5.4.2 Ποσοστό επιτυχίας δικτύου RBF για το σύνολο δεδομένων B



Γράφημα 5.4.3 Λάθη ελέγχου και επαλήθευσης δικτύου RBF για το σύνολο δεδομένων Γ

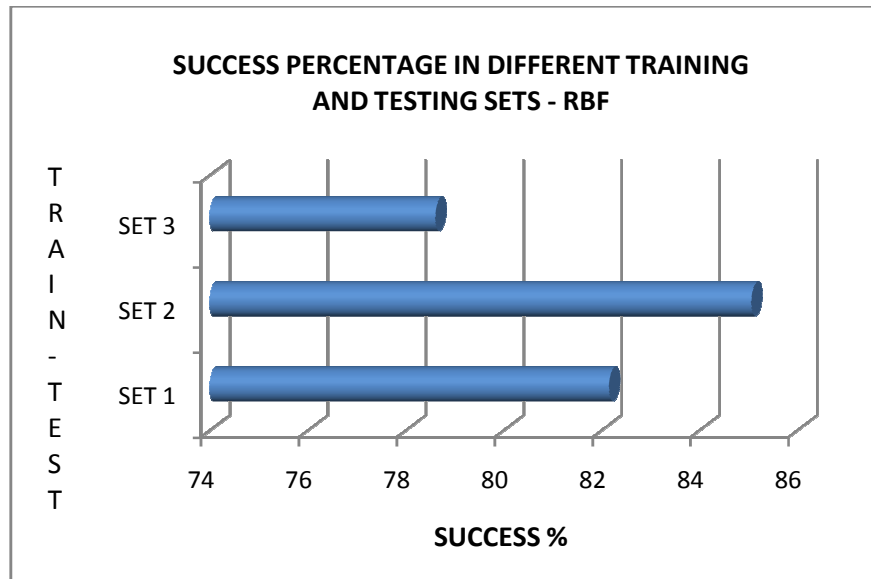


Γράφημα 5.4.4 Ποσοστό επιτυχίας δικτύου RBF για το σύνολο δεδομένων Γ

Από τα πιο πάνω γραφήματα παρατηρούμε ότι το σύνολο δεδομένων Β (70% εκπαίδευση και 30% έλεγχο) παρουσιάζει υψηλότερο ποσοστό επιτυχίας (85.05%) σε σύγκριση με το αρχικό σύνολο δεδομένων (5.2) το οποίο είχε ως αποτέλεσμα επιτυχία 82.152% σε αντίθεση με το σύνολό δεδομένων Γ (55% εκπαίδευση και 45% έλεγχο) που παρουσιάζει το χαμηλότερο ποσοστό επιτυχίας. Συνοπτικά όλα τα πιο πάνω παρουσιάζονται στον πίνακα 5.4.2 και γράφημα 5.4.5 που ακολουθούν.

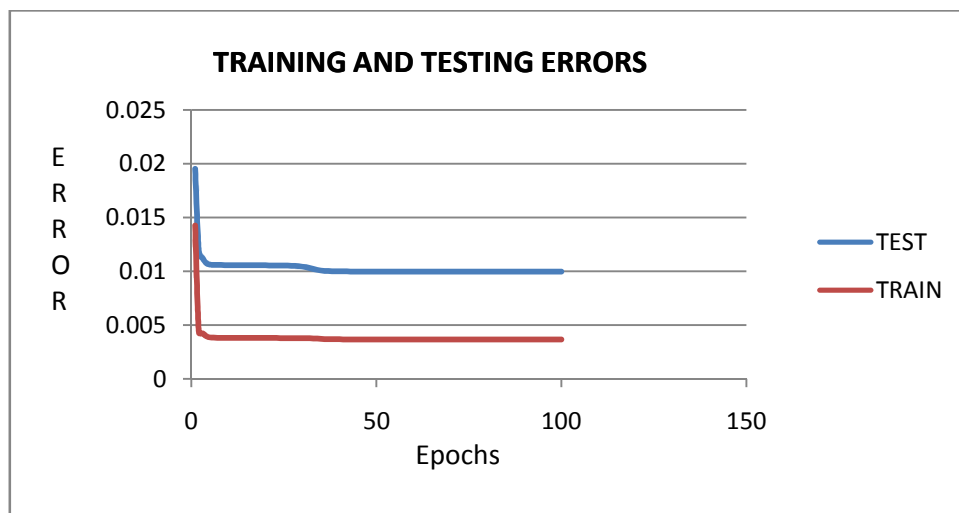
<i>Σύνολο Δεδομένων</i>	A	B	Γ
<i>Ποσοστό επιτυχίας</i>	82.152%	85.057%	78.60%

Πίνακας 5.4.2 Ποσοστά επιτυχίας(%) στα δεδομένα επαλήθευσης στο δίκτυο RBF για τα τρία σύνολα δεδομένων

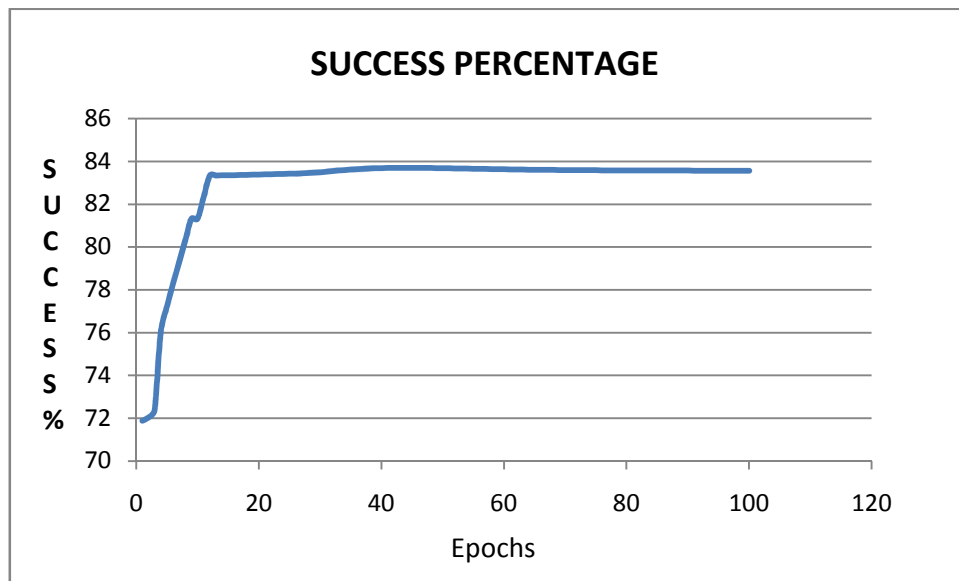


Γράφημα 5.4.5 Σύγκριση ποσοστών επιτυχίας για τα τρία σύνολα δεδομένων στο δίκτυο RBF

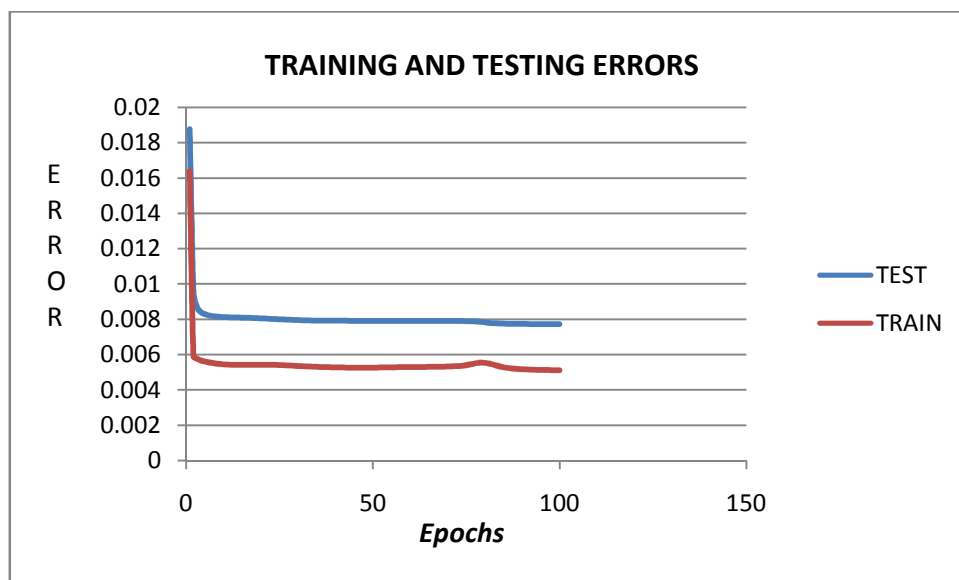
Τα αντίστοιχα πειράματα με τα ίδια σύνολα δεδομένων πραγματοποίησαν και στο υβριδικό δίκτυο το οποίο συνδυάζει τον αλγόριθμό ομαδοποίησης Kohonen – SOM [11] καθώς επίσης και δίκτυο συναρτήσεων αξονικών βάσεων (RBF [7][8][9]). Τα αποτελέσματα που προέκυψαν από την εκπαίδευση του δικτύου αυτού παρουσιάζονται στα γραφήματα 5.4.6-5.4.9 που ακολουθούν.



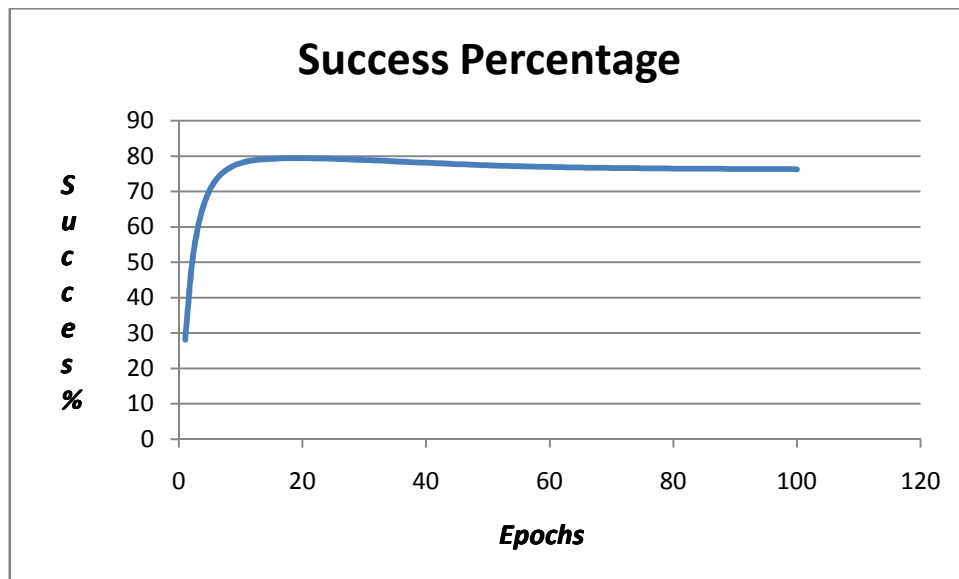
Γράφημα 5.4.6 Λάθη ελέγχου και επαλήθευσης του δικτύου Kohonen SOM -RBF για το σύνολο δεδομένων B



Γράφημα 5.4.7 Ποσοστό επιτυχίας του δικτύου Kohonen SOM -RBF για το σύνολο δεδομένων B



Γράφημα 5.4.8 Λάθη ελέγχου και επαλήθευσης του δικτύου Kohonen SOM -RBF για το σύνολο δεδομένων Γ

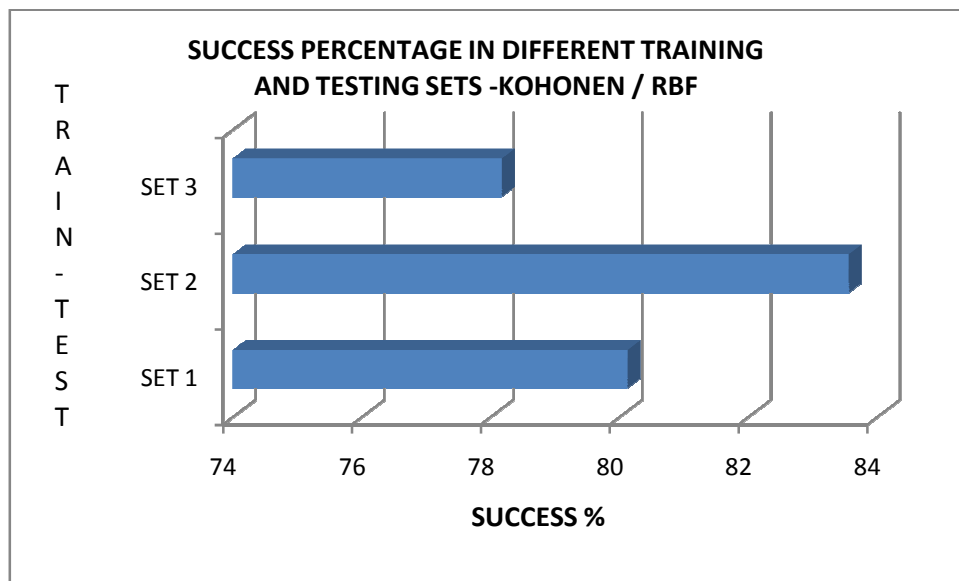


Γράφημα 5.4.9 Ποσοστό επιτυχίας δικτύου του Kohonen SOM -RBF για το σύνολο δεδομένων Γ

Από τα πιο πάνω γραφήματα παρατηρούμε ότι και στο δίκτυο αυτό, το σύνολο δεδομένων Β (70% εκπαίδευση και 30% έλεγχο) παρουσιάζει υψηλότερο ποσοστό επιτυχίας (83.56%) σε σύγκριση με το αρχικό σύνολο δεδομένων (5.3) το οποίο είχε ως αποτέλεσμα επιτυχία 80.12% σε αντίθεση με το σύνολό δεδομένων Γ (55% εκπαίδευση και 45% έλεγχο) που παρουσιάζει το χαμηλότερο ποσοστό επιτυχίας. Συνοπτικά όλα τα πιο πάνω παρουσιάζονται στον πίνακα 5.4.3 και γράφημα 5.4.10 που ακολουθούν.

<i>Σύνολο Δεδομένων</i>	A	B	Γ
<i>Πλήθος κέντρων</i>			
-	34	37	35
<i>Kohonen SOM</i>			
<i>Ποσοστό επιτυχίας</i>	80.12%	83.56%	78.17%

Πίνακας 5.4.3 Ποσοστά επιτυχίας(%) στα δεδομένα επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM (20x20) – RBF για τα τρία σύνολα δεδομένων

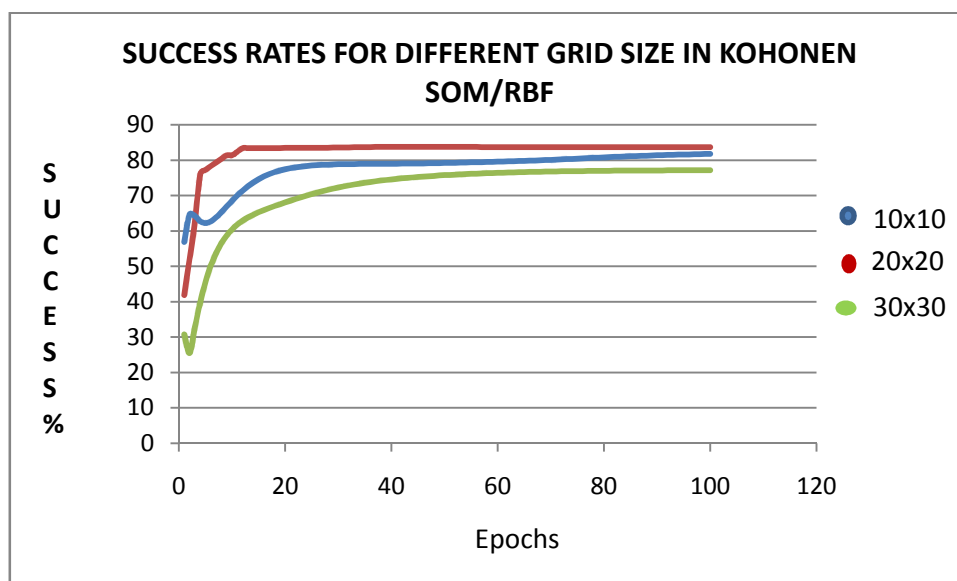


Γράφημα 5.4.10 Σύγκριση ποσοστών επιτυχίας για τα τρία σύνολα δεδομένων στο δίκτυο Kohonen SOM –RBF

Από όλα τα πιο πάνω προκύπτει ότι το δεύτερο σύνολο δεδομένων έχει τα υψηλότερα ποσοστά επιτυχίας και για τα δύο δίκτυα πράγμα που δείχνει ότι είναι το καταλληλότερο για εκπαίδευση τους.

5.5 Αλλαγές στις διαστάσεις του πλέγματος του αλγορίθμου Kohonen SOM

Αφού βρήκαμε ότι ο αλγόριθμος Kohonen SOM [11] – RBF [7][8][9] δίνει καλύτερα αποτελέσματα στο σύνολο δεδομένων που περιλαμβάνει τυχαία δεδομένα (70% για εκπαίδευση και 30% για έλεγχο) σε σύγκριση με τα άλλα δυο σύνολα δεδομένων που περιγράφονται στην ενότητα 5.4, πειραματιστήκαμε στην αλλαγή των διαστάσεων του πλέγματος του αλγορίθμου. Όπως αναφέραμε, μέχρι τώρα το μέγεθος του πλέγματος ήταν 20x20. Στην ενότητα αυτή θα παραθέσουμε τα αποτελέσματα που προέκυψαν με την χρήση πλέγματος διαστάσεων 10x10 και 30x30 με δεδομένα εκπαίδευσης και επαλήθευσης το σύνολο B της ενότητας 5.4. Στο γράφημα 5.5.1 που ακολουθεί παρουσιάζονται τα ποσοστά επιτυχίας (πίνακας 5.5.1) όπως αυτά προέκυψαν με τις αλλαγές των διαστάσεων του πλέγματος.

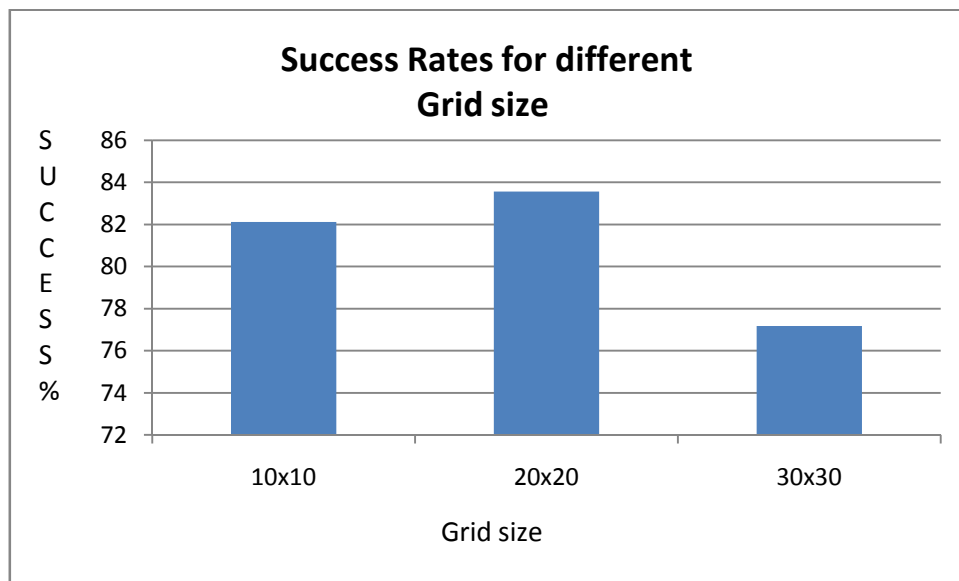


Γράφημα 5.5.1 Ποσοστά επιτυχίας (%), στα δεδομένα επαλήθευσης για διαφορετικές διαστάσεις πλέγματος στο υβριδικό δίκτυο Kohonen SOM – RBF

Διαστάσεις πλέγματος	10x10	20x20	30x30
Πλήθος ομάδων - κέντρων	18	35	78
Ποσοστό επιτυχίας	82.12%	83.56%	77.17%

Πίνακας 5.5.1 Ποσοστά επιτυχίας (%), στα δεδομένα επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM – RBF για διαφορετικές διαστάσεις πλέγματος

Από τα παραπάνω καθώς επίσης και από το γράφημα 5.5.2, καταλήγουμε στο συμπέρασμα ότι το ιδανικότερο μέγεθος του πλέγματος που μας δίνει το υψηλότερο ποσοστό επιτυχίας είναι 20x20 και το ποσοστό αυτό ανέρχεται στο 83.56%.



Γράφημα 5.5.2 Συγκριτικά ποσοστά επιτυχίας (%), στα δεδομένα επαλήθευσης στο υβριδικό δίκτυο Kohonen SOM – RBF για διαφορετικές διαστάσεις πλέγματος

5.6 Αλλαγές στις παραμέτρους του δικτύου με ανάδραση τύπου Jordan -Back Propagation Through Time

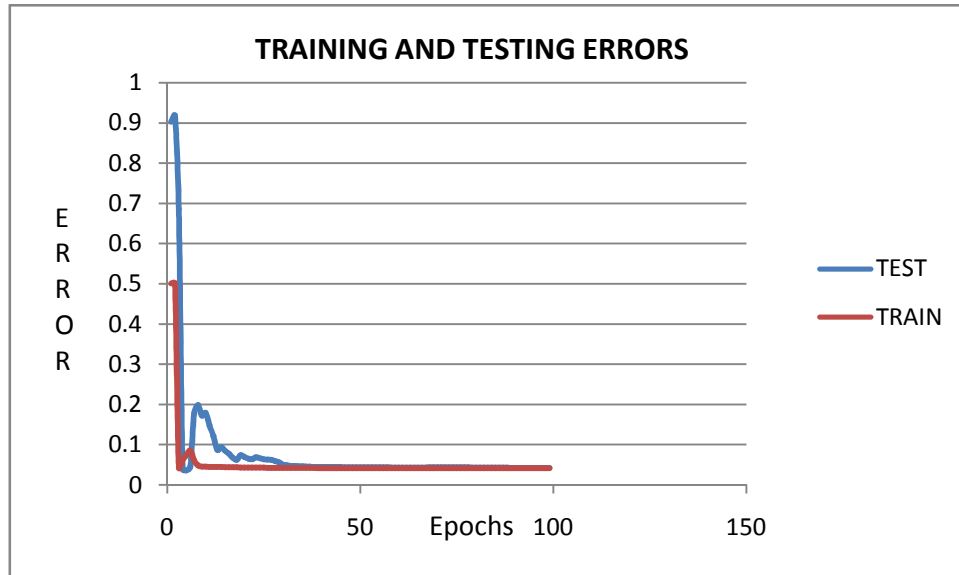
Σ' αυτό το μέρος της έρευνας, πειραματιστήκαμε στις αλλαγές των παραμέτρων του δικτύου με ανάδραση, με στόχο να βρούμε τις τιμές εκείνες οι οποίες μας δίνουν το υψηλότερο ποσοστό επιτυχίας. Αρχικά κρατήσαμε σταθερές τις τιμές για το πλήθος των κρυφών νευρώνων , για το πλήθος των χρονικών βημάτων και μεταβάλαμε την τιμή του ρυθμού μάθησης εκπαιδεύοντας το δίκτυο για 100 εποχές. Στη συνέχεια κρατήσαμε σταθερές τις τιμές για το ρυθμό μάθησης, το πλήθος των χρονικών βημάτων, δηλαδή των εισόδων και μεταβάλαμε την τιμή του πλήθους των νευρώνων. Τέλος, κρατήσαμε σταθερές τις τιμές για το πλήθος των κρυφών νευρώνων ,το ρυθμό μάθησης και μεταβάλαμε την τιμή των χρονικών βημάτων. Για τον υπολογισμό του σφάλματος εκπαίδευσης και επαλήθευσης χρησιμοποιήσαμε την τεχνική του τετραγωνικού λάθους. Το σφάλμα εκπαίδευσης υπολογίζεται πριν την αλλαγή των βαρών του δικτύου σε αντίθεση με το σφάλμα επαλήθευσης που υπολογίζεται αφού τελειώσει η εκπαίδευση – αλλάξουν τα βάρη. Ο κύκλος αυτός αποτελεί μια εποχή εκπαίδευσης. Τ' αποτελέσματα

για κάθε ένα από τους πιο πάνω πειραματισμούς παρουσιάζονται στις υπό ενότητες 5.6.1 – 5.6.3 αντίστοιχα.

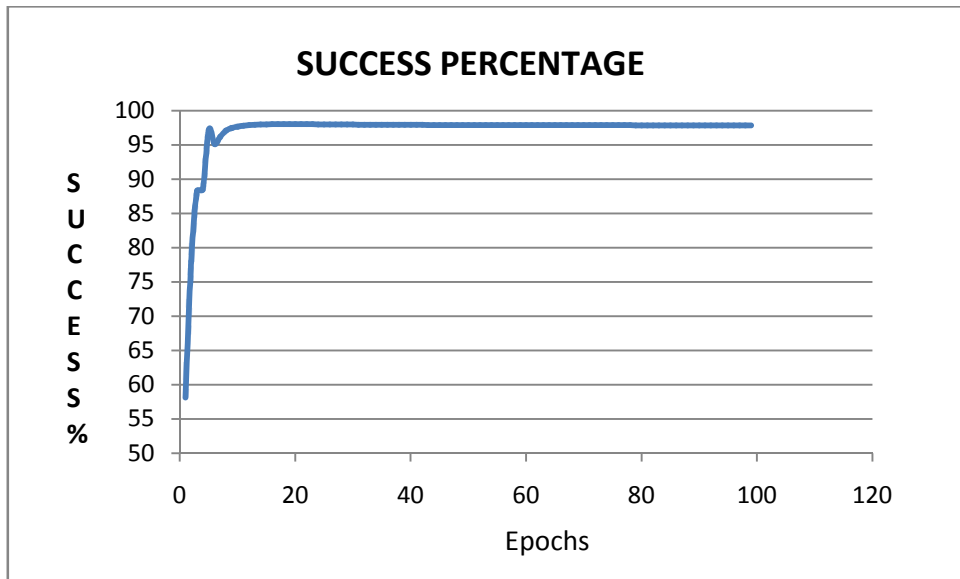
Σημ. Σε όλα τα πειράματα (5.6.1 – 5.6.3) η χρονική σταθερά (time constant) ήταν 1 ($T=1$)

5.6.1 Αλλαγές στο ρυθμό μάθησης

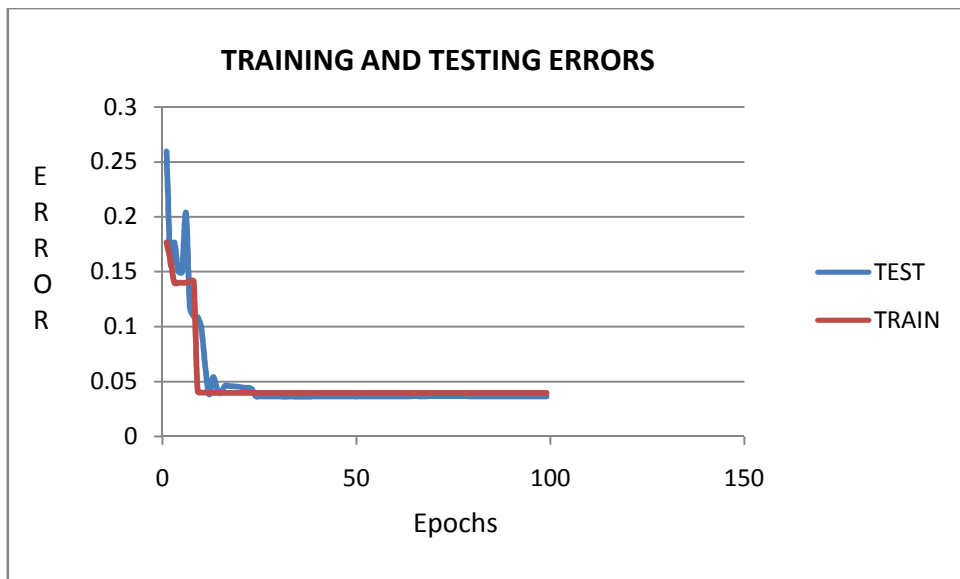
Όπως αναφέρθηκε παραπάνω στην ενότητα αυτή θα παρουσιάσουμε τ' αποτελέσματα που προέκυψαν κρατώντας σταθερό το πλήθος των νευρώνων και το πλήθος των χρονικών βημάτων μεταβάλλοντας το ρυθμό μάθησης. Πιο συγκεκριμένα αρχικοποιήσαμε τους κρυφούς νευρώνες σε 15 και το πλήθος των χρονικών βημάτων σε 10, εκπαιδεύοντας το δίκτυο για 100 εποχές με ρυθμούς μάθησης 0.1, 0.3 και 0.5. Τα αποτελέσματα των σφαλμάτων εκπαίδευσης και έλεγχου καθώς και τα ποσοστά επιτυχίας παρουσιάζονται στα γραφήματα 5.6.1.1 – 5.6.1.6 που ακολουθούν.



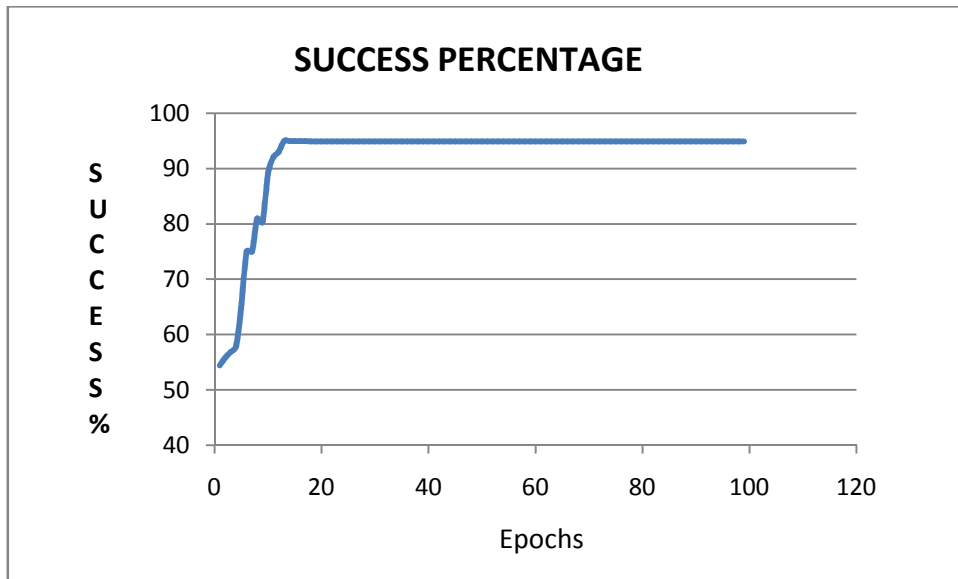
Γράφημα 5.6.1.1 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με ρυθμό μάθησης 0.1



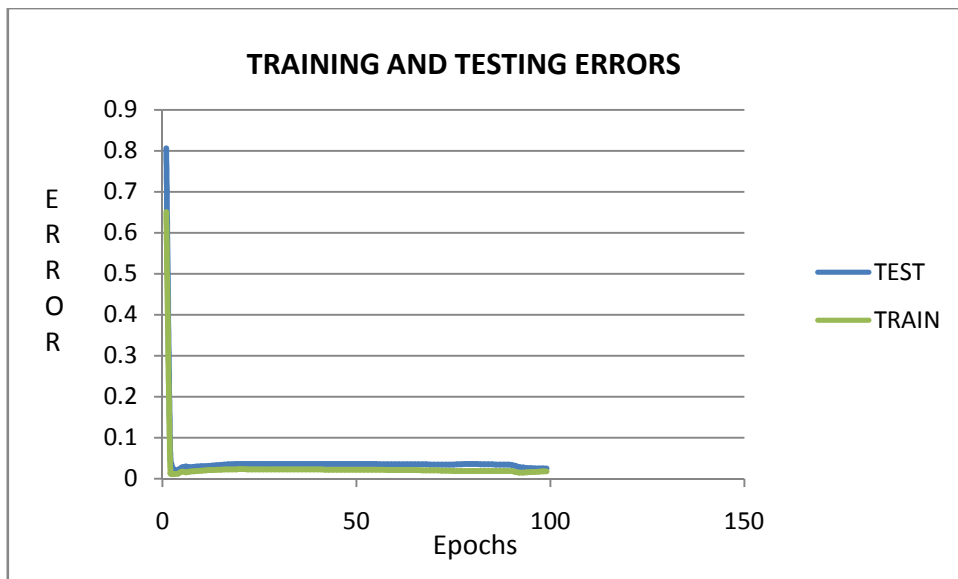
Γράφημα 5.6.1.2 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με ρυθμό μάθησης 0.1



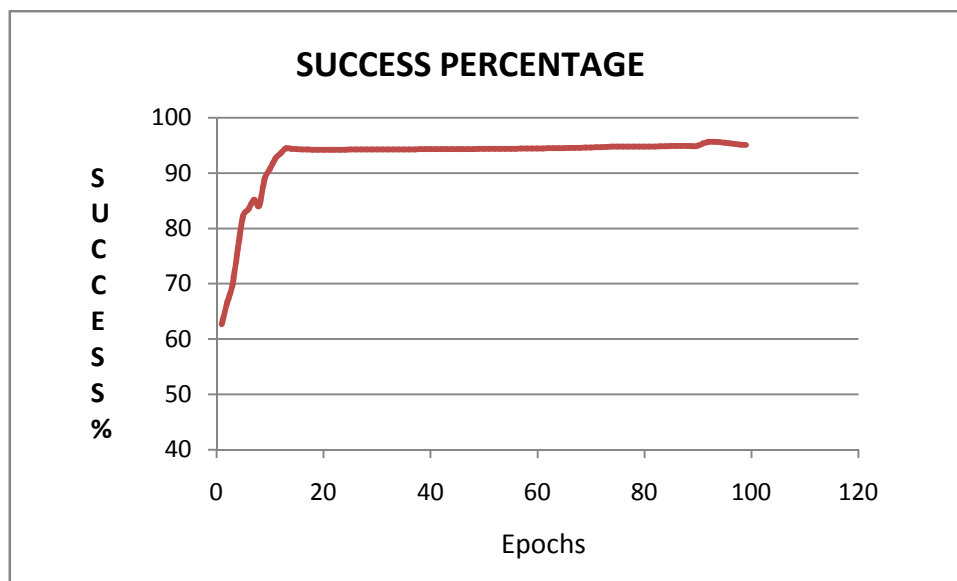
Γράφημα 5.6.1.3 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με ρυθμό μάθησης 0.3



Γράφημα 5.6.1.4 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με ρυθμό μάθησης 0.3



Γράφημα 5.6.1.5 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με ρυθμό μάθησης 0.5

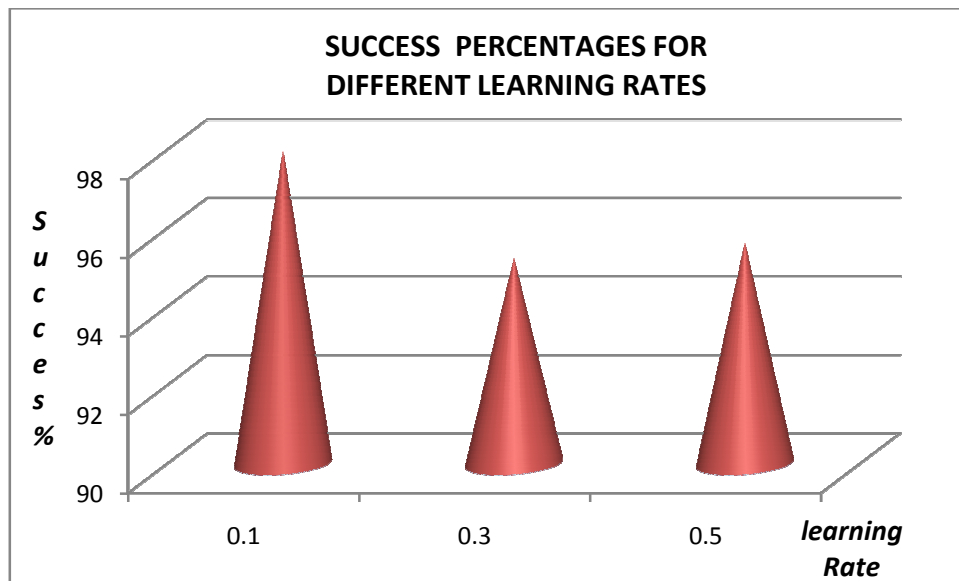


Γράφημα 5.6.1.6 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με ρυθμό μάθησης 0.5

Παρατηρούμε σ' όλες τις πιο πάνω γραφικές ότι τόσο το σφάλμα εκπαίδευσης όσο και ελέγχου μειώνονται με την πάροδο του χρόνου και το ποσοστό επιτυχίας φτάνει πάνω από 95% με ρυθμό μάθησης 0.1. Στον πίνακα 5.6.1.1 που ακολουθεί παρουσιάζονται τα τελικά ποσοστά επιτυχίας όπως αυτά προέκυψαν στην τελευταία επανάληψη.

<i>Ρυθμός Μάθησης</i>	0.1	0.3	0.5
<i>Ποσοστό επιτυχίας</i>	97.85%	94.92%	95.10%

Πίνακας 5.6.1.1 Ποσοστά επιτυχίας(%) στα δεδομένα επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan για ρυθμούς μάθησης 0.1, 0.3 και 0.5

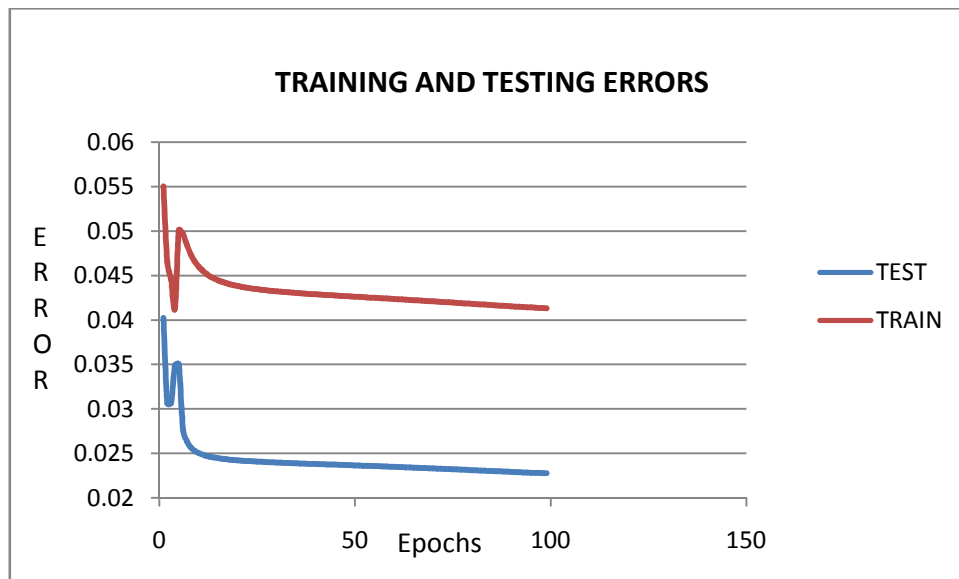


Γράφημα 5.6.1.7 Τελικά ποσοστά επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan για ρυθμούς μάθησης 0.1, 0.3 και 0.5

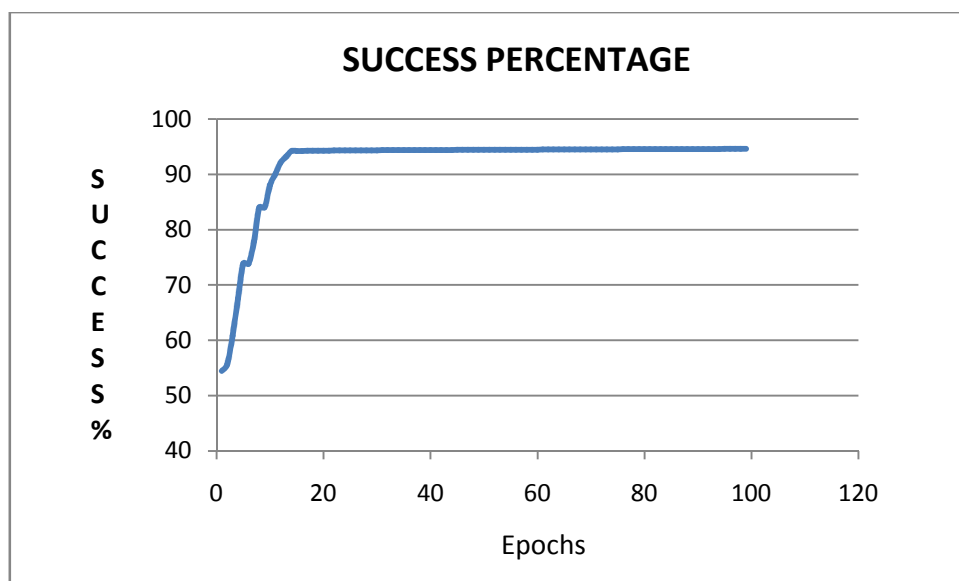
Από το πιο πάνω γράφημα φαίνεται ότι το μεγαλύτερο ποσοστό επιτυχίας παρουσιάζεται με ρυθμό μάθησης 0.1 και φτάνει το 97.85%, **το οποίο αποδεικνύει ότι η χρονοσειράς της ζήτησης έχει μια χρονική επάρκεια.** Στη συνέχεια θα μεταβάλλουμε το πλήθος των κρυφών νευρώνων κρατώντας το ρυθμό μάθησης σταθερό με την τιμή που έδωσε το υψηλότερο ποσοστό, δηλαδή 0.1.

5.6.2 Αλλαγές στον αριθμό των κρυφών νευρώνων

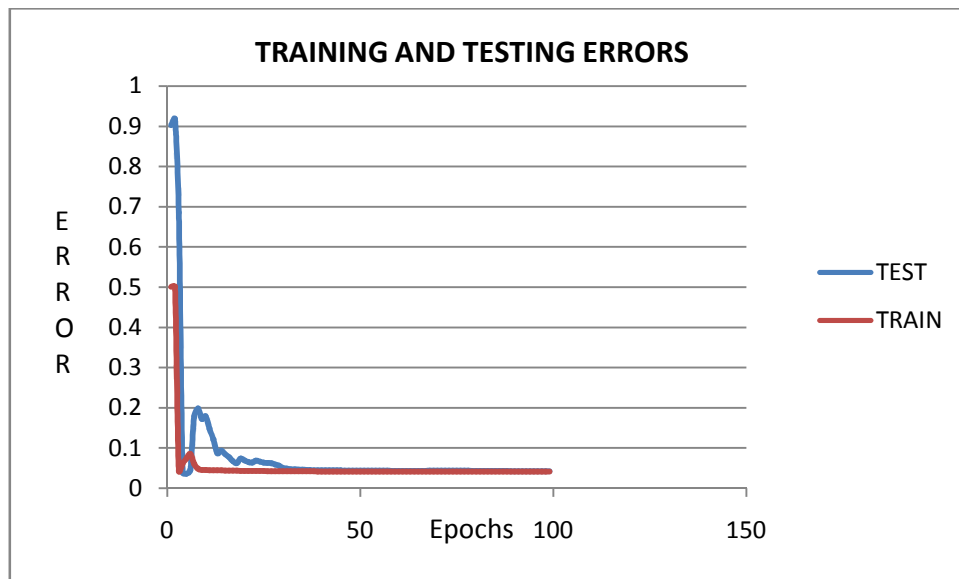
Στην ενότητα αυτή θα παρουσιάσουμε τ' αποτελέσματα που προέκυψαν κρατώντας σταθερό το ρυθμό μάθησης και το πλήθος των χρονικών βημάτων μεταβάλλοντας το πλήθος των κρυφών νευρώνων. Πιο συγκεκριμένα αρχικοποιήσαμε το ρυθμό μάθησης σε 0.1 -ο οποίος δίνει το υψηλότερο ποσοστό επιτυχίας- και το πλήθος των χρονικών βημάτων σε 10 εκπαιδεύοντας το δίκτυο για 100 εποχές με αριθμό νευρώνων 10, 16 και 24. Τα αποτελέσματα των σφαλμάτων εκπαίδευσης και έλεγχου καθώς και τα ποσοστά επιτυχίας παρουσιάζονται στα γραφήματα 5.6.2.1 – 5.6.2.6 που ακολουθούν.



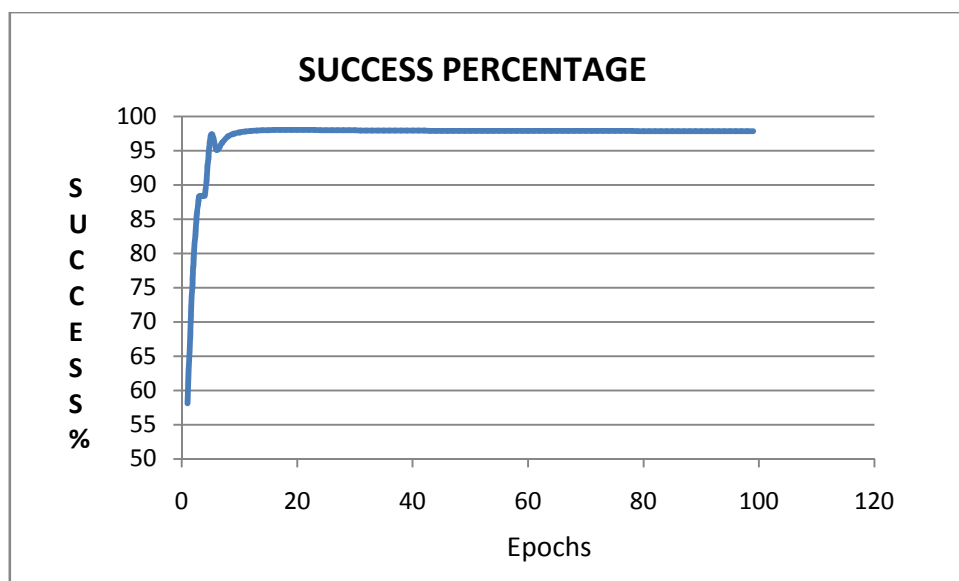
Γράφημα 5.6.2.1 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με 10 κρυφούς νευρώνες



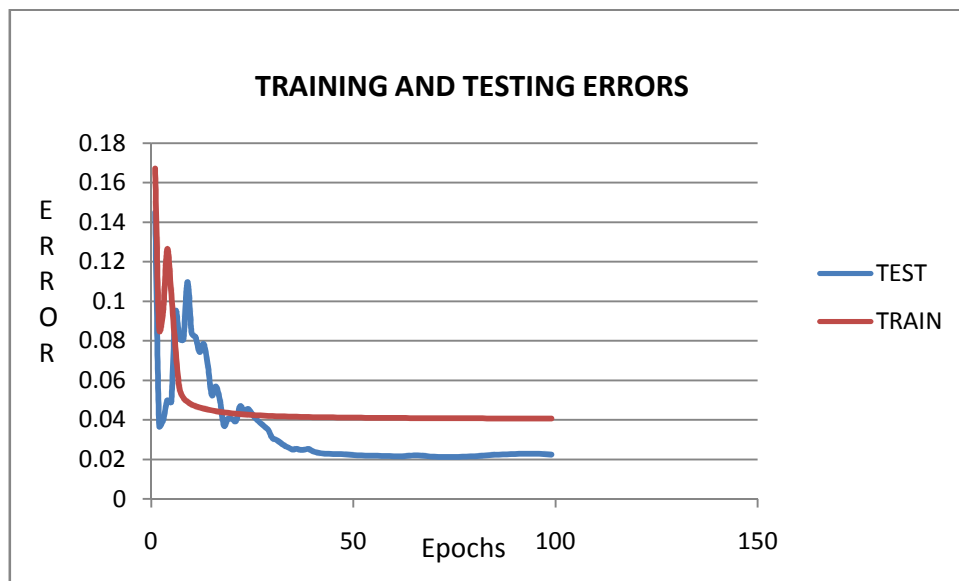
Γράφημα 5.6.2.2 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με 10 κρυφούς νευρώνες



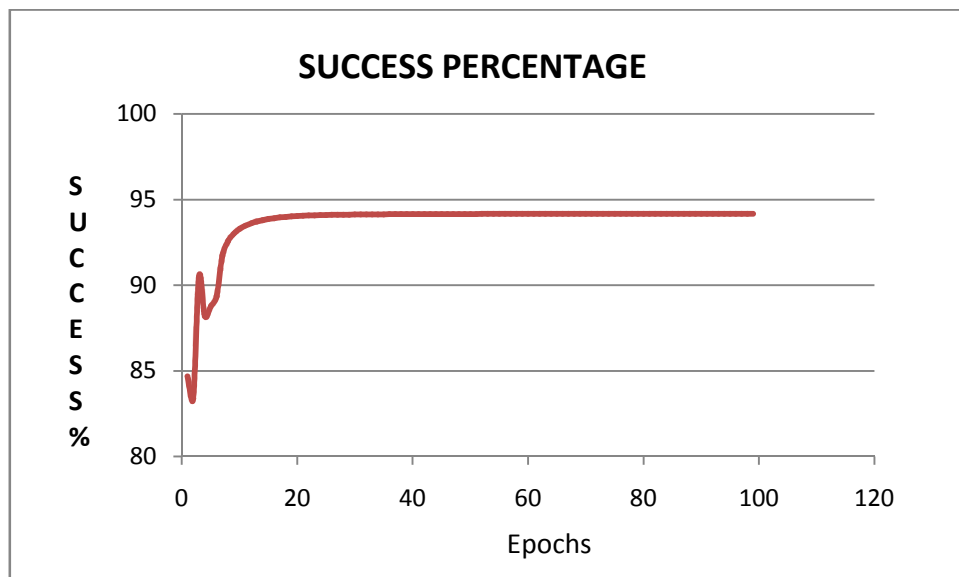
Γράφημα 5.6.2.3 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με 16 κρυφούς νευρώνες



Γράφημα 5.6.2.4 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με 16 κρυφούς νευρώνες



Γράφημα 5.6.2.5 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με 24 κρυφούς νευρώνες



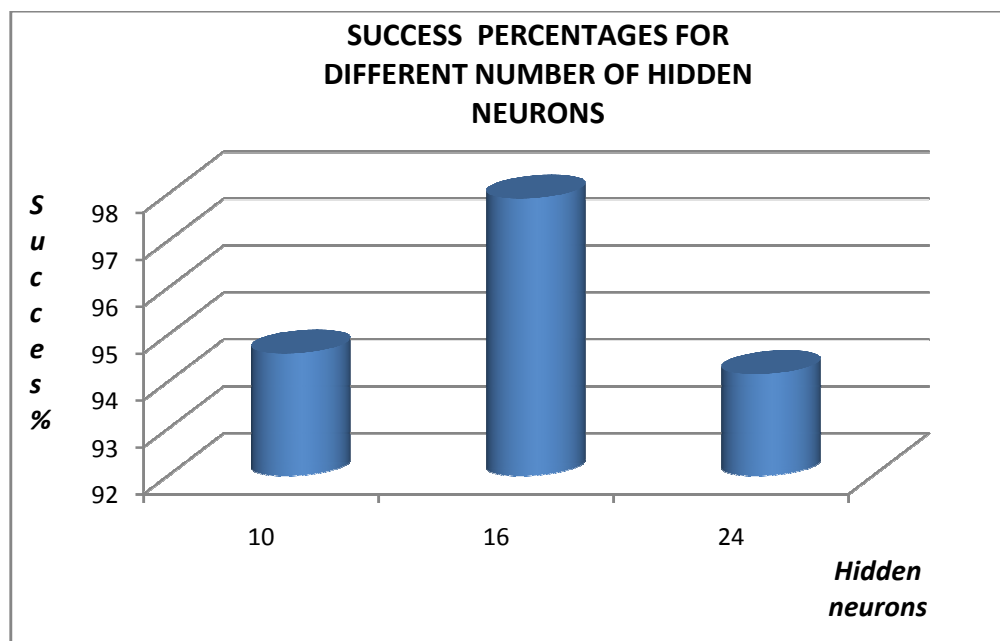
Γράφημα 5.6.2.6 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με 24 κρυφούς νευρώνες

Παρατηρούμε σ' όλες τις πιο πάνω γραφικές ότι τόσο το σφάλμα εκπαίδευσης όσο και ελέγχου μειώνονται με την πάροδο του χρόνου και το ποσοστό επιτυχίας τελικά αυξάνεται φτάνοντας το 97.90% με τη χρήση 16 κρυφών νευρώνων.

Στον πίνακα 5.6.1.1 που ακολουθεί παρουσιάζονται τα τελικά ποσοστά επιτυχίας όπως αυτά προέκυψαν στη τελευταία επανάληψη.

<i>Κρυφοί νευρώνες</i>	10	16	24
<i>Ποσοστό επιτυχίας</i>	94.60%	97.90%	94.16%

Πίνακας 5.6.1.1 Ποσοστά επιτυχίας (%) στα δεδομένα ελέγχου στο δίκτυο με ανάδραση τύπου Jordan για 10, 16 και 24 κρυφούς νευρώνες



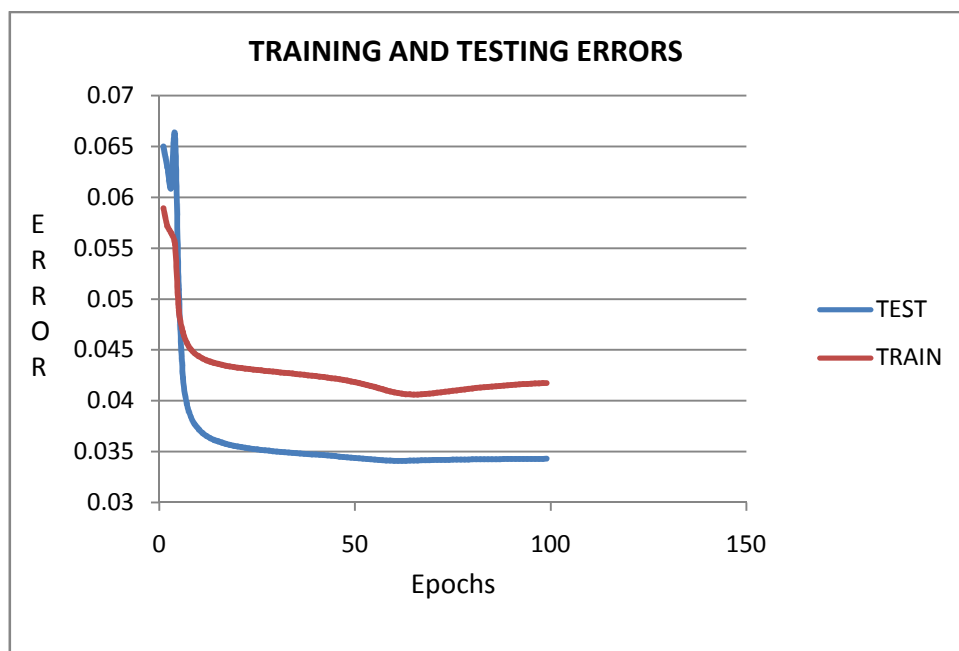
Γράφημα 5.6.1.7 Τελικά ποσοστά επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με 10, 16 και 24 νευρώνες

Από το πιο πάνω γράφημα φαίνεται ότι το μεγαλύτερο ποσοστό επιτυχίας παρουσιάζεται με ρυθμό μάθησης 0.1 και 16 κρυφούς νευρώνες και φτάνει το 97.90%. Στη συνέχεια θα μεταβάλουμε το πλήθος των χρονικών βημάτων κρατώντας το ρυθμό

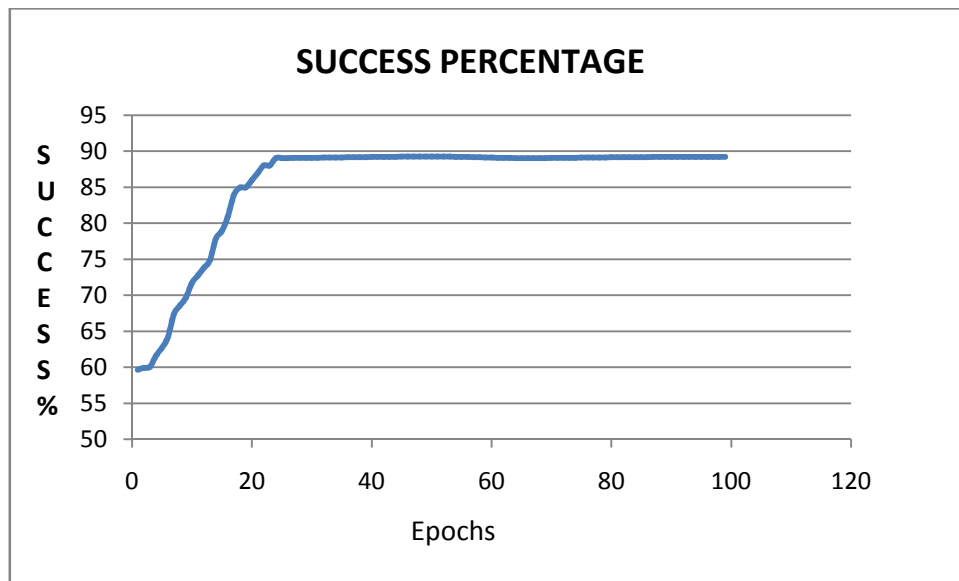
μάθησης σταθερό με την τιμή που έδωσε το υψηλότερο ποσοστό δηλαδή 0.1 καθώς επίσης και 16 κρυφούς νευρώνες.

5.6.3 Αλλαγές στον αριθμό των χρονικών βημάτων (πλήθος εισόδων)

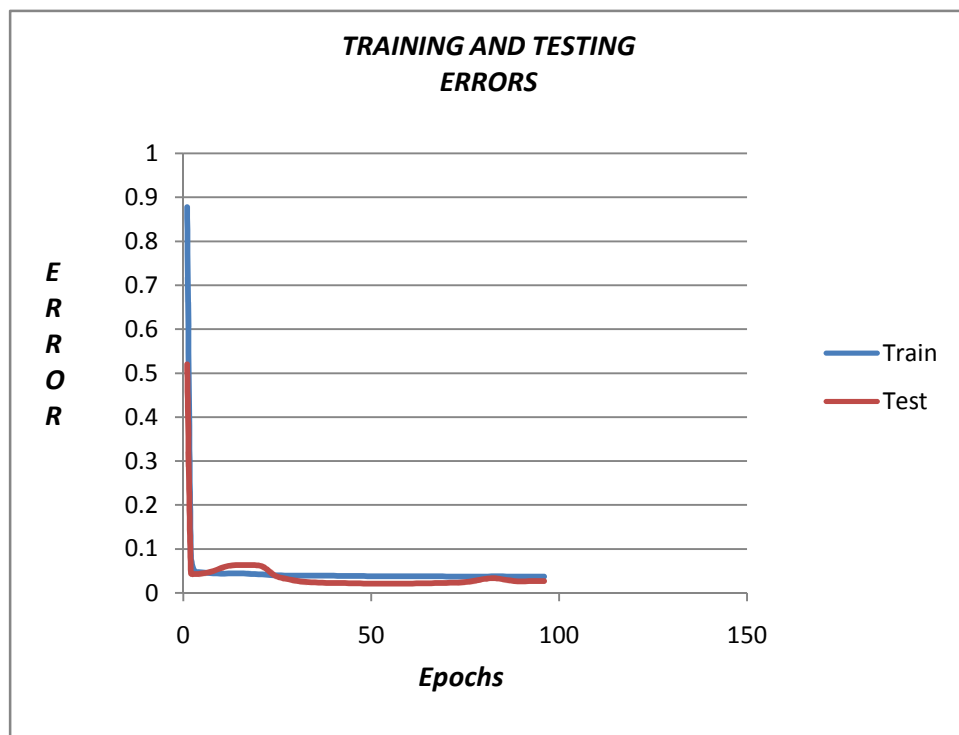
Στην ενότητα αυτή θα παρουσιάσουμε τ' αποτελέσματα που προέκυψαν κρατώντας σταθερό το ρυθμό μάθησης και το πλήθος των κρυφών νευρώνων μεταβάλλοντας το πλήθος των χρονικών βημάτων. Πιο συγκεκριμένα αρχικοποιήσαμε το ρυθμό μάθησης σε 0.3 -ο οποίος δίνει το υψηλότερο ποσοστό επιτυχίας- και το πλήθος κρυφών νευρώνων σε 16 -όπου έχουμε το πιο ψηλό ποσοστό επιτυχίας- εκπαιδεύοντας το δίκτυο για 100 εποχές με αριθμό χρονικών βημάτων 5, 15 και 20. Τα αποτελέσματα των σφαλμάτων εκπαίδευσης και έλεγχου καθώς και τα ποσοστά επιτυχίας παρουσιάζονται στα γραφήματα 5.6.3.1-5.6.3.6 που ακολουθούν.



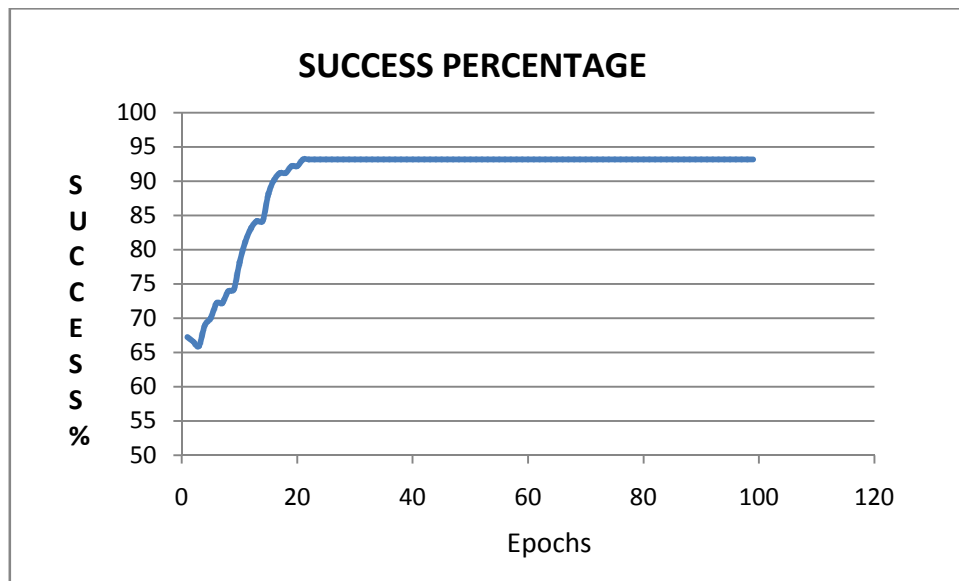
Γράφημα 5.6.3.1 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με 5 χρονικά βήματα



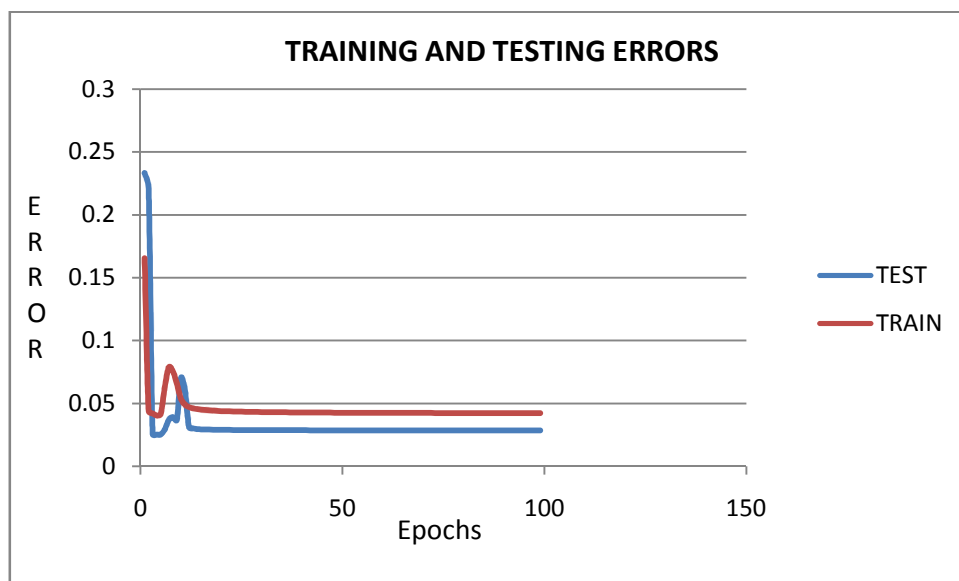
Γράφημα 5.6.3.2 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με 5 χρονικά βήματα



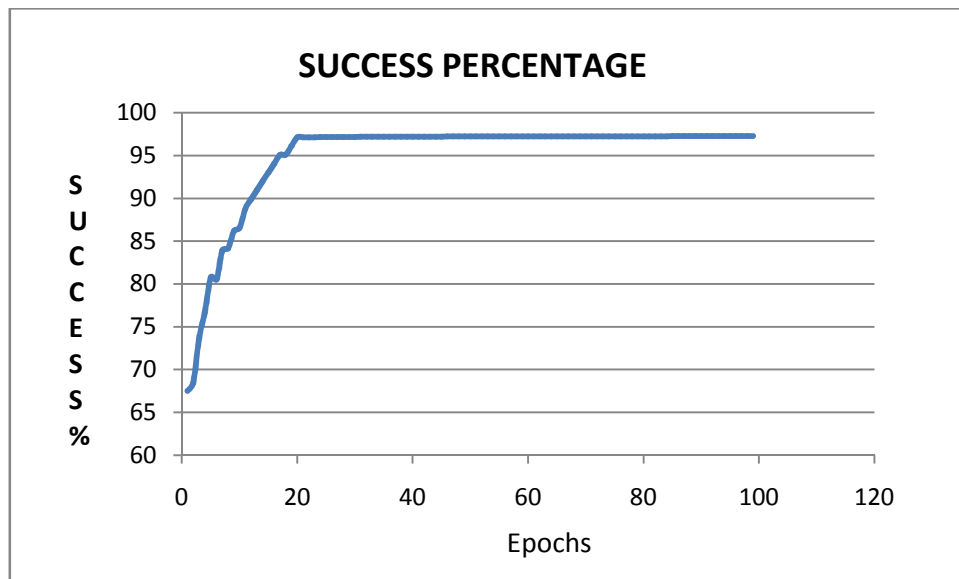
Γράφημα 5.6.3.3 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με 15 χρονικά βήματα



Γράφημα 5.6.3.4 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με 15 χρονικά βήματα



Γράφημα 5.6.3.5 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με 20 χρονικά βήματα

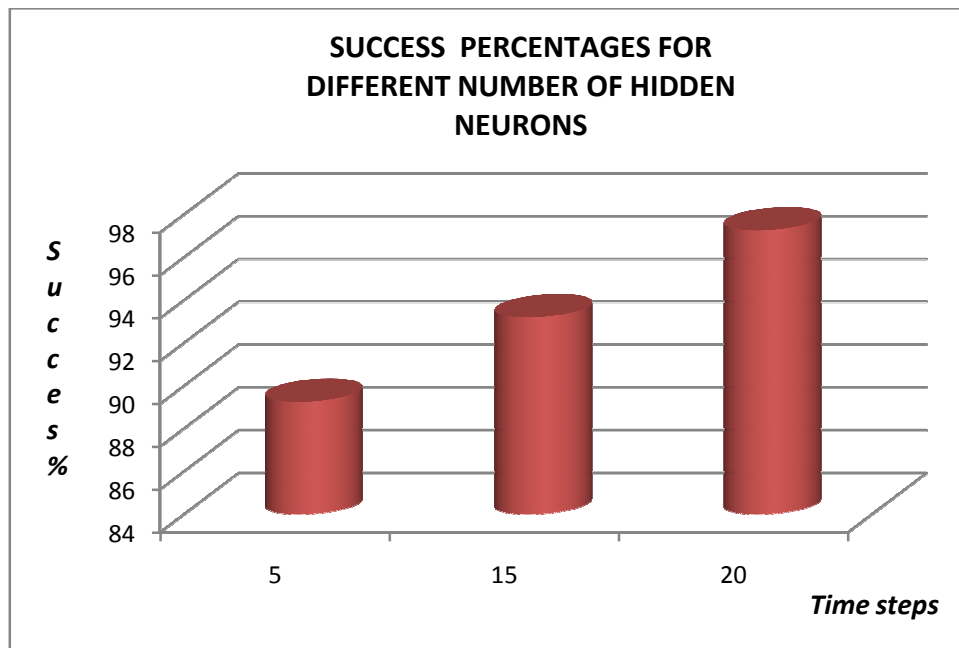


Γράφημα 5.6.3.6 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με 20 χρονικά βήματα

Παρατηρούμε σ' όλες τις πιο πάνω γραφικές παραστάσεις ότι τόσο το σφάλμα εκπαίδευσης όσο και ελέγχου μειώνονται με την πάροδο του χρόνου και το ποσοστό επιτυχίας φτάνει έως το 97.25% όταν χρησιμοποιήσαμε 20 χρονικά βήματα. Στον πίνακα 5.6.3.1 που ακολουθεί παρουσιάζονται τα τελικά ποσοστά επιτυχίας όπως αυτά προέκυψαν στην τελευταία επανάληψη.

<i>Χρονικά βήματα</i>	5	15	20
<i>Ποσοστό επιτυχίας</i>	89.24 %	93.19 %	97.25 %

Πίνακας 5.6.3.1 Ποσοστά επιτυχίας(%) στα δεδομένα επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan για 5, 15 και 20 χρονικά βήματα



Γράφημα 5.6.3.7 Τελικά ποσοστά επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan για 5, 15 και 20 χρονικά βήματα

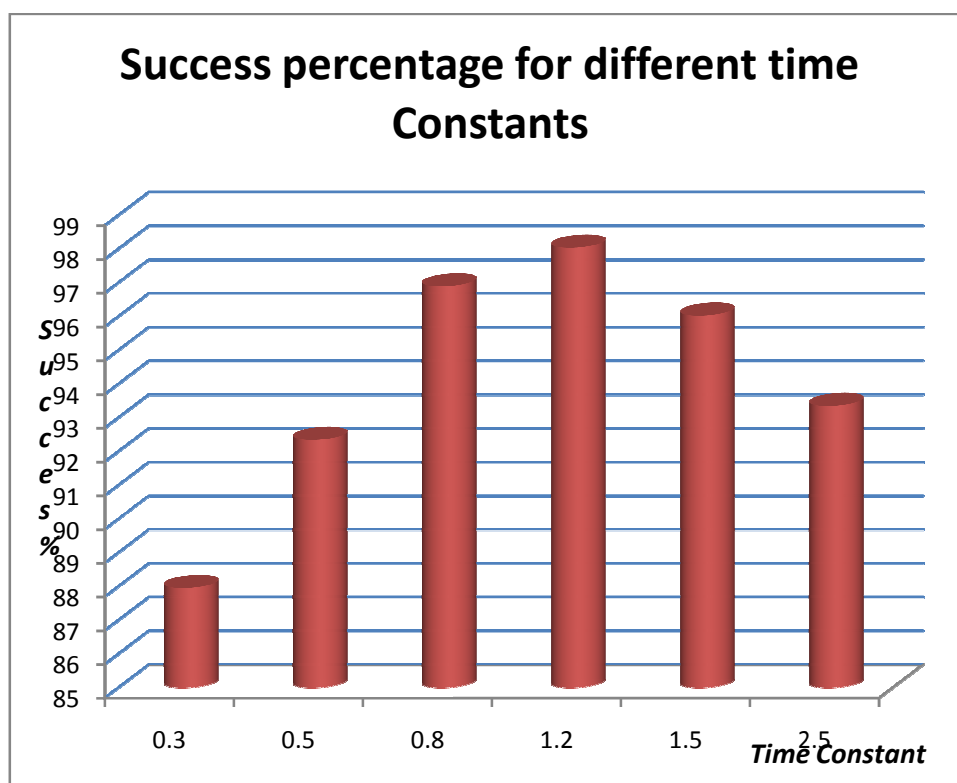
Από το γράφημα 5.6.3.7 φαίνεται ότι το μεγαλύτερο ποσοστό επιτυχίας παρουσιάζεται με 20 χρονικά βήματα και φτάνει το 97.25% σ' αντίθεση με τα 10 χρονικά βήματα της ενότητας 5.6.2 που φτάνει το 97.90%. Επομένως με βάση τα παραπάνω το μεγαλύτερο ποσοστό στο δίκτυο με ανάδραση τύπου Jordan είναι 97.90%, το οποίο παρατηρήθηκε με τη χρήση **16 κρυφών νευρώνων, 10 χρονικά βήματα και ρυθμό μάθησης 0.1.**

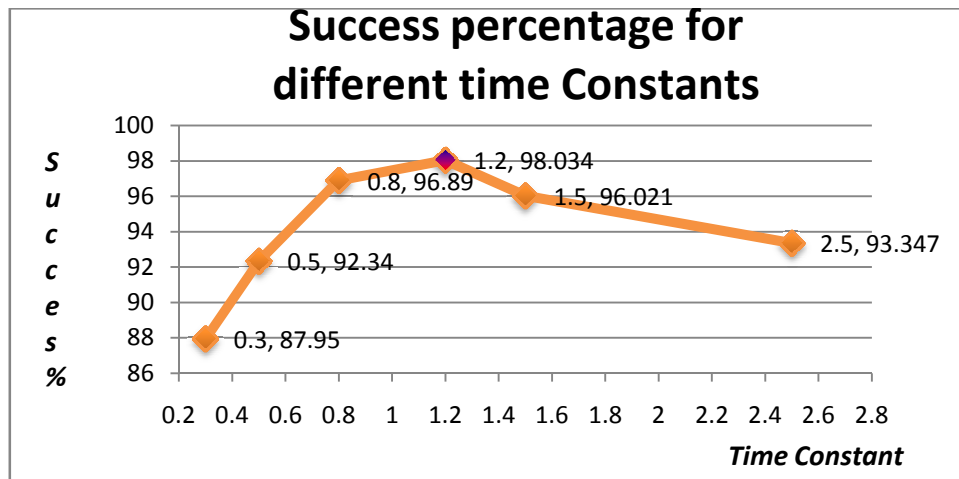
5.6.4 Αλλαγή των τιμών της χρονικής σταθεράς (time constant)

Στις προηγούμενες ενότητες αφού κάναμε αρκετά πειράματα όσον αφορά τις τιμές των παραμέτρων, καταλήξαμε ότι οι το δίκτυο παρουσιάζει το μεγαλύτερο ποσοστό επιτυχίας (97.90%) με τη χρήση **16 κρυφών νευρώνων, 10 χρονικά βήματα και ρυθμό μάθησης 0.1.** Όπως αναφέραμε, στα πειράματα χρησιμοποιήσαμε χρονική σταθερά ίση με 1. Στην υποενότητα αυτή πειραματιστήκαμε αλλάζοντας την τιμή της χρονικής σταθεράς με στόχο να βρούμε τη βέλτιστη τιμή της (optimum value) η οποία μας δίνει το υψηλότερο ποσοστό επιτυχίας. Τα αποτελέσματα που προέκυψαν συνοψίζονται στον πίνακα και γράφημα 5.6.4.1.

<i>Χρονική σταθερά</i>	<i>Ποσοστό επιτυχίας</i>
0.3	87.95 %
0.5	92.34 %
0.8	96.89 %
1.2	98.034 %
1.5	96.021 %
2.5	93.347 %

Πίνακας 5.6.4.1 Ποσοστά επιτυχίας(%) στα δεδομένα επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan με διαφορετικές τιμές της χρονικής σταθεράς





Γράφημα 5.6.4.1 Τελικά ποσοστά επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan με διαφορετικές τιμές της χρονικής σταθεράς

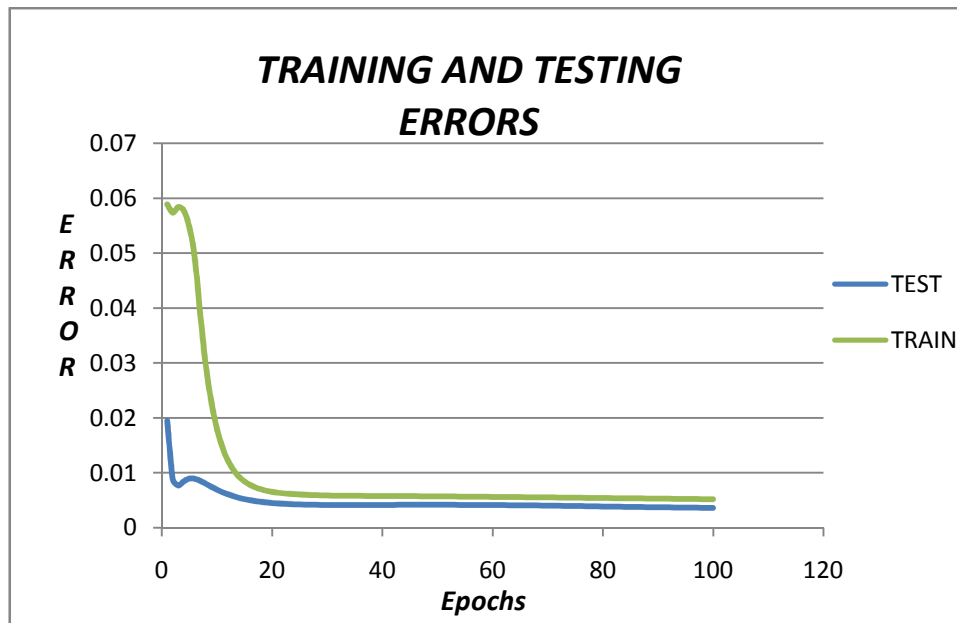
Από τα πιο πάνω, παρατηρούμε ότι με χρονική σταθερά 1.2 επιτυγχάνεται ποσοστό επιτυχίας 98.034%, το οποίο είναι το υψηλότερο ποσοστό που παρατηρήθηκε στο δίκτυο με ανάδραση τύπου Jordan.

Τέλος, παρατηρούμε (γράφημα 5.6.4.1) ότι για τιμές της χρονικής σταθεράς μεγαλύτερες του 1.2 το ποσοστό επιτυχίας αρχίζει να μειώνεται πράγμα που υποδηλώνει ότι βάση των συγκεκριμένων δεδομένων το δίκτυο δεν έχει ανάγκη για μεγαλύτερο βάθος μνήμης (memory depth) και επομένως η βέλτιστη τιμή (optimum value) της χρονικής σταθεράς είναι 1.2. Επιπλέον, καταλήγουμε στο συμπέρασμα ότι στο δίκτυο αυτό υπάρχουν δύο είδη μνήμης, το πλήθος των εισόδων (time steps) και η χρονική σταθερά (time constant) η οποία καθορίζει το βάθος της μνήμης (memory depth).

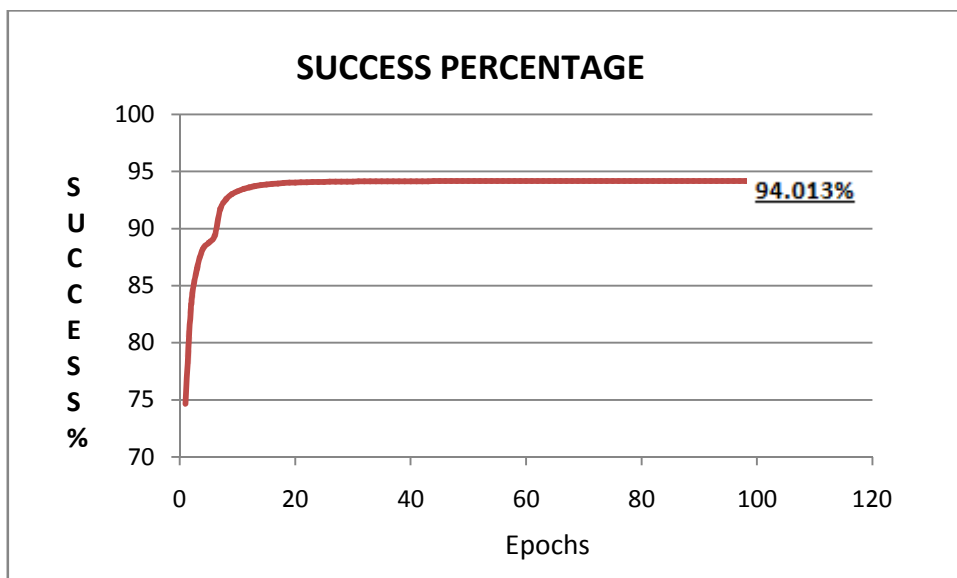
5.6.5 Τροποποίηση στον τρόπο υπολογισμού της επόμενης τιμής της ζήτησης

Στις προηγούμενες ενότητες, για την πρόβλεψη της ζήτησης τη χρονική στιγμή $N+1$, χρησιμοποιούσαμε την προηγούμενη επιθυμητή τιμή της ζήτησης και όχι αυτή που είχε προβλεφθεί. Έτσι τροποποιήσαμε τον κώδικα του δυναμικού δικτύου έτσι ώστε για τον υπολογισμό της επόμενης τιμής ζήτησης να λαμβάνει υπόψη και την προηγούμενη πρόβλεψη. Στη συνέχεια εκπαιδεύσαμε το δίκτυο με τις βέλτιστες παραμέτρους –

ρυθμός μάθησης 0.1, 16 κρυφούς νευρώνες, 10 χρονικά βήματα και χρονική σταθερά 1.2- και τα αποτελέσματα φαίνονται στα γραφήματα 5.6.5.1 και 5.6.5.2.



Γράφημα 5.6.5.1 Λάθη ελέγχου και επαλήθευσης στο δίκτυο με ανάδραση τύπου Jordan λαμβάνοντας υπόψη την προηγούμενη πρόβλεψη

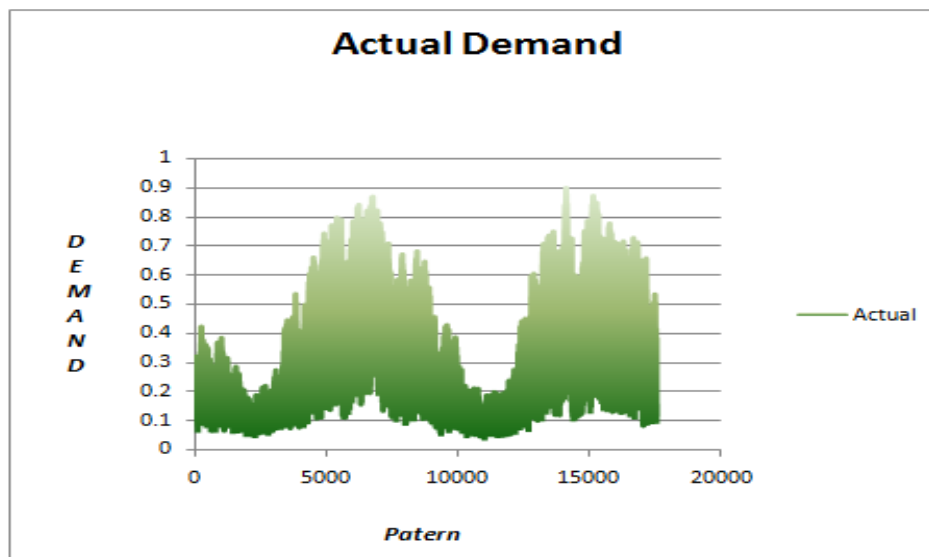


Γράφημα 5.6.5.2 Ποσοστό επιτυχίας στο δίκτυο με ανάδραση τύπου Jordan λαμβάνοντας υπόψη την προηγούμενη πρόβλεψη

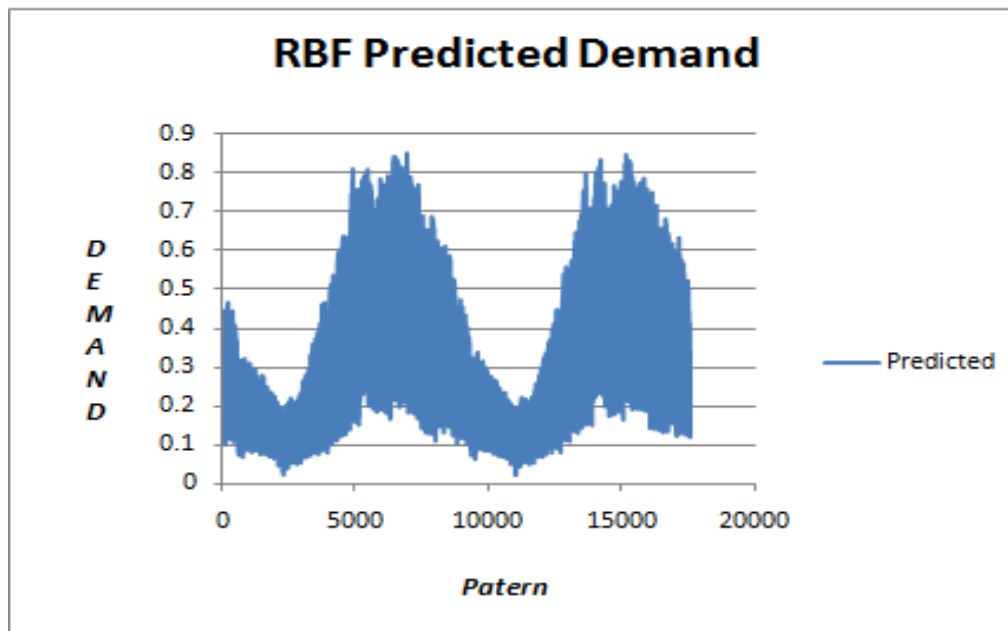
Από το πιο πάνω διάγραμμα (5.6.5.2) παρατηρούμε ότι το ποσοστό επιτυχίας μειώθηκε από 98.034% της υποενότητας 5.6.4 σε 94.013%. Το αποτέλεσμα αυτό ήταν αναμενόμενο αφού στην προηγούμενη πρόβλεψη εμπεριέχεται και κάποιο σφάλμα – έστω και μικρό- με αποτέλεσμα με την πάροδο του χρόνου το σφάλμα αυτό να αυξάνεται με αποτέλεσμα το ποσοστό επιτυχίας να ελαττωθεί.

5.7 Σύγκριση πραγματικού - επιθυμητού αποτελέσματος

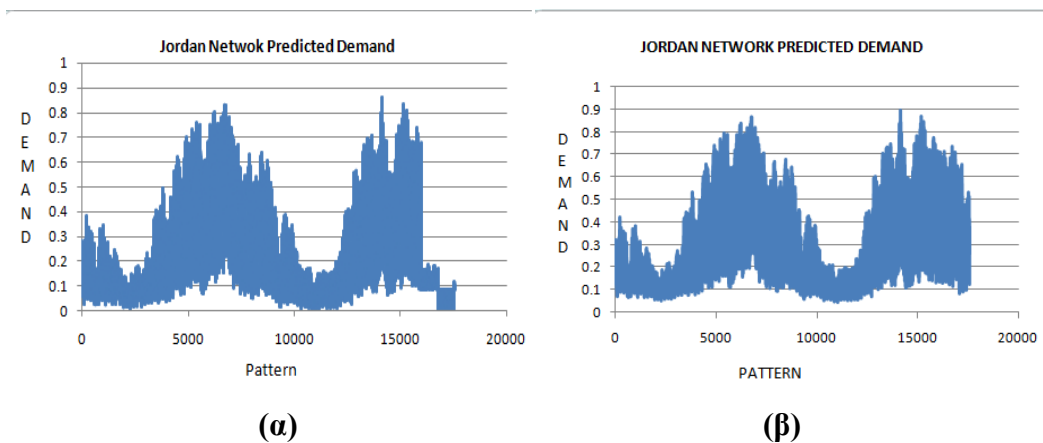
Αφού κάναμε τα πειράματα μας στις δυο μορφές νευρωνικών δικτύων βρήκαμε τις τιμές εκείνες των παραμέτρων οι οποίες μας δίνουν το καλύτερο δυνατό αποτέλεσμα. Έτσι εκπαιδεύσαμε τα δύο δίκτυα μ' αυτές τις τιμές και με το τέλος της εκπαίδευσης πήραμε την πρόβλεψη της ζήτησης για κάθε ένα στιγμιότυπο των δεδομένων ελέγχου. Με αυτόν τον τρόπο μπορέσαμε να συγκρίνουμε τη διαφορά του επιθυμητού από το πραγματικό αποτέλεσμα για τα δυο δίκτυα. Τα αποτελέσματα της σύγκρισης αυτής παρουσιάζονται στις γραφικές παραστάσεις 5.7.1-5.7.7 που ακολουθούν.



Γράφημα 5.7.1 Επιθυμητές τιμές ζήτησης ανά ώρα

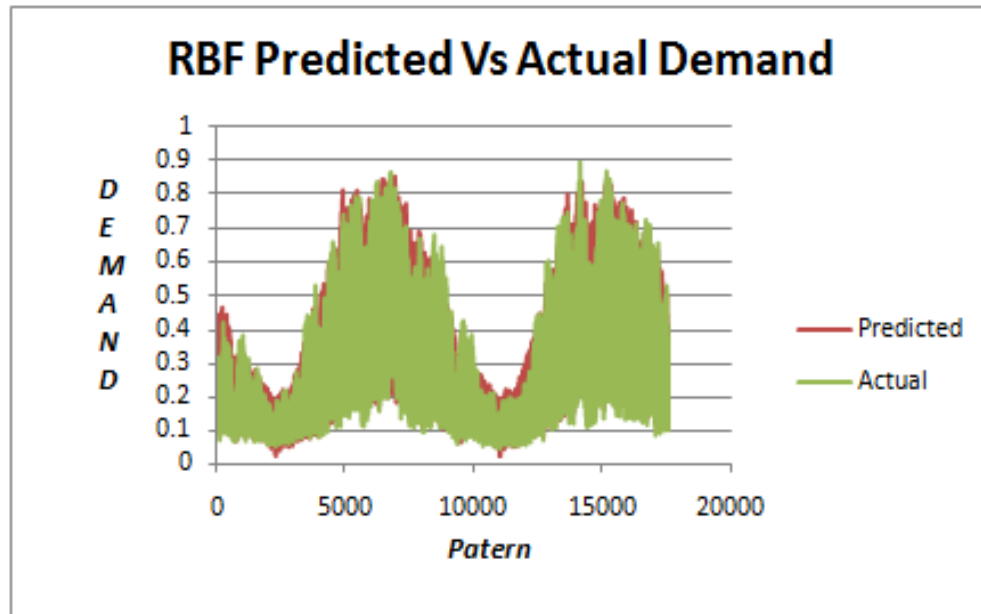


Γράφημα 5.7.2 Τιμές ζήτησης που προέκυψαν από το δίκτυο RBF ανά ώρα

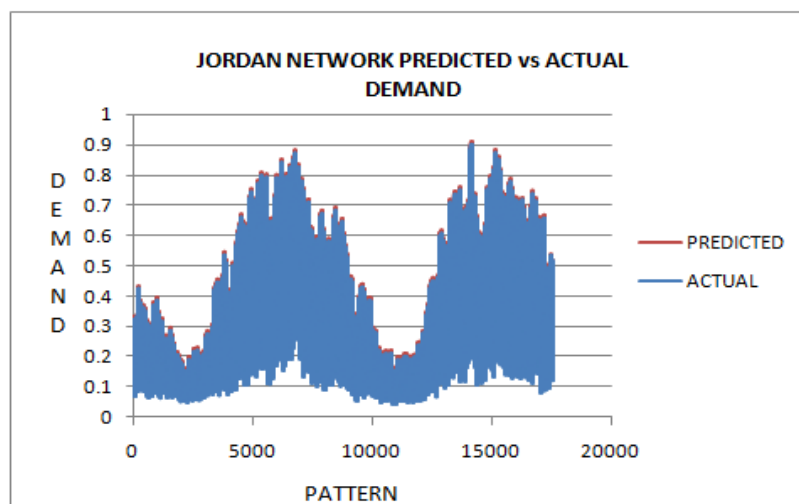


- (α) – Αποτελέσματα με την χρήση της προηγούμενης πρόβλεψης της ζήτησης
 (β) – Αποτελέσματα με την χρήση της προηγούμενης επιθυμητής τιμής ζήτησης

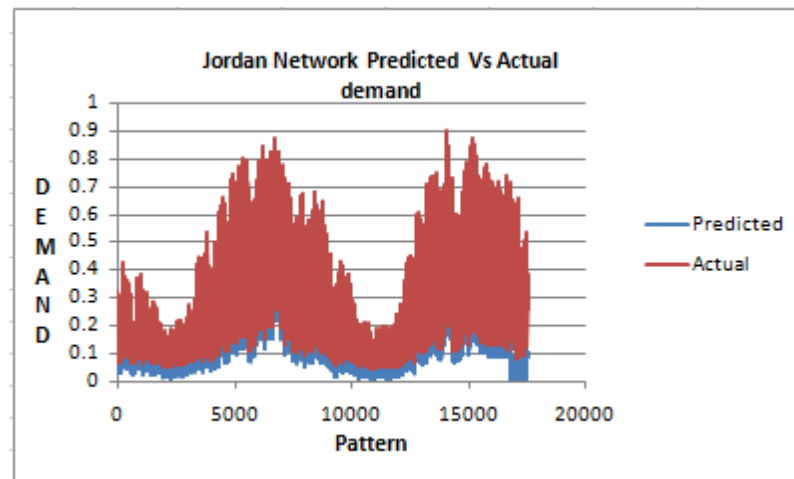
Γράφημα 5.7.3 Τιμές ζήτησης που προέκυψαν από το δίκτυο με ανάδραση τύπου Jordan



Γράφημα 5.7.4 Σύγκριση πραγματικού με το επιθυμητό αποτέλεσμα για δίκτυο RBF



(α)

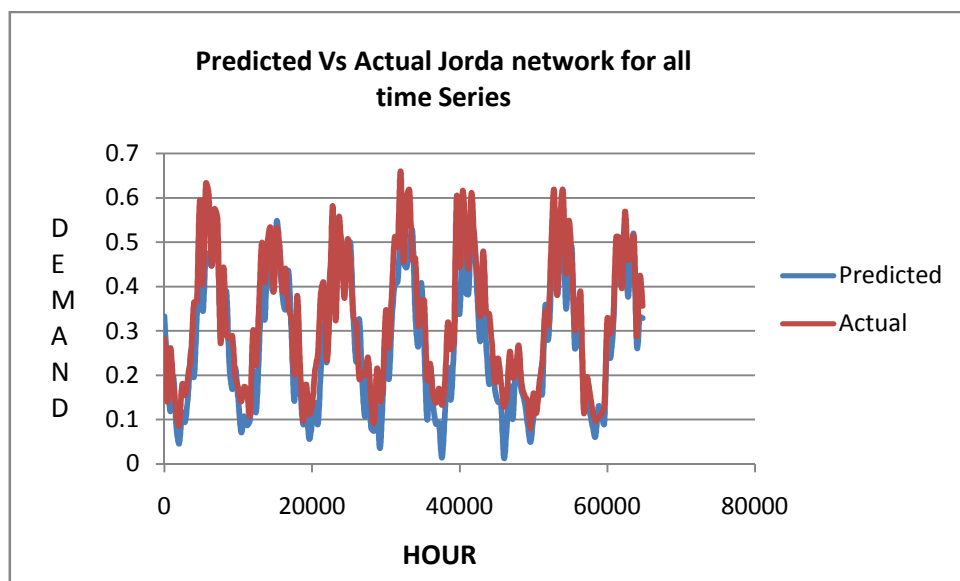


(β)

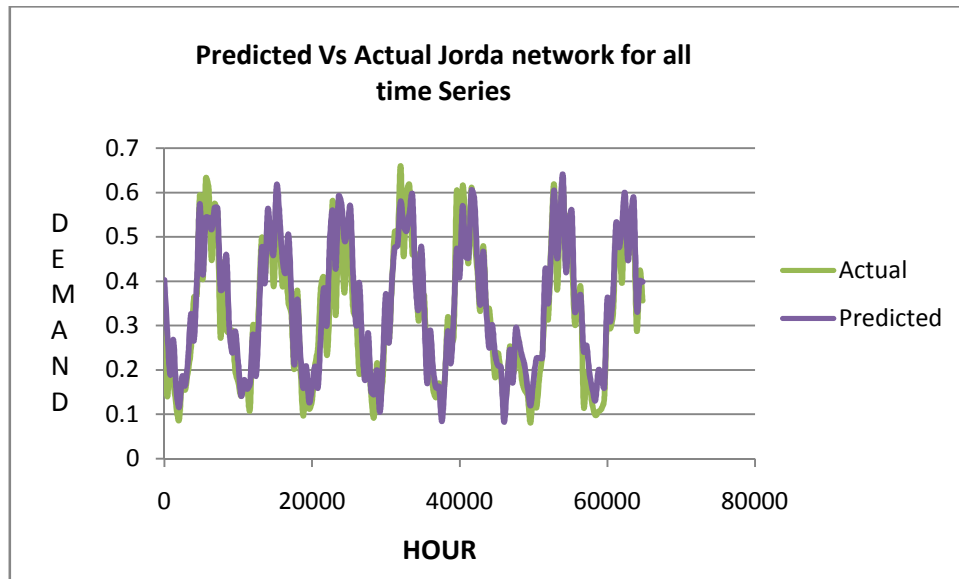
(α) – Επιθυμητό Vs πραγματικό αποτέλεσμα με την χρήση της προηγούμενης επιθυμητής τιμής ζήτησης

(β) – Επιθυμητό Vs πραγματικό αποτέλεσμα με την χρήση της προηγούμενης πρόβλεψης της ζήτησης

Γράφημα 5.7.5 Σύγκριση πραγματικού με το επιθυμητό αποτέλεσμα για δίκτυο με ανάδραση τύπου Jordan



(α)



(β)

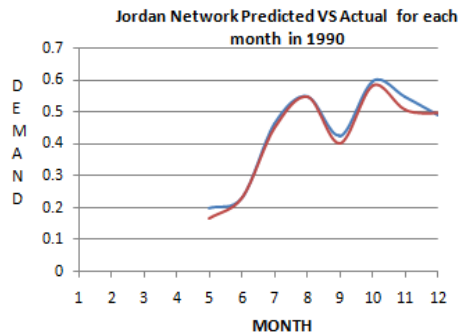
(α) – Επιθυμητό Vs πραγματικό αποτέλεσμα με την χρήση της προηγούμενης επιθυμητής τιμής ζήτησης

(β) – Επιθυμητό Vs πραγματικό αποτέλεσμα με την χρήση της προηγούμενης πρόβλεψης της ζήτησης

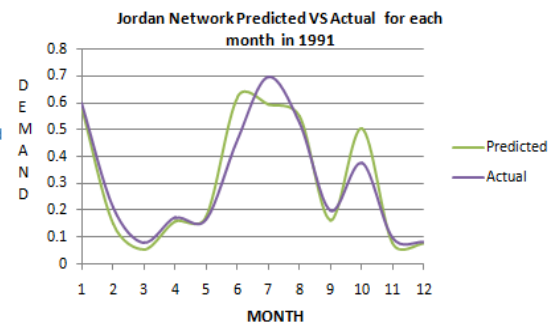
Γράφημα 5.7.6 Σύγκριση πραγματικού με το επιθυμητό αποτέλεσμα ανά ώρα για δίκτυο με ανάδραση τύπου Jordan

Από τις πιο πάνω γραφικές φαίνεται καθαρά ότι η προβλεπόμενη τιμή ζήτησης του δικτύου RBF είναι αρκετά κοντά στην πραγματική τιμή σε σύγκριση με το δίκτυο Jordan το οποίο έχει πολύ μικρή απόκλιση από την πραγματική τιμή. Αυτό δικαιολογείται από τ' αποτελέσματα που προέκυψαν αφού το RBF δίκτυο είχε ποσοστό επιτυχίας 85,057% σε αντίθεση με το δίκτυο Jordan που το ποσοστό επιτυχίας έφτασε το 98.034%, το οποίο ίσως να οφείλεται στο ότι δεν υπάρχουν απότομες διακυμάνσεις στα δεδομένα. Παρόλα αυτά και τα δύο δίκτυα έδωσαν ικανοποιητικά αποτελέσματα αφού οι προβλεπόμενες τιμές για τη ζήτηση είναι αρκετά κοντά στις επιθυμητές. Επιπλέον παρατηρούμε ότι στο δίκτυο τύπου Jordan όπου για να προβλέψουμε την επόμενη τιμή της ζήτησης χρησιμοποιούμε την προηγούμενη πρόβλεψη υπάρχει ελαφρώς μεγαλύτερη απόκλιση της πραγματικής με την επιθυμητή τιμής της ζήτησης σε σύγκριση με το δίκτυο Jordan όπου για να προβλέψουμε την επόμενη τιμή της ζήτησης χρησιμοποιούμε την προηγούμενη επιθυμητή τιμή ζήτησης.

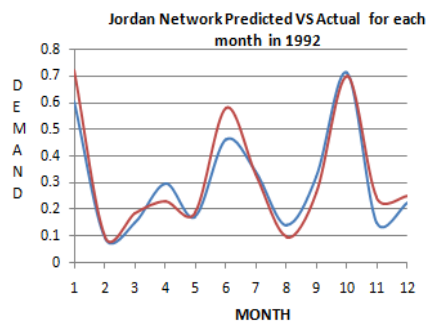
Από τα γραφήματα 5.7.5 και 5.7.6 δεν μπορούμε να βγάλουμε κάποια συγκεκριμένα συμπεράσματα σχετικά με την απόδοση του δικτύου μας για δεδομένους μήνες του χρόνου. Για το λόγο αυτό αναπαραστήσαμε τα γραφήματα του επιθυμητού με πραγματικό αποτέλεσμα ανά μήνα -και για τις δυο μορφές του δικτύου Jordan -για κάθε έτος. Τα αποτελέσματα παρουσιάζονται στα γραφήματα 5.7.7 και 5.7.8 που ακολουθούν.



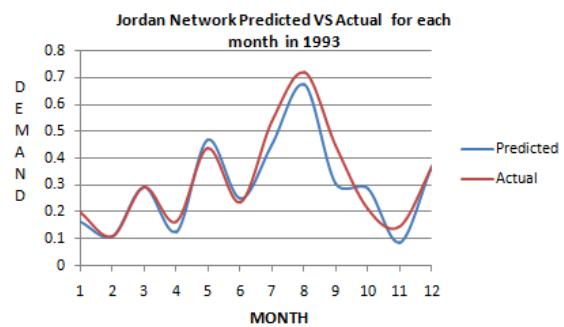
(α)



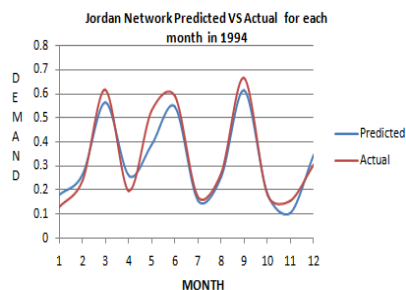
(β)



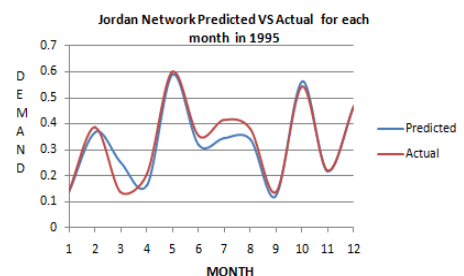
(γ)



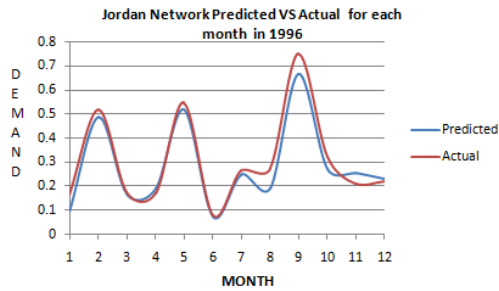
(δ)



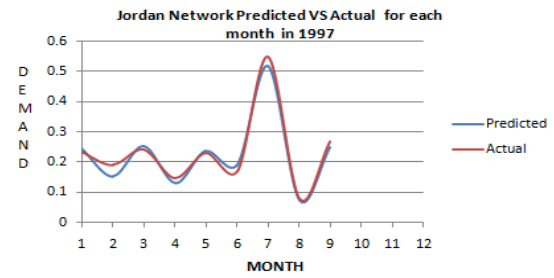
(ε)



(ζ)



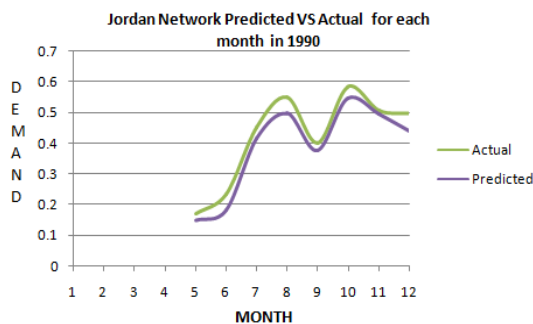
(η)



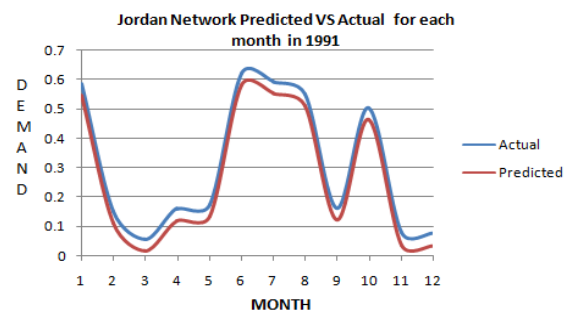
(θ)

- (α) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1990
- (β) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1991
- (γ) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1992
- (δ) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1993
- (ε) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1994
- (ζ) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1995
- (η) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1996
- (θ) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1997

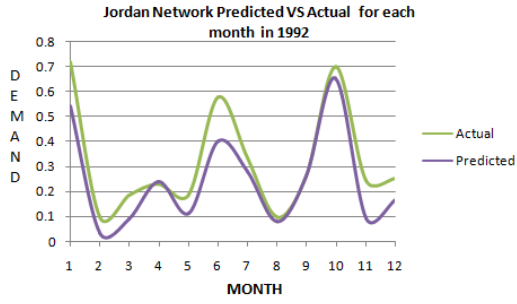
Γράφημα 5.7.7 Γραφήματα επιθυμητού - πραγματικού αποτελέσματος ανά έτος (δίκτυο Jordan) όπου για τον υπολογισμό της επόμενης τιμής ζήτησης χρησιμοποιούμε την προηγούμενη επιθυμητή τιμή



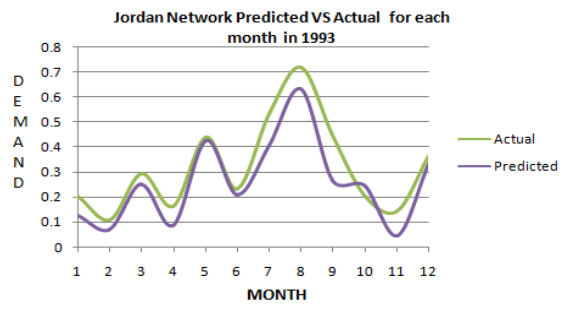
(α)



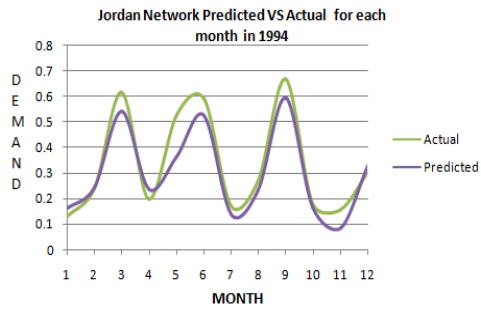
(β)



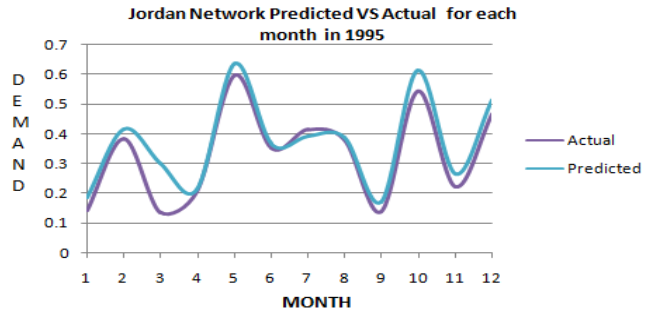
(γ)



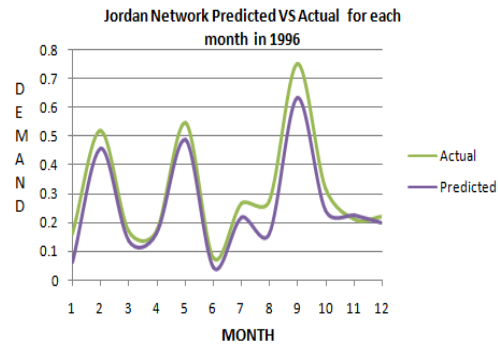
(δ)



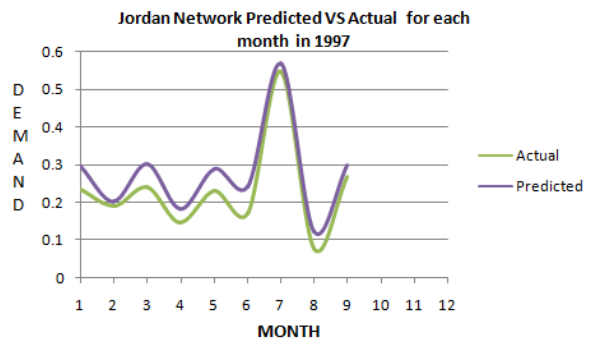
(ε)



(ζ)



(η)



(θ)

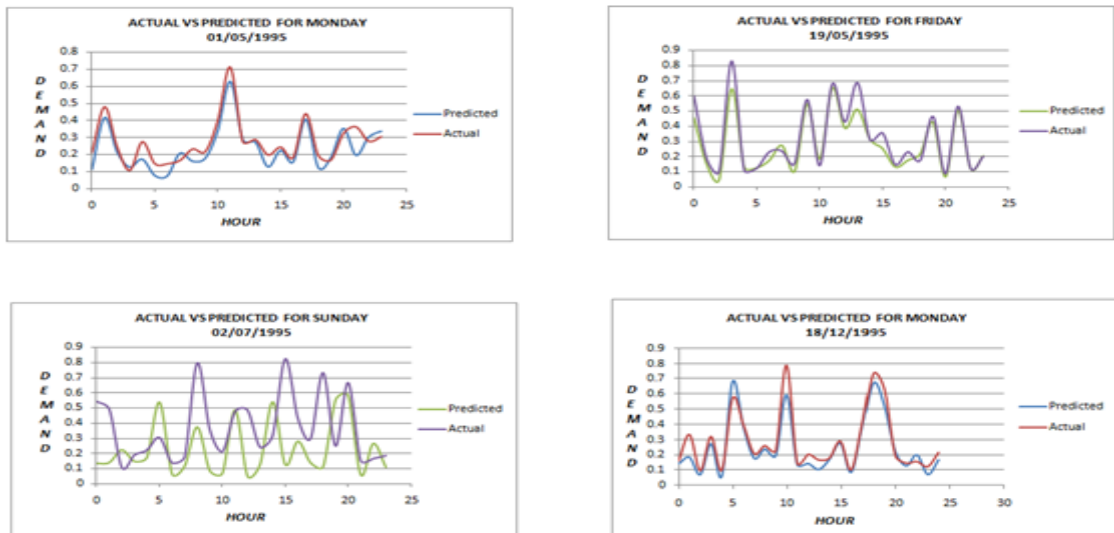
- (α) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1990
- (β) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1991
- (γ) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1992
- (δ) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1993
- (ε) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1994
- (ζ) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1995
- (η) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1996
- (θ) – Επιθυμητό Vs πραγματικό αποτέλεσμα για κάθε μήνα του 1997

**Γράφημα 5.7.8 Γραφήματα επιθυμητού - πραγματικού αποτελέσματος ανά έτος
(δίκτυο Jordan) όπου για τον υπολογισμό της επόμενης τιμής ζήτησης
χρησιμοποιούμε την προηγούμενη πρόβλεψη της ζήτησης**

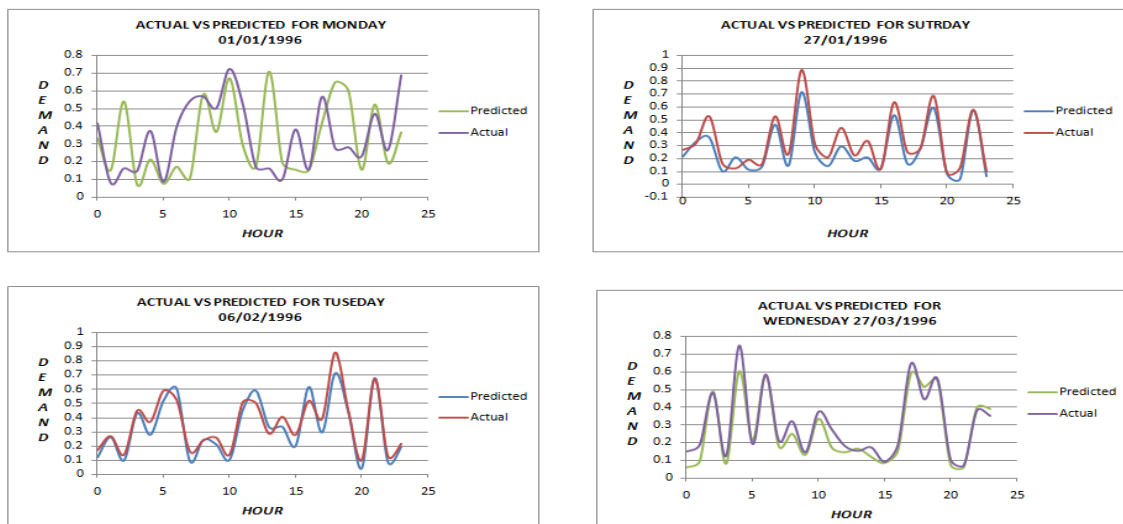
Από τα παραπάνω γραφήματα παρατηρούμε ότι τα πραγματικά αποτελέσματα είναι αρκετά κοντά στα επιθυμητά με εξαίρεση τον Απρίλιο και Μάιο του 1991, 1992 και 1994. Η διαφορά αυτή πολύ πιθανόν να οφείλεται στο ότι του μήνες αυτούς έχουμε μια αστάθεια στις καιρικές συνθήκες λόγω αλλαγής της εποχής, με αποτέλεσμα να έχουμε και μια πιο απότομη αυξομείωση της ζήτησης. Επιπρόσθετα, παρατηρούμε ότι υπάρχει μεγαλύτερη απόκλιση του πραγματικού με το επιθυμητό αποτέλεσμα στο δίκτυο Jordan όπου για τον υπολογισμό της επόμενης τιμής ζήτησης χρησιμοποιούμε την προηγούμενη πρόβλεψη της ζήτησης, πράγμα το οποίο είναι αναμενόμενο αφού το δίκτυο αυτό έδωσε ποσοστό επιτυχίας 94.013% σε σύγκριση με το 98.034% του δικτύου Jordan στο οποίο για τον υπολογισμό της επόμενης τιμής ζήτησης χρησιμοποιούμε την προηγούμενη επιθυμητή τιμή ζήτησης.

Στη συνέχεια θα μελετήσουμε την απόδοση του δυναμικού δικτύου σε μεμονωμένες μέρες. Συγκεκριμένα από τις χρονιές 1995, 1996 και 1997 πήραμε μεμονωμένα τέσσερις μέρες από την κάθε μια και δημιουργήσαμε τα γραφήματα του επιθυμητού και πραγματικού αποτελέσματος. Προσπαθήσαμε να πάρουμε τόσο εργάσιμες όσο και μη εργάσιμες μέρες έτσι ώστε να δούμε πως ανταποκρίνεται το δίκτυο μας στις συνθήκες αυτές. Τα αποτελέσματα για το δυναμικό δίκτυο όπου για τον υπολογισμό της επόμενης τιμής ζήτησης χρησιμοποιούμε την προηγούμενη επιθυμητή τιμή ζήτησης παρουσιάζονται στα γραφήματα 5.7.9 - 5.7.11 ενώ τα αντίστοιχα αποτελέσματα για τις

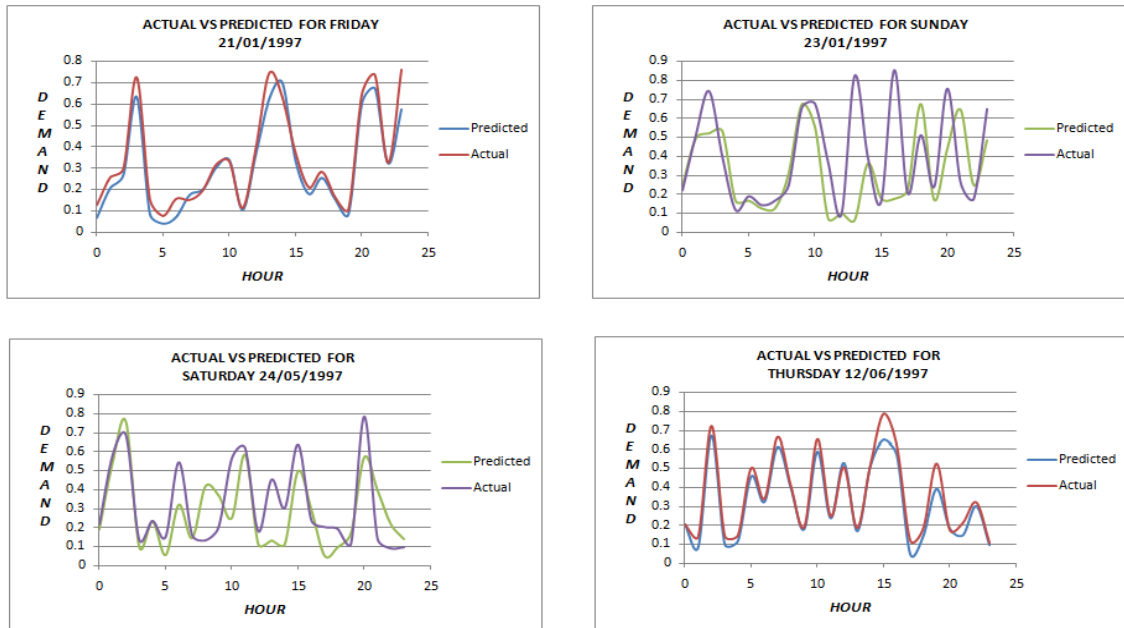
ίδιες μέρες για το δίκτυο Jordan στο οποίο για τον υπολογισμό της επόμενης τιμής ζήτησης χρησιμοποιούμε την προηγούμενη πρόβλεψη παρουσιάζονται στα γραφήματα 5.7.12-5.7.14.



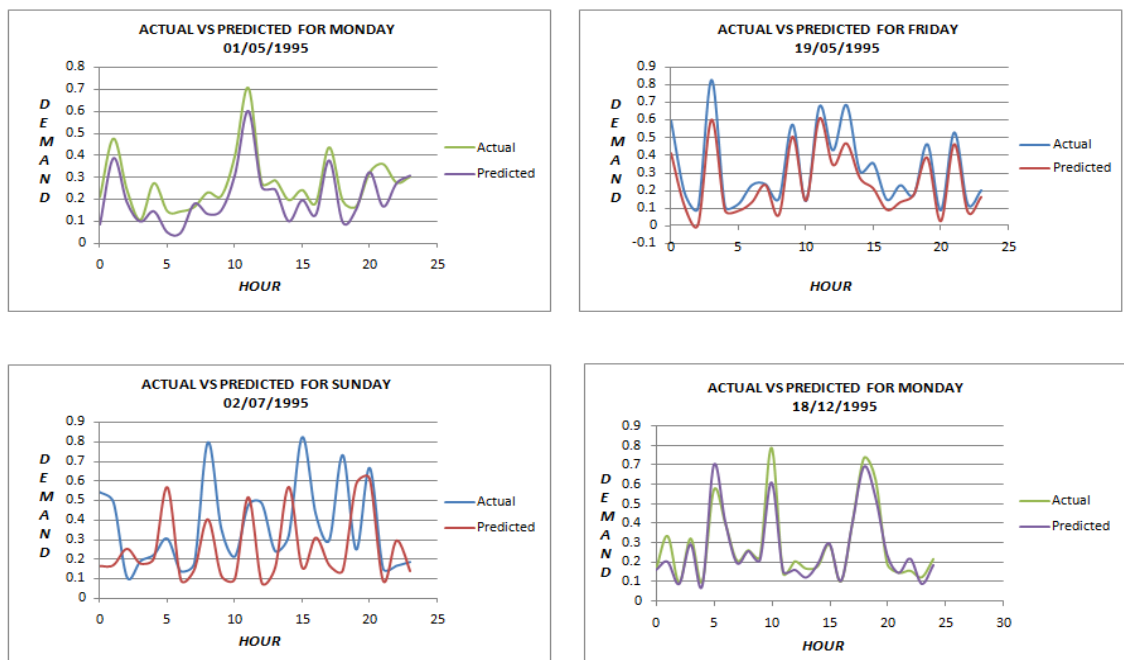
Γράφημα 5.7.9 Γραφήματα επιθυμητού - πραγματικού αποτελέσματος ανά ώρα για τέσσερις μεμονωμένες μέρες του 1995 (δίκτυο Jordan – χρήση προηγούμενης επιθυμητής τιμής ζήτησης)



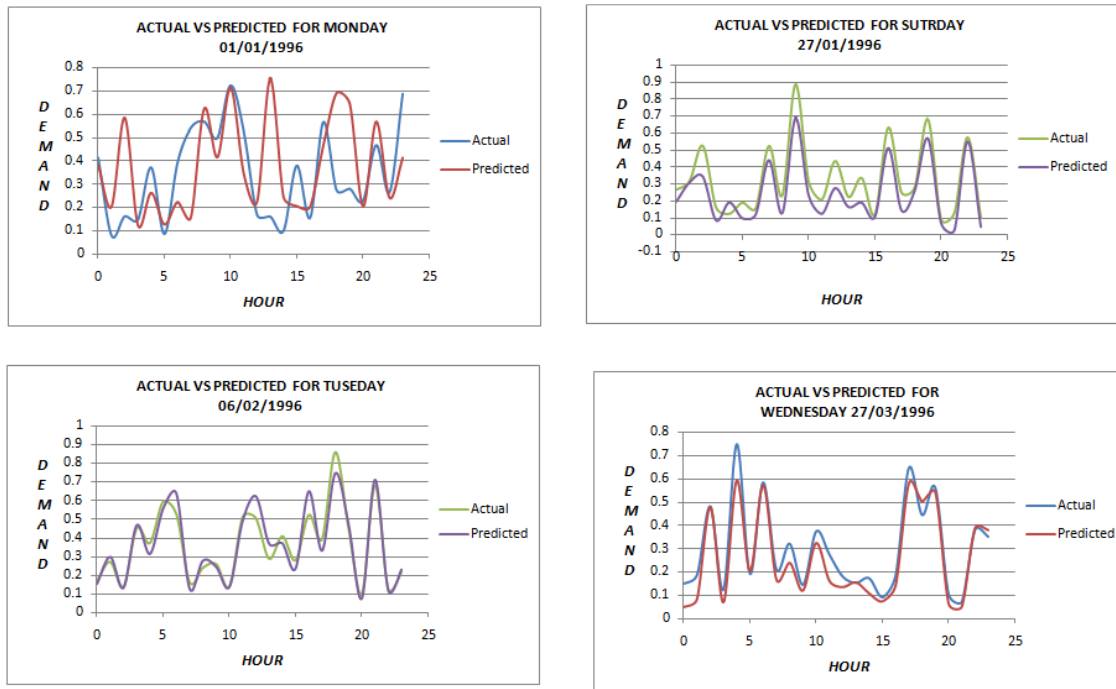
Γράφημα 5.7.10 Γραφήματα επιθυμητού - πραγματικού αποτελέσματος ανά ώρα για τέσσερις μεμονωμένες μέρες του 1996 (δίκτυο Jordan – χρήση προηγούμενης επιθυμητής τιμής ζήτησης)



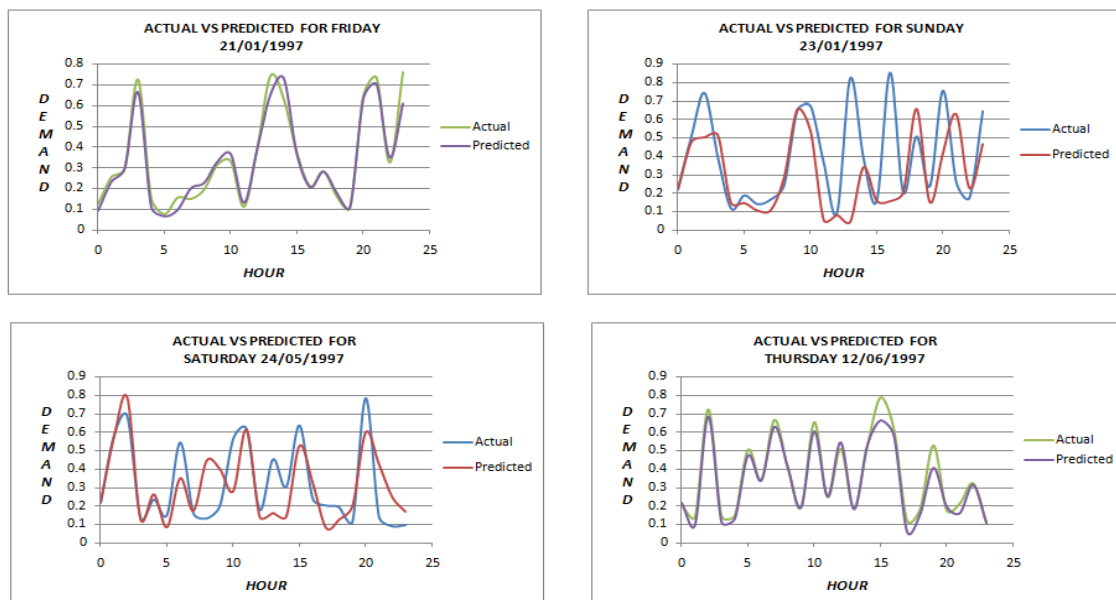
Γράφημα 5.7.11 Γραφήματα επιθυμητού με πραγματικού αποτελέσματος ανά ώρα για τέσσερις μεμονωμένες μέρες του 1997 (δίκτυο Jordan – χρήση προηγούμενης επιθυμητής τιμής ζήτησης)



Γράφημα 5.7.12 Γραφήματα επιθυμητού - πραγματικού αποτελέσματος ανά ώρα για τέσσερις μεμονωμένες μέρες του 1995 (δίκτυο Jordan – χρήση προηγούμενης πρόβλεψης της ζήτησης)



Γράφημα 5.7.13 Γραφήματα επιθυμητού - πραγματικού αποτελέσματος ανά ώρα για τέσσερις μεμονωμένες μέρες του 1996 (δίκτυο Jordan – χρήση προηγούμενης πρόβλεψης της ζήτησης)



Γράφημα 5.7.14 Γραφήματα επιθυμητού - πραγματικού αποτελέσματος ανά ώρα για τέσσερις μεμονωμένες μέρες του 1997 (δίκτυο Jordan – χρήση προηγούμενης πρόβλεψης της ζήτησης)

Από τα γραφήματα 5.7.9 – 5.7.14 παρατηρούμε ότι έχουμε μεγαλύτερη απόκλιση του επιθυμητού από το πραγματικό αποτέλεσμα στις μη εργάσιμες μέρες σε σύγκριση με τις εργάσιμες. Πιο συγκεκριμένα μεγαλύτερη απόκλιση έχουμε την Κυριακή 02/07/1995, την Πρωτοχρονιά του 1996, την Κυριακή 23/01/1997 και το Σάββατο 24/05/1995. Επίσης το Σάββατο 27/01/1996 παρατηρούμε ότι πάλι υπάρχει μια απόκλιση όμως είναι κατά πολύ πιο μικρή σε σύγκριση με τις υπόλοιπες. Τα αποτελέσματα έρχονται να ενισχύσουν την άποψη μας ότι οι μη εργάσιμες μέρες επηρεάζουν την πρόβλεψη της ζήτησης.

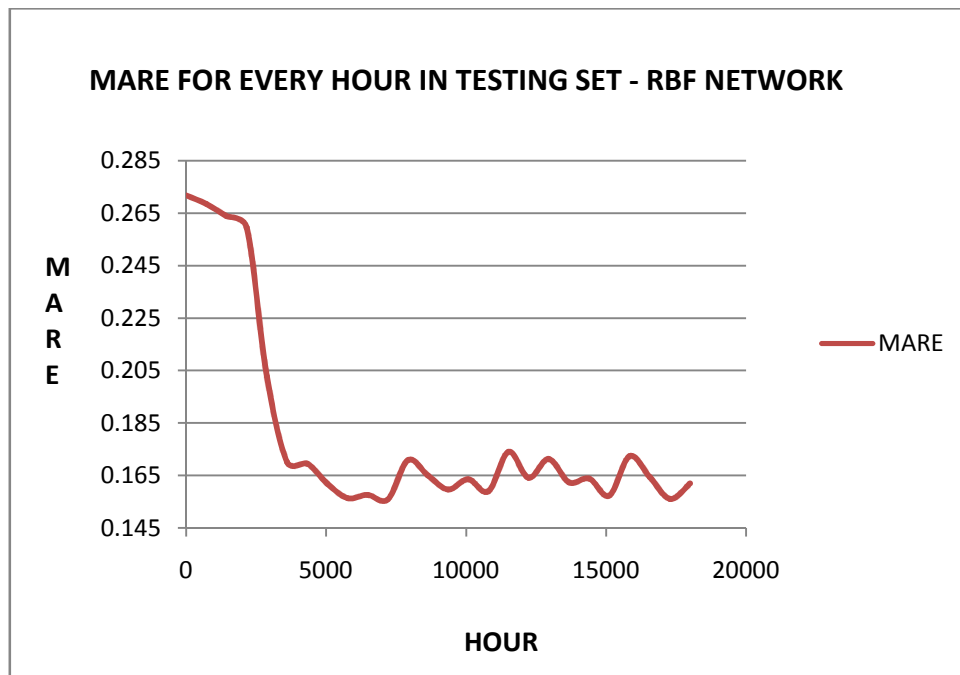
5.8 Χρήση του Μέσου Απόλυτου Σχετικού Σφάλματος (MARE) για τον υπολογισμό του σφάλματος

Στις προηγούμενες ενότητες όπως αναφέραμε για τον υπολογισμό του σφάλματος χρησιμοποιήθηκε το μέσο τετραγωνικό σφάλμα (MSE – κεφάλαιο 2). Στην ενότητα αυτή θα παραθέσουμε τα αποτελέσματα χρησιμοποιώντας μια άλλη τεχνική υπολογισμού του σφάλματος που ονομάζεται μέσο απόλυτο σχετικό σφάλμα (*Mean Absolute Relative Error*). Για τον υπολογισμό του σφάλματος αυτού χρησιμοποιήσαμε την εξίσωση 26 που ακολουθεί.

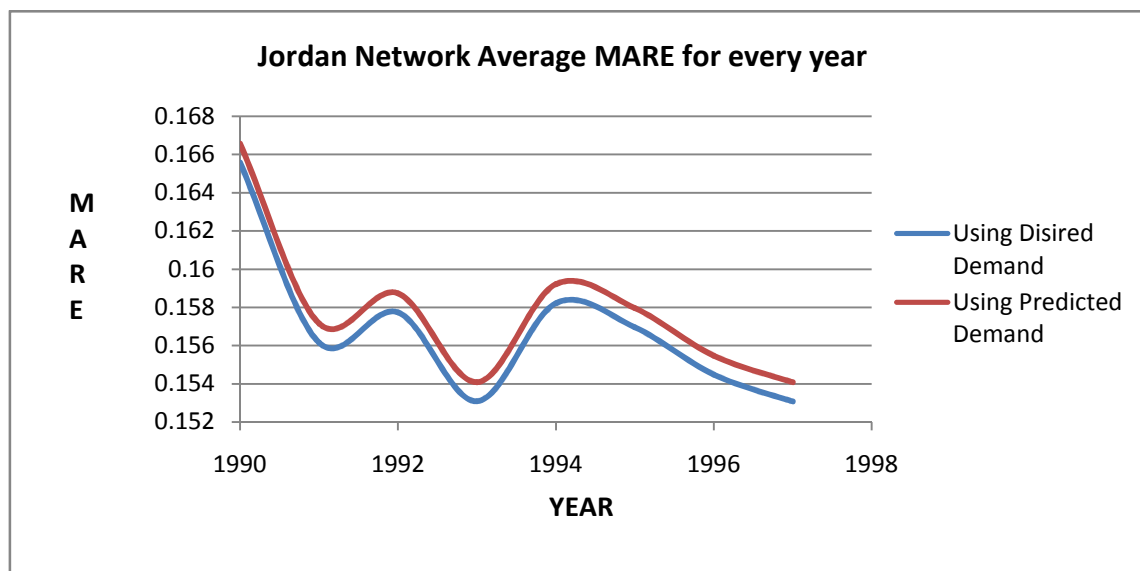
$$MARE(n) = \frac{1}{n} \sum_{i=1}^n \frac{|d(i)-p(i)|}{d(i)} \quad (26)$$

- n – το πλήθος των προβλέψεων
- $d(i)$ – η πραγματική τιμή του δείγματος i
- $p(i)$ – η προβλεπόμενη τιμή του δείγματος i

Αφού εκπαιδεύσαμε τα δυο δίκτυα –στατικό και δυναμικό- υπολογίζοντας το σφάλμα με τη χρήση **μέσου απόλυτου σχετικού σφάλματος** και τις **παραμέτρους αυτές που έδωσαν τα καλύτερα αποτελέσματα** παρουσιάζουμε τα γραφήματα του σφάλματος για το δίκτυο RBF (Γράφημα 5.8.1) και το δίκτυο με ανάδραση τύπου Jordan (Γράφημα 5.8.2).

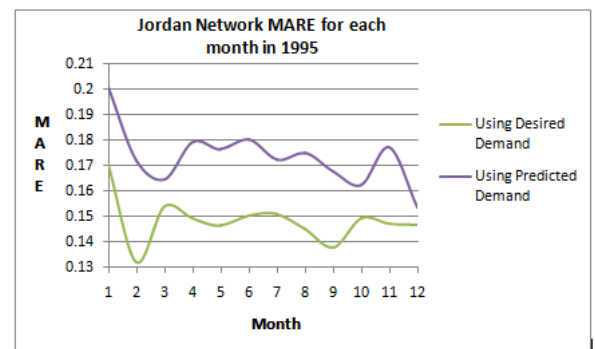
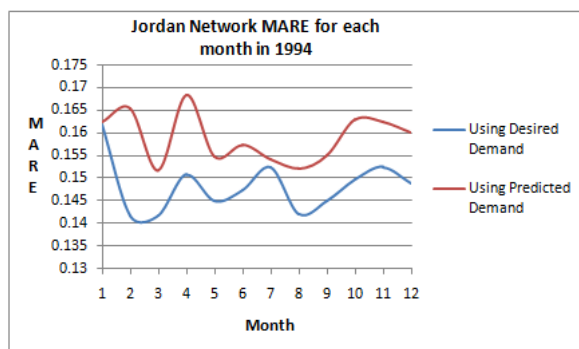
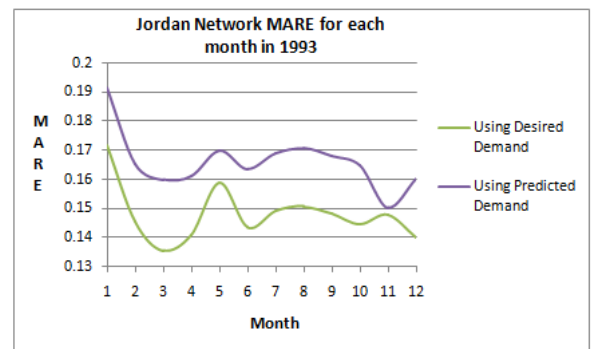
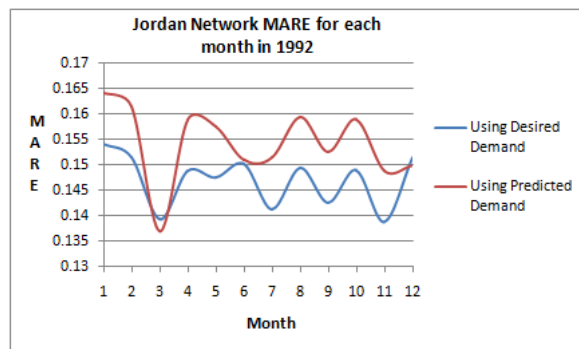
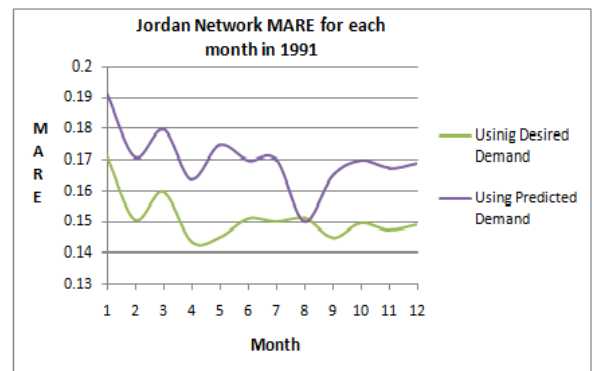
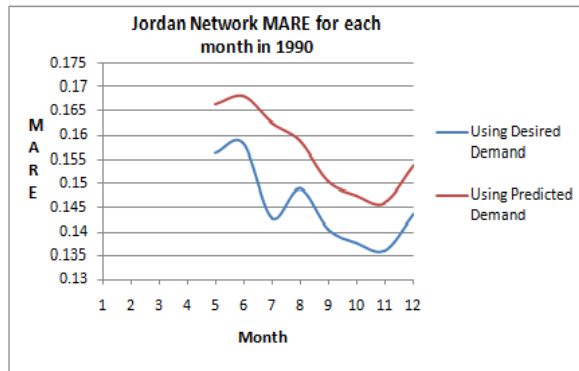


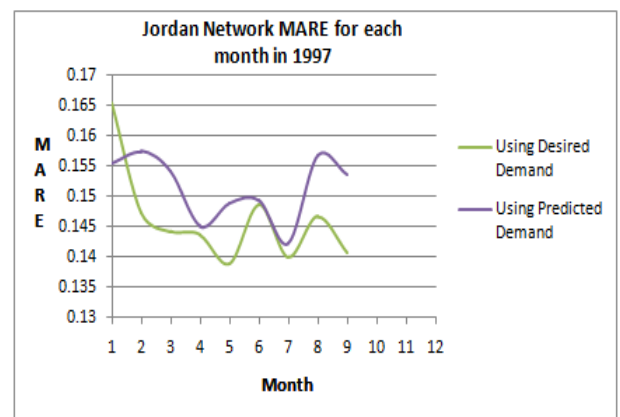
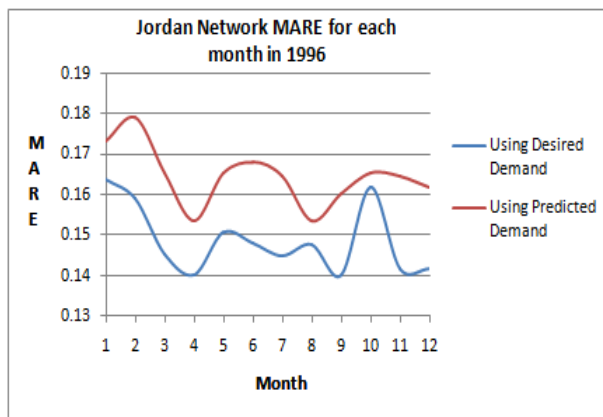
Γράφημα 5.8.1 Mean Absolute Relative Error ανά ώρα για δίκτυο RBF



Γράφημα 5.8.2 Mean Absolute Relative Error ανά έτος για δίκτυο Jordan (με τις δυο τεχνικές που χρησιμοποιήθηκαν)

Από το γράφημα 5.8.2 δεν μπορούμε να καταλήξουμε σε κάποιο συμπέρασμα όσο αφορά τη διακύμανση του σφάλματος. Για το λόγο αυτό υπολογίσαμε το Μέσο Απόλυτο Σχετικό Σφάλμα για κάθε μήνα κάθε έτους και τα αποτελέσματα παρουσιάζονται στο γράφημα 5.8.3 που ακολουθεί.





Γράφημα 5.8.3 Γραφήματα Μέσου Απόλυτου Σχετικού Σφάλματος για κάθε μήνα από 02/05/1990 μέχρι 30/09/1997 (δίκτυο Jordan και για τις δυο μεθόδους για πρόβλεψη της επόμενης τιμής ζήτησης)

Από τα πιο πάνω γραφήματα παρατηρούμε ότι υπάρχουν αυξομειώσεις του μέσου απόλυτου σχετικού σφάλματος. Συγκεκριμένα, σε κάποιους μήνες (π.χ. τον Οκτώβριο του 1996) το μέσο απόλυτο σχετικό σφάλμα είναι μεγαλύτερο σε σύγκριση με τους υπόλοιπους μήνες του χρόνου. Το γεγονός αυτό υποδηλώνει ότι στους συγκεκριμένους μήνες είχαμε απότομες αλλαγές των καιρικών συνθηκών οι οποίες συνεπάγονται και απότομες αλλαγές στη ζήτηση και συνεπώς επηρεάζεται και η πρόβλεψη της.

Επιπλέον, σχεδόν σε όλους του μήνες και για τα επτά χρόνια το μέσο απόλυτο σφάλμα για το δίκτυο Jordan όπου λαμβάνουμε υπόψη την προηγούμενη πρόβλεψη και όχι την επιθυμητή τιμή της ζήτησης, είναι μεγαλύτερο από αυτό που πήραμε στο δίκτυο Jordan στο οποίο χρησιμοποιούμε την προηγούμενη επιθυμητή τιμή της ζήτησης. Αυτό είναι αναμενόμενο αφού στην προηγούμενη πρόβλεψη εμπεριέχεται και κάποιο σφάλμα – έστω και μικρό- με αποτέλεσμα με την πάροδο του χρόνου το σφάλμα αυτό να αυξάνεται.

Κεφάλαιο 6

Συμπεράσματα

6.1 Συμπεράσματα	123
6.2 Μελλοντική Εργασία	127

6.1 Συμπεράσματα

Η πρόβλεψη της ζήτησης του φυσικού αερίου όπως αναφέραμε και στην εισαγωγή είναι καθοριστική και αναγκαία για τις εταιρίες παροχής του, αφού με την πρόβλεψη της ζήτησης θα μειώσουν τα έξοδα τους αυξάνοντας παράλληλα τα έσοδα τους. Όμως για να μπορέσει κάποιος να προβλέψει με αρκετή ακρίβεια τη ζήτηση θα πρέπει να λάβει υπόψη όλους τους παράγοντες που ίσως να επηρεάζουν τη ζήτηση. Συγκεκριμένα θα πρέπει να μελετήσει το μήνα για τον οποίο επιθυμεί να κάνει πρόβλεψη, την ώρα της ημέρας, αν η συγκεκριμένη μέρα είναι εργάσιμη ή όχι, την ταχύτητα των ανέμων, τη θερμοκρασία καθώς επίσης και οποιοδήποτε άλλο παράγοντα που κρίνει ότι μπορεί να επηρεάσει τη ζήτηση.

Έτσι αφού βασιστήκαμε στους πιο κύριους παράγοντες που επηρεάζουν τη ζήτηση καταφέραμε να δημιουργήσουμε ένα εργαλείο (Gas Prediction Software) με τη χρήση του οποίου μπορεί να γίνει πρόβλεψη της ζήτησης του φυσικού αερίου. Το συγκεκριμένο εργαλείο είναι *φιλικό προς το χρήστη* καθώς επίσης εύκολα μπορεί να *επεκταθεί* με την πρόσθεση επιπλέον τεχνικών πρόβλεψης.

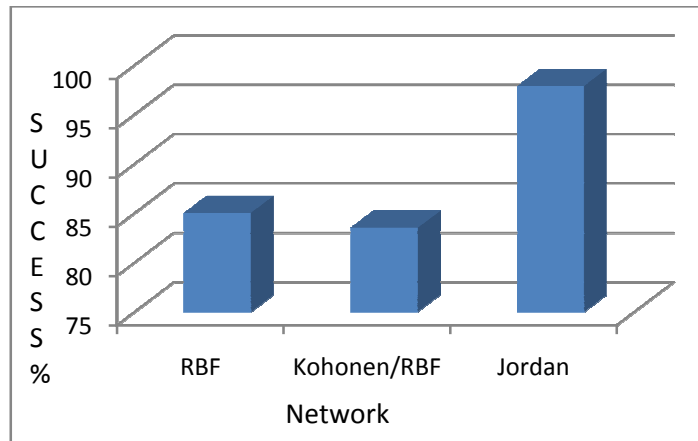
Έτσι στη διπλωματική αυτή εργασία προσπαθήσαμε να προσεγγίσουμε το συγκεκριμένο πρόβλημα με τη χρήση νευρωνικών δικτύων. Συγκεκριμένα χρησιμοποιήσαμε δύο κύριες αρχιτεκτονικές νευρωνικών δικτύων, τα δίκτυα αξονικών βάσεων (RBF) και ένα δίκτυο με ανάδραση από την έξοδο τύπου Jordan. Στην πρώτη αρχιτεκτονική χρησιμοποιήσαμε δύο παραλλαγές των δικτύων RBF[7][8][9]. Στην πρώτη παραλλαγή οι αρχικές θέσεις των κέντρων ήταν τυχαίες μέσα από το σύνολο δεδομένων εκπαίδευσης ενώ στη δεύτερη παραλλαγή οι αρχικές θέσεις καθώς και το πλήθος του προέρχονταν από τη χρήση αλγορίθμου ομαδοποίησης και συγκεκριμένα χάρτες αυτό οργάνωσης (Kohonen SOM [11]). Στη δεύτερη αρχιτεκτονική χρησιμοποιήσαμε ένα δίκτυο πολλαπλών στρωμάτων perceptron με διαφορά ότι η προηγούμενη έξοδος ήταν η είσοδος στην επόμενη χρονική στιγμή. Το δίκτυο αυτό εκπαιδεύτηκε με τον αλγόριθμο ανάστροφης μετάδοσης του λάθους στο χρόνο (back propagation through time - BPTT [14]).

Τα νευρωνικά δίκτυα μπορούν να παρομοιαστούν ως ένα μαύρο κουτί το οποίο δέχεται κάποιες εισόδους και μας δίνει κάποια έξοδο. Η έξοδος αυτή εκτός από τις εισόδους επηρεάζεται σε αρκετό βαθμό και από τις παραμέτρους της συγκεκριμένης αρχιτεκτονικής. Αφού στηριχθήκαμε στο πιο πάνω προσπαθήσαμε να βρούμε εκείνες τις τιμές οι οποίες μας δίνουν το βέλτιστο αποτέλεσμα. Για να γίνει αυτό απαιτούσε αρκετούς πειραματισμούς πάνω στα δίκτυα οι οποίοι περιγράφονται στο κεφάλαιο 5.

Στον πίνακα 6.1.1 παρουσιάζονται οι τιμές των παραμέτρων με τις οποίες επιτεύχθηκαν τα υψηλότερα ποσοστά επιτυχίας.

Παράμετρος	RBF – Random Centers	Kohonen SOM - RBF	Jordan Network	Jordan Network
Learning Rate	0.5	0.1	0.1	0.1
Number of Centers	10	35 με μέγεθος του πλέγματος 20x20 για τον αλγόριθμο Kohonen SOM	-	-
Number of Hidden Neurons	-	-	16	16
Number of Time Steps	-	-	10	10
Time Constant	-	-	1.2	1.2
Method			Χρήση προηγούμενης επιθυμητής τιμής για πρόβλεψη της επόμενης τιμής ζήτησης	Χρήση προηγούμενης πρόβλεψης για πρόβλεψη της επόμενης τιμής ζήτησης
Total Success (%)	85.057	83.56	98.034	94.013

Πίνακας 6.1.1 Συνοπτικός πίνακας αποτελεσμάτων



Γράφημα 6.1.1 Τελικά ποσοστά επιτυχίας

Όπως φαίνεται από το παραπάνω γράφημα και τον πίνακα 6.1.1 έχουμε και για τις τρεις μορφές δικτύων αρκετά ικανοποιητικά αποτελέσματα. Παρατηρούμε ότι και στις δύο περιπτώσεις (δίκτυο συναρτήσεων αξονικών βάσεων και στο υβριδικό δίκτυο) το ποσοστό επιτυχίας κυμαίνεται γύρω στο 85% με το υψηλότερο ποσοστό να παρουσιάζεται στο δίκτυο με ανατροφοδότηση τύπου Jordan και ανέρχεται στο 98.034%.

Η επιτυχία του δικτύου Jordan ίσως να οφείλεται σε δύο κύριους λόγους. Ο πρώτος λόγος είναι ότι τα δεδομένα είναι ομαλά δηλαδή δεν έχουμε απότομες διακυμάνσεις της θερμοκρασίας ή της ταχύτητας των ανέμων και συνεπώς της ζήτησης. Ο δεύτερος λόγος βρίσκεται στη θεωρία αφού γνωρίζουμε ότι οι είσοδοι στα νευρωνικά δίκτυα δεν πρέπει να έχουν αλληλεξαρτήσεις μεταξύ τους πράγμα που συμβαίνει στο στατικό δίκτυο αφού η μέση ζήτηση της προηγούμενης εβδομάδας εξαρτάται από τη θερμοκρασία και την ταχύτητα των ανέμων καθώς επίσης και η θερμοκρασία εξαρτάται από το μήνα. Το γεγονός αυτό ίσως δημιουργεί αυτή τη μεγάλη διαφορά στα ποσοστά επιτυχίας των δύο δικτύων – στατικό και δυναμικό-.

Τα πιο πάνω αποτελέσματα αφορούν το δίκτυο με ανατροφοδότηση τύπου Jordan όπου για την πρόβλεψη της ζήτησης τη χρονική στιγμή $N+1$, χρησιμοποιούσαμε την προηγούμενη επιθυμητή τιμή της ζήτησης και όχι αυτή που είχε προβλεφθεί. Όταν κάναμε τις κατάλληλες τροποποιήσεις έτσι ώστε για την πρόβλεψη της επόμενης τιμής ζήτησης να χρησιμοποιούμε αυτή που είχε προβλεφθεί το ποσοστό αυτό μειώθηκε σε 94.013%.

Αυτό δικαιολογείται με το γεγονός ότι, στην προηγούμενη πρόβλεψη εμπεριέχεται και κάποιο σφάλμα – έστω και μικρό- με αποτέλεσμα με την πάροδο του χρόνου το σφάλμα αυτό να αυξάνεται και το ποσοστό επιτυχίας να ελαττωθεί.

Τα αποτελέσματα που παρουσιάζονται στην αναφορά αυτή όσο αφορά το στατικό δίκτυο [7][8][9] – δίκτυο συναρτήσεων αξονικών βάσεων RBF – κυμαίνονται στα ίδια επίπεδα με την προηγούμενη διπλωματική εργασία [4] όπου χρησιμοποιήθηκε ένα δίκτυο πολλαπλών στρωμάτων Perceptron. Αντίθετα το δίκτυο με ανάδραση τύπου Jordan έδωσε καλύτερα αποτελέσματα σε σύγκριση με την τεχνική του κινητού παραθύρου της προηγούμενης εργασίας [4].

Στον πίνακα 6.1.2 που ακολουθεί παρουσιάζονται συγκριτικά τα ποσοστά επιτυχίας μεταξύ των δύο διπλωματικών εργασιών καθώς και της έρευνας που έγινε από την εταιρία παροχής φυσικού αερίου από όπου προέρχονται και τα δεδομένα που χρησιμοποιήθηκαν.

Έρευνα	Στατικό Δίκτυο	Δίκτυο με χρήση χρονοσειράς
Υφιστάμενη Έρευνα	85.057 (RBF)	98.034 (Jordan Recurrent Network)
Προηγούμενη Έρευνα	85.90 (MLP)	95.90 (Moving Window)
Έρευνα Βρετανικής εταιρίας παροχής φυσικού αερίου	89.68 (MPL)	-

Πίνακας 6.1.2 Συγκριτικός πίνακας ποσοστών επιτυχίας (%), στα δεδομένα επαλήθευσης στα δίκτυα της υφιστάμενης, της προηγούμενης [4] διπλωματικής εργασίας και της έρευνας Βρετανικής εταιρίας παροχής φυσικού αερίου

Όπως φαίνεται και από τα αποτελέσματα, τα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν με επιτυχία για το πρόβλημα της πρόβλεψης της ζήτησης του φυσικού αερίου.

6.2 Μελλοντική Εργασία

Σχετικά με τη μελλοντική εργασία υπάρχουν αρκετές προσεγγίσεις και πειραματισμοί που μπορούν να εφαρμοστούν για τη βελτίωση του αποτελέσματος. Συγκεκριμένα μπορούν να χρησιμοποιηθούν **γενετικοί αλγόριθμοι** για να βρεθούν οι βέλτιστες τιμές των παραμέτρων των δικτύων που θα δώσουν τα καλύτερα αποτελέσματα.

Επιπλέον μπορεί να διαφοροποιηθούν οι είσοδοι για το πρόβλημα. Αυτό μπορεί να γίνει με την **αλλαγή της κωδικοποίησης των δεδομένων**. Για παράδειγμα θα μπορούσαμε ν' αφαιρέσουμε από τα δεδομένα όλες τις αργίες και να εκπαιδευτεί το δίκτυο μόνο με την χρήση των εργάσιμων ημερών ή και το αντίθετο συγκρίνοντας παράλληλα τα αποτελέσματα.

Επίσης, μπορούμε να χρησιμοποιήσουμε άλλες αρχιτεκτονικές δικτύων όπως **support vector machines**, **νευρωνικά δίκτυα καθυστέρησης χρόνου** (time delay neural networks – TDNNs) καθώς και κάποιο **δίκτυο ανατροφοδότησης τύπου Elman** όπου η ανατροφοδότηση προς τα πίσω γίνεται από το κρυφό επίπεδο.

Τέλος, στη προσπάθεια βελτίωσης των αποτελεσμάτων μπορούν να χρησιμοποιηθούν **Network Ensembles** οι οποίες είναι συλλογές ΝΔ οι οποίες εκπαιδεύονται για την επίλυση του ιδίου προβλήματος.

Βιβλιογραφία

- [1] Nguyen Hoang Viet and Jacek Mandziuk, “Neural and Fuzzy Neural Networks in Prediction of Natural Gas Consumption”, Neural Parallel & Scientific Computations, Pages : 265-286 vol. 13, 2005.
- [2] Petr Musilek, Emil Pelikan, Tomas Brabec and Milan Simunek, “Recurrent Neural Network Based Gating for Natural Gas Load Prediction System”, International Joint Conference on Neural Networks, July 16-21 2006.
- [3] Zhang, R.L., Walton, D.J., Hoskins, W.D. “Prediction of Natural Gas Consumption”, Proceedings of the Canadian conference on Electrical and Computer Engineering, 14-17 Sep 1993 Page(s):1156 - 1160 vol.2, 1993.
- [4] Χρίστος – Αιμίλιος Πραστίτης, «Πρόβλεψη ζήτησης φυσικού αερίου», Ατομική Διπλωματική Εργασία, Ιούνιος 2009, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου.
- [5] Danilo Mandic and Jonathon Chambers, “Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability” Pages : 45 -84 , 2001.
- [6] Dr. Herve Abdi; Dominique Valentin and Dr. Betty Edelman, “Neural Networks ” Pages : 61 -69 , 1999.
- [7] Haykin, S. Neural Networks, Prentice Hall, Englewood Cliffs (1998).
- [8] D.S. Broomhead and D. Lowe, “Multivariable functional interpolation and adaptive networks,” Complex Systems, vol. 2,Pages: 321–355, 1988.

- [9] Alireza Khotanzad, Hassan Elragal and Tsun- Liang Lu, “Combination of Artificial Neural-Networks Forecasters for Prediction of Natural Gas Consumption ”, IEEE Transactions on Neural Networks Pages: 464-473, vol.11 no.2 , 2000.
- [10] Διαλέξεις μαθήματος ΕΠΛ442 «Πληροφοριακά συστήματα μάθησης», Δρ. Χρίστος Χριστοδούλου, Πανεπιστήμιο Κύπρου.
- [11] Kohonen, T. and Honkela, T. "Kohonen network". Scholarpedia ,2007.
- [12] Elman, J.L. (1990). "Finding Structure in Time". Cognitive Science 14, Pages :179–211
- [13] M. Jordan, “Serial order: A parallel distributed processing approach”, Univ. California San Diego, Inst. Cognitive Sci., ICS Rep. 8604, 1986
- [14] Werbos, P., Backpropagation through time: What it does and how to do it, Proceedings of the IEEE, 78(10), 1990.
- [15] M.C. Mozer, A focused back-propagation algorithm for temporal pattern recognition. Complex Syst. 3 ,Pages. 349–381, 1989.

Παράρτημα Α

Παράρτημα Α: Κώδικας για ανάγνωση και εγγραφή δεδομένων	A-1
---	-----

Παράρτημα Α: Κώδικας για ανάγνωση και εγγραφή δεδομένων:

ReadingData.java

```
/* Lesson:      Thesis Project
 * Programme:   Gas Prediction Software
 * Target:      To make to make a program that predicts the gas demand
 * Author:      Kyprianos Hadjidemetrioy (cs06ck3)
 * Student Id:  880829
 *
 */

package data;

/*****
 * import Statements */
*****/
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.StreamTokenizer;
import java.util.StringTokenizer;

import javax.swing.JOptionPane;

/*****
 *
 * ReadingData Class
 *
 *****/
public class ReadingData {

    //train patterns number
    private int TRAIN;

    //test patterns number
    private int TEST;

    //train disired output number
    private int Dtrain;

    //test disired output number
    private int Dtest;

    //center number
    private int CNUM;

    //number of input neurons,hidden neurons and output neurons
    private int numOfInputs,numOfNeurons,numOfOutputs;

    //tables for reading datas
    private double trainData[][];
    private double testData[][];
```

```

private double idealTrain[][];
private double idealTest[][];

//filenames
private String FileTrain,FileTest,FileTrainActual,FileTestActual;

/*****
 * ReadingData Class constructor for rbf network
 *
 *****/
//constructor of rbfNetwork
public ReadingData(int cNUM,String FileTrain,String FileTest,String
FileTrainActual,String FileTestActual) {
    super();
    this.FileTest=FileTest;
    this.FileTestActual=FileTestActual;
    this.FileTrain=FileTrain;
    this.FileTrainActual=FileTrainActual;
    try {

        this.numOfInputs= this.InitData();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    CNUM = cNUM;
    trainData = new double[TRAIN][this.numOfInputs];
    testData = new double[TEST][this.numOfInputs];
    idealTrain = new double[TRAIN][1];
    idealTest = new double[TEST][1];

}

/*****
 *
 * ReadingData Class constructor for recurrent network
 *
 *****/
//constructor of RNNNetwork
public ReadingData(int numOfNeurons, int numOfOutputs, String FileTrain,String
FileTest,String FileTrainActual,String FileTestActual) {
    super();
    this.FileTest=FileTest;
    this.FileTestActual=FileTestActual;
    this.FileTrain=FileTrain;
    this.FileTrainActual=FileTrainActual;
    try {
        this.numOfInputs= this.InitData();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    this.check();

    this.numOfNeurons=numOfNeurons;
    this.numOfOutputs=numOfOutputs;

```

```

        trainData = new double[TRAIN][numOfInputs];
        testData = new double[TEST][numOfInputs];
        idealTrain = new double[TRAIN][1];
        idealTest = new double[TEST][1];
    }

    /**
     *
     * Set-Get Methods
     */

    public int getNumOfInputs() {
        return numOfInputs;
    }
    public void setNumOfInputs(int numOfInputs) {
        this.numOfInputs = numOfInputs;
    }
    public int getNumOfNeurons() {
        return numOfNeurons;
    }
    public void setNumOfNeurons(int numOfNeurons) {
        this.numOfNeurons = numOfNeurons;
    }
    public int getNumOfOutputs() {
        return numOfOutputs;
    }
    public void setNumOfOutputs(int numOfOutputs) {
        this.numOfOutputs = numOfOutputs;
    }

    /**
     *
     * InitData Methods
     */

    /** this method reads train data, test data,
     * actual output for train data and test data
     * and computes their number.
     */
    public int InitData() throws IOException {
        int NumInputs=0;
        FileInputStream fstream = new FileInputStream(FileTrain);
        DataInputStream in = new DataInputStream(fstream);
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        String strLine;
        strLine = br.readLine();

        StringTokenizer st = new StringTokenizer(strLine);
        int counter=0;
        while (st.hasMoreElements()) {
            st.nextToken();
            counter++;
        }
        NumInputs=counter;

        //train data

```

```

Reader r = new BufferedReader(new FileReader(FileTrain));
StreamTokenizer stok = new StreamTokenizer(r);
stok.parseNumbers();
TRAIN= 1;
int col = 0;
stok.nextToken();
while (stok.ttype != StreamTokenizer.TT_EOF) {
    if (stok.ttype == StreamTokenizer.TT_NUMBER) {
        if (col == NumInputs) {
            col = 0;
            TRAIN++;
        }
        col++;
    }

    stok.nextToken();
}

//=====
//test data
r = new BufferedReader(new FileReader(FileTest));
stok = new StreamTokenizer(r);
stok.parseNumbers();
TEST = 1;
col = 0;
stok.nextToken();
while (stok.ttype != StreamTokenizer.TT_EOF) {
    if (stok.ttype == StreamTokenizer.TT_NUMBER) {
        if (col == NumInputs) {
            col = 0;
            TEST++;
        }
        col++;
    }

    stok.nextToken();
}

//=====
//actual output for train data
r = new BufferedReader(new FileReader(FileTrainActual));
stok = new StreamTokenizer(r);
stok.parseNumbers();
Dtrain = 0;
col = 0;
stok.nextToken();
while (stok.ttype != StreamTokenizer.TT_EOF) {
    if (stok.ttype == StreamTokenizer.TT_NUMBER) {
        Dtrain++;
    }

    stok.nextToken();
}
System.out.println(Dtrain);

//=====
//actual output for test data
r = new BufferedReader(new FileReader(FileTestActual));
stok = new StreamTokenizer(r);
stok.parseNumbers();
Dtest = 0;

```

```

        col = 0;
        stok.nextToken();
        while (stok.ttype != StreamTokenizer.TT_EOF) {
            if (stok.ttype == StreamTokenizer.TT_NUMBER) {
                Dtest++;
            }

            stok.nextToken();
        }
        return NumInputs;
    }

}

/*****
 *
 * check Method
 *
 *****/
/* this method checks if training and disired results files
 * have the same number of lines.Also check the same for testing files.
 * Finally checks if user set all files correctly
 */
public boolean check() {
    boolean flag;
    if((this.TRAIN==this.Dtrain) &&(this.TEST==this.Dtest)) {

        flag=true;
    }
    else {JOptionPane.showMessageDialog(null,"Please insert 4 files");
        flag=false;
    }
    return flag;
}

/*****
 *
 * Read Method
 *
 *****/
/* this method reads train data, test data,
 * actual output for train data and test data
 * and saves them to the appropriate tables.
 */
public void Read() throws IOException {
    //train data
    Reader r = new BufferedReader(new FileReader(FileTrain));
    StreamTokenizer stok = new StreamTokenizer(r);
    stok.parseNumbers();
    int line = 0;
    int col = 0;
    stok.nextToken();
    while (stok.ttype != StreamTokenizer.TT_EOF) {
        if (stok.ttype == StreamTokenizer.TT_NUMBER) {
            if (col == this.numOfInputs) {
                col = 0;
                line++;
            }
            trainData[line][col] = stok.nval;
            col++;
        }
    }
}

```

```

        }

        stok.nextToken();
    }

//=====
//test data
r = new BufferedReader(new FileReader(FileTest));
stok = new StreamTokenizer(r);
stok.parseNumbers();
line = 0;
col = 0;
stok.nextToken();
while (stok.ttype != StreamTokenizer.TT_EOF) {
    if (stok.ttype == StreamTokenizer.TT_NUMBER) {
        if (col == this.numOfInputs) {
            col = 0;
            line++;
        }
        testData[line][col] = stok.nval;
        col++;
    }

    stok.nextToken();
}

//=====
//actual output for train data
r = new BufferedReader(new FileReader(FileTrainActual));
stok = new StreamTokenizer(r);
stok.parseNumbers();
line = 0;
col = 0;
stok.nextToken();
while (stok.ttype != StreamTokenizer.TT_EOF) {
    if (stok.ttype == StreamTokenizer.TT_NUMBER) {

        idealTrain[line][0] = stok.nval;
        line++;
    }

    stok.nextToken();
}

//=====
//actual output for test data
r = new BufferedReader(new FileReader(FileTestActual));
stok = new StreamTokenizer(r);
stok.parseNumbers();
line = 0;
col = 0;
stok.nextToken();
while (stok.ttype != StreamTokenizer.TT_EOF) {
    if (stok.ttype == StreamTokenizer.TT_NUMBER) {

        idealTest[line][0] = stok.nval;
        line++;
    }
}

```

```

        stok.nextToken();
    }

}

/*****
 *
 * Get-Set Methods
 *
 *****/
public int getTRAIN() {
    return TRAIN;
}

public void setTRAIN(int tRAIN) {
    TRAIN = tRAIN;
}

public int getTEST() {
    return TEST;
}

public void setTEST(int tEST) {
    TEST = tEST;
}

public int getCNUM() {
    return CNUM;
}

public void setCNUM(int cNUM) {
    CNUM = cNUM;
}

public double[][] getTrainData() {
    return trainData;
}

public void setTrainData(double[][] trainData) {
    this.trainData = trainData;
}

public double[][] getTestData() {
    return testData;
}

```

```

public void setTestData(double[][] testData) {
    this.testData = testData;
}

public double[][] getIdealTrain() {
    return idealTrain;
}

public void setIdealTrain(double[][] idealTrain) {
    this.idealTrain = idealTrain;
}

public double[][] getIdealTest() {
    return idealTest;
}

public void setIdealTest(double[][] idealTest) {
    this.idealTest = idealTest;
}

public String getFileTrain() {
    return FileTrain;
}

public void setFileTrain(String fileTrain) {
    FileTrain = fileTrain;
}

public String getFileTest() {
    return FileTest;
}

public void setFileTest(String fileTest) {
    FileTest = fileTest;
}

public String getFileTrainActual() {
    return FileTrainActual;
}

public void setFileTrainActual(String fileTrainActual) {
    FileTrainActual = fileTrainActual;
}

public String getFileTestActual() {
    return FileTestActual;
}

```



```

        public void setFileTestActual(String fileTestActual) {
            FileTestActual = fileTestActual;
        }
    }
}

```

WritingOut.java

```

/* Lesson:      Thesis Project
 * Programme:   Gas Prediction Software
 * Target:      To make to make a program that predicts the gas demand
 * Author:      Kyprianos Hadjidemetrioy (cs06ck3)
 * Student Id:  880829
 *
 */

package data;

/*****
 * import Statements */
*****/

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.text.SimpleDateFormat;
import java.util.Calendar;

/*****
 *
 * WritingOut method
 *
 *****/
public class WritingOut {

    //name of algorythm which we want to write the results
    private String AlgorithmName;
    //date and time for the specific training
    private String DATE_FORMAT_NOW ;
    private Calendar cal ;
    //filename of the output file
    private String Filename;

    /*****
     *
     * WritingOut constructor
     *
     *****/
    public WritingOut(String AlgorithmName) {
        this.AlgorithmName=AlgorithmName;
        DATE_FORMAT_NOW = "yyyyMMdd_HH-mmss";
        cal = Calendar.getInstance();
    }

    /*****

```

```

*
* writeOut method
*
*****/

public void writeOut(int flag,int epoch,double totalEr,int TRAIN,double testingEr
,double totSuc){

    //if it is the first time which this method called
    //set the current date and time
    if (flag==1){
        DATE_FORMAT_NOW = "yyyy-MM-dd_HH-mm-ss";
        cal = Calendar.getInstance();

    }
    SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW);

    //Write the epoch, training , testing error and total success
    //in a file which it's name consist of the
    //algorithym name-current date and time
    try{
        // Create file

        this.Filename=AlgorithmName+sdf.format(cal.getTime());
        FileWriter fstream = new FileWriter(Filename,true);
        BufferedWriter out = new BufferedWriter(fstream);
        out.write(epoch + "\t" + (totalEr / (TRAIN/2)) + "\t"
                + testingEr + "\t" + totSuc);

        out.newLine();
        //Close the output stream
        out.close();
    }catch (Exception e){//Catch exception if any
        System.err.println("Error: " + e.getMessage());
    }
}

public String getFilename() {
    return Filename;
}

}

```

Παράρτημα Β

Παράρτημα Β: Κώδικας υλοποίησης δικτύου RBF και υβριδικού δικτύου B-1

Παράρτημα Β: Κώδικας υλοποίησης δικτύου RBF:

Input.java

```
/* Lesson:      Thesis Project
 * Programme:   Gas Prediction Software
 * Target:      To make to make a program that predicts the gas demand
using RBF network
 * Author:      Kyprianos Hadjidemetrioy (cs06ck3)
 * Student Id:  880829
 *
 */

package rbfNetwork;

/*****
 *
 * Input Class
 *
 *****/
public class Input {

    //generate input neurons
    private double inp[] = new double[8];

    /*****
     *
     * initInp Class
     *
     *****/

    //This method takes the data array and a specific position
    //and sets the input neurons the specific pattern
    public void initInp(double[][] inp, int pos) {
        for (int i = 0; i < 8; i++) {
            this.inp[i] = inp[pos][i];
        }
    }

    public double[] getInp() {
        return inp;
    }
}
```

Center.java

```
/* Lesson:      Thesis Project
 * Programme:   Gas Prediction Software
 * Target:      To make to make a program that predicts the gas demand
using RBF network
 * Author:      Kyprianos Hadjidemetrioy (cs06ck3)
 * Student Id:  880829
 *
```

```

*/

package rbfNetwork;

/*****
/* import Statements */
*****/
import java.util.Random;

/*****
*
* Center Class
*
*****/

//This class represents a center

public class Center {

    //Center's vector
    private double values[] = new double[8];
    //Center's sigma
    private double sigma;
    //Center's coefficient
    private double coefficient;
    //euclidian distance
    private double edist;
    //output of center
    private double outC;
    //distance center-input
    private double dist[] = new double[8];
    //new value for sigma
    private double newsigma;
    //new value for coefficient
    private double newcoefficient;
    //new values for center's vector
    private double newval[] = new double[8];

    /*****
    *
    * initParameters Method
    * *****/
    //this method inits center's coefficient to small value
    public void initParameters(double train[][]) {

        Random generator = new Random();

        coefficient = generator.nextDouble() - 0.6;

    }

    /*****
    *

```

```

* computeApplyNewParams Method
*
*****/

//This method computes the new value for sigma,coefficient
//and center's vector

public void computeApplyNewParams(double lr, double lc, double ls,
double error) {

    // compute new value for sigma
    newsigma = ls * error * coefficient * outC * Math.pow(edist, 2)
        / (Math.pow(sigma, 3));

    // compute new value for coefficient
    newcoefficient = lc * error * outC;

    // compute new value for centers's vector position
    for (int i = 0; i < 8; i++) {
        newval[i] = lr * error * coefficient * outC * dist[i]
            / (Math.pow(sigma, 2));
        values[i] += newval[i];
    }

    //apply new values
    coefficient += newcoefficient;
    sigma += newsigma;
}

```

```

/*****
*
* computeApplyNewParams Method
*
*****/

//this method calculate the euclidian distance center-input
public void computeEuclidianDist(double inp[]) {
    double sum = 0;

    for (int i = 0; i < 8; i++) {
        dist[i] = (inp[i] - values[i]);
        sum += (inp[i] - values[i]) * (inp[i] - values[i]);
    }

    edist = Math.sqrt(sum);
}

/*****
*
* computeOut Method
*
*****/

```

```

//This method computes center's output
//using gaussian function

public void computeOut() {

    outC = Math.exp(-Math.pow(edist, 2) / (2 * Math.pow(sigma, 2)));
}

/*****
 *
 * Set-Get Methods
 *
 *****/

public void setSigma(double sigmaInit) {
    sigma = sigmaInit;
}

public double[] getValues() {
    return values;
}

public double getCoefficient() {
    return coefficient;
}

public double getOutC() {
    return outC;
}

public void setValues(int pos, double index) {
    this.values[pos] = index;
}

}

```

Output.java

```

/* Lesson:      Thesis Project
 * Programme:   Gas Prediction Software
 * Target:      To make to make a program that predicts the gas demand
using RBF network
 * Author:      Kyprianos Hadjidemetrio (cs06ck3)
 * Student Id:  880829
 *
 */

package rbfNetwork;

/*****
 * import Statements */
 *****/
import java.util.ArrayList;

/*****
 *
 * Output Class

```

```

*
*****/
public class Output {

    //the network output
    private double output;

    //the network error
    public double error;

    public double biasw = 0.246;

    /**
     * changeBias method
     */
    /**
     * //this method change the bias value
     public void changewBias(double n, double er) {
         biasw += n * er;

     }

    /**
     * computeNetworkOut method
     */
    /**
     * //this method computes output using
     //the formula sum(center[i]*coef[i])
     public void computeNetworkOut(ArrayList<Center> cen) {
         double sum = 0;
         for (int i = 0; i < cen.size(); i++) {

             sum += cen.get(i).getOutC()* cen.get(i).getCoefficient();

         }

         output = sum + biasw;

     }

    /**
     * computeNetworkError method
     */
    /**
     * //this method computes network error using
     //the formula disired-actual
     public void computeNetworkError(double dis) {
         error = (dis - output);

     }

    /**
     * Set-Get Methods

```



```

*
*****/

    public double getOutput() {
        return output;
    }

    public double getError() {
        return error;
    }
}

```

Som.java

```

/* Lesson:      Thesis Project
 * Programme:   Gas Prediction Software
 * Target:      To make to make a program that predicts the gas demand
using RBF network
 * Author:      Kyprianos Hadjidemetrioy (cs06ck3)
 * Student Id:  880829
 *
 */

package rbfNetwork;

*****/
/* import Statements */
*****/

import java.util.ArrayList;

import kohonen.LearningData;
import kohonen.WTALearningFunction;
import learningFactorFunctional.ConstantFunctionalFactor;
import metrics.EuclidesMetric;
import network.DefaultNetwork;
import topology.MatrixTopology;

*****/
*
* SOM Class
*
*****/

public class Som {

    *****/
    *
    * execute method
    *
    *****/

    //this method finds the initial positions
    //of centers and sigmas using Som clustering algorhythm

    public double[] execute(ArrayList<Center> cen, int size,String fileNameTrain) {

```

```

//Grid Size
size=10;

GaussNeighbourhoodFunction f = new GaussNeighbourhoodFunction(5);
//generate map (size*size)
MatrixTopology topology = new MatrixTopology(size, size);

double[] maxWeight = { 1, 1, 1, 1, 1, 1, 1, 1 };

//Set network parametrs
DefaultNetwork network = new DefaultNetwork(8, maxWeight, topology);

ConstantFunctionalFactor constantFactor = new ConstantFunctionalFactor(
    0.8);
LearningData fileData = new LearningData(fileNameTrain);
WTMLearningFunction learning = new WTMLearningFunction(network,
    100, new EuclidesMetric(), fileData, constantFactor, f);

//Start SOM
int clusters=0;
int[] total;
total=learning.learn(size);

//finding the number of clusters-centers
for (int i = 0; i < size*size; i++) {
    if (total[i]>=30000) clusters++;
}
System.out.println("Total Centers= " + clusters);
for (int i = 0; i < clusters; i++) {
    Center tmp = new Center();
    cen.add(tmp);
}
//Set the initial values od center's vectors
//and compute the sigma for each center
double sigma[] = new double[clusters];
for (int i = 0; i < clusters; i++) {
    for (int j = 0; j < 8; j++)
        cen.get(i).setValues(j, network.getNeuron(i).getWeight()[j]);
    sigma[i] = network.getNeuron(i).getValue(
        network.getNeuron(i).getWeight());
}

return sigma;
}
}

```

Rbfgas.java

```

/* Lesson:      Thesis Project
* Programme:    Gas Prediction Software
* Target:       To make to make a program that predicts the gas demand using RBF
                network
* Author:       Kyprianos Hadjidemetrioy (cs06ck3)
* Student Id:   880829
*
*/

```

```

package rbfNetwork;

/*****
/* import Statements */
*****/

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

import GUI.ProgressWindow;
import GUI.ReportWindow;

import data.ReadingData;
import data.WritingOut;

import kohonen.LearningData;
import kohonen.WTALearningFunction;

import learningFactorFunctional.ConstantFunctionalFactor;
import metrics.EuclidesMetric;

import network.DefaultNetwork;

import topology.MatrixTopology;

/*****
*
* RBFGas Class
*
*****/
public class RbfGas {

    private int TRAIN;           //number of train patterns
    private int TEST;           //number of test patterns
    private int CNUM;           //number of centers
    private double LR;           //learning rate for centers
    private double LC;           //learning rate for coefficients
    private double LS;           //learning rate for sigma
    private int ITNUM;           //number of iterations
    private double trainData[][]; //table with all training data
    private double testData[][]; //table with all testing data
    private double idealTrain[][]; //table with disired demands of training patterns
    private double idealTest[][]; //table with disired demands of testing patterns

    //input neurons
    private Input input = new Input();

    //output neurons
    private Output out = new Output();

```

```

//success percentage
private double totSuc = 0;

//list with centers objects
private ArrayList<Center> centers = new ArrayList<Center>();

//readingdata object which holds all data which had read
private ReadingData r1;

/*****
 *
 * RBFGas Class Constructor
 *
 *****/

//Initialize all parametrs with the appropriate values

public RbfGas(ReadingData r1,double LR,double LC,double LS,int ITNUM) {
    super();
    this.r1=r1;
    TRAIN = r1.getTRAIN();
    TEST = r1.getTEST();
    CNUM = r1.getCNUM();
    this.LR = LR;
    this.LC = LC;
    this.LS = LS;
    this.ITNUM = ITNUM;
    trainData = new double[TRAIN][r1.getNumOfInputs()];
    testData = new double[TEST][r1.getNumOfInputs()];
    idealTrain = new double[TRAIN][1];
    idealTest = new double[TEST][1];
    this.trainData=r1.getTrainData();
    this.testData=r1.getTestData();
    this.idealTest=r1.getIdealTest();
    this.idealTrain=r1.getIdealTrain();
}

/*****
 *
 * computeSigma Method
 *
 *****/

//this method computes for each center the initial value of sigma
public double[] computeSigma(int size) {

    double[] sigma = new double[size];

    //for each center
    for (int i = 0; i < size; i++) {
        Center tmp = centers.get(i);
        double min = 1;
        double max = 0;
    }
}

```

```

        //find the maximun and minimun values of the specific center's values
        for (int j = 0; j < 8; j++) {
            if (tmp.getValues()[j] >= max) {
                max = tmp.getValues()[j];
            }
            if (tmp.getValues()[j] <= min)
                min = tmp.getValues()[j];
        }
        //compute the center's i sigma
        sigma[i] = (max - min) / 2;
    }

    return sigma;
}

/*****
 *
 * doTesting Method
 *
 *****/

//this method tests the network on testing data
public double doTesting(double test[][], double dis[][], int size) {

    //input neurons
    Input testing = new Input();

    //testing ereror
    double totalEr = 0;

    //temporary error
    double error = 0;

    //for each testing pattern
    for (int i = 0; i < size; i++) {

        //set input
        testing.initInp(test, i);

        //for each center compute the euclidian distance and output
        for (int j = 0; j < centers.size(); j++) {

            centers.get(j).computeEuclidianDist(testing.getInp());
            centers.get(j).computeOut();

        }

        //compute network output
        out.computeNetworkOut(centers);

        //compute network error
        out.computeNetworkError(idealTest[i][0]);

        //compute MSE

```

```

        totalEr += Math.pow(out.error, 2);

        //COMPUTE total success percentage
        error = Math.abs(idealTest[i][0] - out.getOutput());
        error = (error * 100) / idealTest[i][0];

        error = 100 - error;

        totSuc = totSuc + error;

        error = 0.0;

    }
    totalEr /= size;

    return totalEr;

}

/*****
 *
 * finalizeTesting Method
 *
 *****/

//this method pass the testing data in the network when training has been finished and
//prints in file the network output and the disired demand for each testing pattern

public void finalizeTesting() {

    //input neuron
    Input test = new Input();
    //for each testing pattern
    for (int i = 0; i < TEST; i++) {

        //set input
        test.initInp(testData, i);

        //for each center compute the euclidian distance and output
        for (int j = 0; j < centers.size(); j++) {

            centers.get(j).computeEuclidianDist(test.getInp());
            centers.get(j).computeOut();

        }
        //compute the final output
        out.computeNetworkOut(centers);

        //comput the error
        out.computeNetworkError(idealTest[i][0]);

        //write to file the current output and disired demand
        try{
            // Create file
            FileWriter fstream = new FileWriter("outRBF",true);
            BufferedWriter out1 = new BufferedWriter(fstream);
            out1.write(i + "\t" + out.getOutput() + "\t" + idealTest[i][0]);
            out1.newLine();
        }
    }
}

```

```

        out1.close();
    } catch (Exception e){//Catch exception if any
        System.err.println("Error: " + e.getMessage());
    }
}

}

/*****
 *
 * startRBF Method
 *
 *****/

//this method implements the hybriic network of SOM-RBF and
//writes in file for each epoch the training,
//testing and successes percentage

public void startRBF(WritingOut wr) {

    //create SOM class object
    Som findCenters = new Som();

    //create centers
    for (int i = 0; i < CNUM; i++) {
        Center tmp = new Center();
        tmp.initParameters(trainData);
        centers.add(tmp);
    }

    double sigma[] = new double[CNUM];

    System.out
        .println("Clustering Algoritym to find Centers Please Wait...");

    //find initial position of centers using SOM clusterin algorithim
    sigma = findCenters.execute(centers, CNUM,r1.getFileTrain());

    //for each center set the sigma value
    for (int i = 0; i < CNUM; i++) {
        centers.get(i).setSigma(sigma[i]);
    }

    //variable for traing error
    double totalEr = 0;

    int counter = 0;

    int epoch = 0;

    double testingEr = 0;

    Random generator = new Random();

    int flag=1;
    //Reapeat until we reach epoch number
    do {

```

```

        //set input pattern
        input.initInp(trainData, counter);

        //for each center compute the euclidian distance and output
        for (int i = 0; i < centers.size(); i++) {

            centers.get(i).computeEuclidianDist(input.getInp());
            centers.get(i).computeOut();

        }

        //compute network output
        out.computeNetworkOut(centers);

        // compute network error
        out.computeNetworkError(idealTrain[counter][0]);

        //compute training error
        totalEr += Math.pow(out.error, 2);

        //change centers positions, sigmas and coefficients
        for (int i = 0; i < centers.size(); i++) {
            centers.get(i).computeApplyNewParams(LR, LC, LS,
                idealTrain[counter][0] - out.getOutput());
        }

        //change bias value
        out.changewBias(LC, idealTrain[counter][0] - out.getOutput());

        //increase pattern counter
        counter++;

        //we reach the end of an epoch
        if (counter == TRAIN) {

            epoch++;

            //comput testing error and total success for the specific epoch
            testingEr = doTesting(testData, idealTest, TEST);
            totSuc = totSuc / TEST;

            //writing results to file
            wr.writeOut(flag,epoch,totalEr,TRAIN,testingEr,totSuc);
            flag=0;

            counter = 0;
            totalEr = 0;
        }

    } while (epoch != ITNUM);
    totSuc = 0;

    //check the network after training
    finalizeTesting();

}

```



```

/*****
*
* startRBForiginal Method
*
*****/

//this method implements the RBF network and
//writes in file for each epoch the training,
//testing and successes percentage

public void startRBForiginal(WritingOut wr) {

    //Create and initialize centers
    for (int i = 0; i < CNUM; i++) {
        Center tmp = new Center();
        tmp.initParameters(trainData);
        centers.add(tmp);
    }

    double sigma[] = new double[CNUM];

    System.out
        .println("Initializing Centers Randomly.....");

    //set the initial position of each center randomly
    sigma = findCentersRBF();

    //for each center set the sigma value
    for (int i = 0; i < CNUM; i++) {
        centers.get(i).setSigma(sigma[i]);
    }

    //variable for traing error
    double totalEr = 0;

    int counter = 0;

    int epoch = 0;

    double testingEr = 0;

    Random generator = new Random();

    int flag=1;
    //Reapeat until we reach epoch number
    do {
        //set input pattern
        input.initInp(trainData, counter);

        //for each center compute the euclidian distance and output
        for (int i = 0; i < centers.size(); i++) {

            centers.get(i).computeEuclidianDist(input.getInp());
            centers.get(i).computeOut();

        }

    }

```

```

        //compute network output
        out.computeNetworkOut(centers);

        // compute network error
        out.computeNetworkError(idealTrain[counter][0]);

        //compute training error
        totalEr += Math.pow(out.error, 2);

        //change centers positions, sigmas and coefficients
        for (int i = 0; i < centers.size(); i++) {
            centers.get(i).computeApplyNewParams(LR, LC, LS,
                idealTrain[counter][0] - out.getOutput());
        }

        //change bias value
        out.changewBias(LC, idealTrain[counter][0] - out.getOutput());

        //increase pattern counter
        counter++;

        //we reach the end of an epoch
        if (counter == TRAIN) {

            epoch++;

            //comput testing error and total success for the specific epoch
            testingEr = doTesting(testData, idealTest, TEST);
            totSuc = totSuc / TEST;

            //writing results to file
            wr.writeOut(flag,epoch,totalEr,TRAIN,testingEr,totSuc);
            flag=0;

            counter = 0;
            totalEr = 0;
        }

    } while (epoch != ITNUM);
    totSuc = 0;

    //check the network after training
    finalizeTesting();
}
}

```

Παράρτημα Γ

Παράρτημα Γ: Κώδικας υλοποίησης δικτύου Jordan

Γ-1

Παράρτημα Γ: Κώδικας υλοποίησης δικτύου Jordan:

RNNetwork.java

```
/* Lesson:      Thesis Project
 * Programme:   Gas Prediction Software
 * Target:      To make to make a program that predicts the gas demand
 * Author:      Kyprianos Hadjidemetrioy (cs06ck3)
 * Student Id:  880829
 *
 */
package bpttNetwork;

/*****
 * import Statements */
*****/
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.util.Random;

import GUI.ReportWindow;

import data.ReadingData;
import data.WritingOut;

/*****
 *
 * RNNetwork Class
 *
 *****/
public class RNNetwork {

    //table for inputs patterns
    private double input[][];

    //table for hiddenlayer neurons outputs or each time step
    private double hiddenLayer[][];

    //table for output neurons outputs or each time step
    private double output[][];

    //weights from input to hidden neurons
    private double weightInputs[][];

    //weights from hidden neurons to output
    private double weightOutputs[][];

    //reccurent weigths for previous output to hidden
    private double wRec[];

    //bias to hidden neurons
    private double bias[];

    //bias to output neuron
    private double biasO;

    //time Constant for state units
```

```

private double timeConst;

//success percentage
private double totSuc;

//the number of input,hidden and output neurons
private int numOfInputs,numOfNeurons,numOfOutputs;

//number of training patterns
public int TRAIN;

//number of testing patterns
public int TEST;

//error of each time step
private double errorTotal[];
public int CNUM;

//learning rate
private double LR;
private int cv ;

//iteration number
private int ITNUM;

//number of time steps
private int steps;

//tables for training and testing data
private double trainData[][];
private double testData[][];
private double idealTrain[][];
private double idealTest[][];

//tables for delta value
private double deltaOutput[][];
private double deltaHidden[][];

/*****
*
* RNNetwork constructor
*
*****/

//Initializes the network parametrs with the appropriate values
public RNNetwork(ReadingData r1,double LR,int steps,int ITNUM,double TC) {
    TRAIN = r1.getTRAIN();
    TEST = r1.getTEST();
    CNUM = r1.getCNUM();
    this.LR=LR;
    this.steps=steps;
    this.ITNUM=ITNUM;
    this.totSuc=0;
    cv=0;

    //Set time constant value
    timeConst = TC;

```

```

numOfInputs=r1.getNumOfInputs();
errorTotal= new double[steps];

numOfNeurons=r1.getNumOfNeurons();
numOfOutputs=r1.getNumOfOutputs();
bias= new double[numOfNeurons];
weightInputs = new double[numOfInputs][numOfNeurons];
weightOutputs = new double[numOfNeurons][numOfOutputs];
hiddenLayer=new double[steps][numOfNeurons];
wRec=new double[numOfNeurons];
trainData = new double[TRAIN][r1.getNumOfInputs()];
testData = new double[TEST][r1.getNumOfInputs()];
idealTrain = new double[TRAIN][1];
idealTest = new double[TEST][1];
output = new double[steps+1][this.numOfOutputs];
input = new double[steps][this.numOfInputs];
idealTest = new double[TEST][1];
deltaOutput=new double[steps][r1.getNumOfOutputs()];
deltaHidden=new double[steps][r1.getNumOfNeurons()];
this.trainData=r1.getTrainData();
this.testData=r1.getTestData();
this.idealTest=r1.getIdealTest();
this.idealTrain=r1.getIdealTrain();
initWeights();
}

/*****
*
* initWeights method
*
*****/

//this method initialize all the weights
//with random values from -1 to 1
public void initWeights() {
    int max=1;
    int min=-1;
    biasO=(Math.random() * (max - min + 1) ) + min;
    for(int j=0;j<numOfInputs;j++)
        for(int k=0;k<numOfNeurons;k++)
            weightInputs[j][k]= (Math.random() * (max - min + 1) ) + min;

    for(int j=0;j<numOfNeurons;j++)
        for(int k=0;k<numOfOutputs;k++)
            weightOutputs[j][k]= (Math.random() * (max - min + 1) ) + min;

    for(int p=0;p<steps;p++)
        for(int c=0;c<numOfNeurons;c++){
            bias[c]=(Math.random() * (max - min + 1) ) + min;
            wRec[c] = (Math.random() * (max - min + 1) ) + min;
        }
    for (int i=0;i<steps;i++)
        output[i][0]=0;
}

```

```

/*****
*
* ForwardPass method
*
*****/

//This method makes a forward pass through the network and
//computes the hidden output,the network
//output and the error for each time step

public void ForwardPass() {
    //Set the input data for each time step
    for (int i=0;i<steps;i++){

        if (cv==TRAIN){
            return;
        }

        for (int j=0;j<numOfInputs;j++){
            //set data
            input[i][j]=idealTrain[cv][j];
        }
        cv++;

    }

    //find the sum of the inp*weight
    //and then calculate the hidden neuron i output
    //for each time step
    double sum=0;
    int i;
    for ( i=0;i<steps;i++){
        sum=0;
        for (int p=0;p<numOfNeurons;p++){
            for (int j=0;j<numOfInputs;j++){
                sum+=weightInputs[j][p]*input[i][j];

            }

            //for local recurrent use exponential decay
            if (i!=0)
                for (int j=0;j<numOfOutputs;j++)
                    sum+=(output[i-1][j]*Math.exp(-i/timeConst))*wRec[p];

            //compute hidden neuron output
            hiddenLayer[i][p]=1/(1+Math.exp(-sum));
        }

        //calculate network output for each time step
        sum=0;
        for (int p=0;p<numOfNeurons;p++){
            for (int j=0;j<numOfOutputs;j++){

sum+=weightOutputs[p][j]*hiddenLayer[i][p];

```

```

    }

    }

    output[i][numOfOutputs-1]= 1/(1+Math.exp(-sum));

    //calculate error for the ith time step
    errorTotal[i]=idealTrain[Math.abs(cv-(i+steps))][0]-
output[i][numOfOutputs-1];
    }
    trainEr = idealTrain[cv][0]-output[steps-1][numOfOutputs-1];
}

/*****
*
* backwardpass method
*
*****/

//This method makes a backward pass throught the network and
//computes the error for transfer to previous layer known as delta

public void backwardpass() {

    //start rom the last time step
    int timeStep=steps-1;
    do{
        //if we are at the last time step
        if (timeStep==steps-1){

            //compute delta for output
            deltaOutput[timeStep][0]=output[timeStep][0]*(1-
output[timeStep][0])*errorTotal[timeStep];

            //compute delta for hidden layer at the last step
            for (int i=0;i<numOfNeurons;i++){
                double s=0;

                for (int j=0;j<numOfOutputs;j++)
                    s+=deltaOutput[timeStep][0]*weightOutputs[i][j];

                deltaHidden[timeStep][i]=hiddenLayer[timeStep][i]*(1-
hiddenLayer[timeStep][i])*s;
                s=0;
            }

        }

        //to any other time step
        else{

            //compute delta for output for the specific step
            double sum=0;

```



```

        for (int i=0;i<numOfNeurons;i++){
            sum+=wRec[i]*deltaHidden[timeStep+1][i];
        }

        deltaOutput[timeStep][0]=output[timeStep][0]*(1-
        output[timeStep][0])*(sum+errorTotal[timeStep]);

        //compute delta for hidden neurons for the specific step
        for (int i=0;i<numOfNeurons;i++){
            double s=0;

            for (int j=0;j<numOfOutputs;j++)

                s+=deltaOutput[timeStep][j]*weightOutputs[i][j];

            deltaHidden[timeStep][i]=hiddenLayer[timeStep][i]*(1-
            hiddenLayer[timeStep][i])*s;
            s=0;
        }

    }
    timeStep--;

} while(timeStep!=0); //we reach the first time step

}

/*****
 *
 * updateweights method
 *
 *****/

//this method updates all the weights fo the network
//using the appropriate formulas
public void updateweights() {

    double []sum=new double [numOfNeurons];
    double [][]sum1=new double [numOfInputs][numOfNeurons];
    double [][]sum2=new double [numOfNeurons][numOfOutputs];

    //computing sums for all time steps
    int pos=0;
    for (int i=1;i<steps;i++){
        if (i==1) output[i-1][0]=0;

        for(int k=0;k<numOfNeurons;k++){
            sum[k]+=output[i-1][0]*deltaHidden[i][k];

            for(int j=0;j<numOfInputs;j++)
                sum1[j][k]+=deltaHidden[i][k] * input[i][j];
        }
    }
}

```

```

        for(int j=0;j<numOfOutputs;j++)
            sum2[k][j]+=hiddenLayer[i-1][k] * deltaOutput[i][j];

    }

}

for(int k=0;k<numOfNeurons;k++){
    //update reccurent weights
    wRec[k]+=LR*sum[k];

    for(int j=0;j<numOfInputs;j++)
        //update weights from input to hidden
        weightInputs[j][k]+=LR*sum1[j][k];
    for(int j=0;j<numOfOutputs;j++){
        //update weights from hidden to output
        weightOutputs[k][j]+=LR*sum2[k][j];
    }

}

}

}

/*****
*
* test method
*
*****/
public void test() {

    //temporary error
    double error=0;

    //set the testing data to network inputs
    for (int i=0;i<steps;i++){

        if (cv==TEST){
            return;
        }

        for (int j=0;j<numOfInputs;j++){
            //set inputs
            input[i][j]=testData[cv][j];
        }

        cv++;
    }

    //forward pass
    double sum=0;

    //for each time step calculate hidden neurons output
    //and network output
    for (int i=0;i<steps;i++){
        sum=0;

```

```

        for (int p=0;p<numOfNeurons;p++){
            for (int j=0;j<numOfInputs;j++){
                sum+=weightInputs[j][p]*input[i][j];

            }
            if(i!=0)
                for (int j=0;j<numOfOutputs;j++){
                    sum+=output[i][j]*wRec[p];

                    //pth hidden neuron of ith time step output
                    hiddenLayer[i][p]=1/(1+Math.exp(-sum));
                }

            sum=0;
            for (int p=0;p<numOfNeurons;p++){
                for (int j=0;j<numOfOutputs;j++){
                    sum+=weightOutputs[p][j]*hiddenLayer[i][p];

                }

            }

            //compute error and output
            output[i][numOfOutputs-1]= 1/(1+Math.exp(-sum));
            errorTotal[i]=idealTest[Math.abs(cv-(i+steps))][0]-
            output[i][numOfOutputs-1];

            //calculate total success
            double dif = Math.abs(idealTest[Math.abs(cv-(i+steps-1))][0]-
            output[i][numOfOutputs-1])/2;

            error = (dif) / idealTest[Math.abs(cv-(i+steps))][0];

            totSuc = totSuc + error;

            error = 0.0;

        }

    }

    /*****
    *
    * finalizing Method
    *
    *****/

```

```
//this method pass the testing data in the network when training has been finished and
//prints in file the network output and the disired demand for each testing pattern
```

```
public void finalizing(){
cv=0;
//for all testing paterns
do {
this.test(); //testing network

for(int i=0;i<steps;i++) {
//write to file the current output and disired demand
try{
// Create file

FileWriter fstream = new FileWriter("outBPTT",true);
BufferedWriter out1 = new BufferedWriter(fstream);
out1.write(cv-
(i+steps)+"\t"+output[i][0)+"\t"+idealTest[Math.abs(cv-(i+steps))][0]);
out1.newLine();
//Close the output stream
out1.close();
} catch (Exception e){//Catch exception if any
System.err.println("Error: " + e.getMessage());
}
}
}
```

```
}while (cv!=TEST);
```

```
}
```

```

/*****
*
* execute Method
*
*****/
```

```
//this method trains the network
//using the BPTT algorythm
```

```
public void execute(WritingOut wr) {
int epoch=1;
int flag=1;
double sumT;

double sum=0;
do{

//make forward pass through the network
this.ForwardPass();

//make backward pass through the network to compute delta
this.backwardpass();

//update all the weights
this.updateweights();

//compute MSE
```

```

sum+=errorTotal[steps-1]*errorTotal[steps-1];

//END of an epoch
if (cv==TRAIN){
    cv=0;
    sumT=0;
    do {
        //testing the network
        this.test();

        //compute testing mse
        sumT+=errorTotal[steps-1]*errorTotal[steps-1];

    } while (cv!=TEST);

    //compute success percentage
    totSuc = 1-(totSuc / (TEST));
    totSuc*=100;
    sumT=sumT/(TEST/steps);

    //write training error,testing error and %success to file
    wr.writeOut(flag,epoch,sum,TRAIN/steps,sumT,totSuc);
    flag=0;

    epoch++;
    cv=0;
    sum=0;
}

} while(epoch!=ITNUM);

//finilize network
this.finalizing();
}
}

```

Παράρτημα Δ

Παράρτημα Δ: Κώδικας υλοποίησης διεπαφής – GUI

Δ-1

Παράρτημα Δ: Κώδικας υλοποίησης διεπαφής – GUI:

ReportWindow.java

```
/* Lesson:      Thesis Project
 * Programme:   Gas Prediction Software
 * Target:      To make to make a program that predicts the gas demand
 * Author:      Kyprianos Hadjidemetrioy (cs06ck3)
 * Student Id:  880829
 *
 */

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * Report.java
 *
 * Created on Jul 30, 2010, 3:16:51 PM
 */

package GUI;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class ReportWindow extends javax.swing.JFrame {

    /** Creates new form Report */
    public ReportWindow() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        // Δημιουργία συστατικών της φόρμας
        jPanel1 = new javax.swing.JPanel();
        OkButton = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        MainTextArea = new javax.swing.JTextArea();
        jMenuBar1 = new javax.swing.JMenuBar();
        FileMenu = new javax.swing.JMenu();
        SaveMenuItem = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        OkButton.setText("OK");
    }
}
```



```

        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );
    this.MainTextArea.setEditable(false);
    pack();
} // </editor-fold>
//Υλοποίηση ακροατών συμβάντων -> OkButton
private void OkButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.dispose();
}

private void SaveMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

//μέθοδος η οποία διαβάζει και εμφανίζει τα αποτελέσματα

public void ReadAndShow(String Filename) {
    try{
        // Open the file that is the first
        // command line parameter
        FileInputStream fstream = new FileInputStream(Filename);
        // Get the object of DataInputStream
        DataInputStream in = new DataInputStream(fstream);
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        String strLine;
        String epoch=null;
        //Read File Line By Line
        while ((strLine = br.readLine()) != null) {epoch=strLine;}
        //Close the input stream
        StringTokenizer st = new StringTokenizer(epoch);
        int count=0;
        this.MainTextArea.setText(epoch);
        /* while (st.hasMoreTokens()) {
            count++;
            if(count == 3) {
                this.MainTextArea.setText(st.nextToken());
            }
        }
        */
        in.close();
    } catch (Exception e){//Catch exception if any
        System.err.println("Error: " + e.getMessage());
    }
}
/**
 * @param args the command line arguments
 */
//δήλωση ιδιοτήτων ReportWindow
// Variables declaration - do not modify
private javax.swing.JMenu FileMenu;
private javax.swing.JTextArea MainTextArea;
private javax.swing.JButton OkButton;

```

```

private javax.swing.JMenuItem SaveMenuItem;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
// End of variables declaration

}

```

MainWindow.java

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * NewJFrame.java
 *
 * Created on Jul 30, 2010, 12:38:01 PM
 */

package GUI;

import data.ReadingData;
import data.WritingOut;

import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

import bpttNetwork.RNNNetwork;

import rbfNetwork.*;

import java.io.*;

/**
 *
 * @author souzaboy
 */
public class MainWindow extends javax.swing.JFrame {

    //Κατασκευαστής του MainWiondow
    public MainWindow() {
        ps1 = new ProgressWindow();
        ps1.start();
        rep = new ReportWindow();
    }

```

```

rep.setVisible(false);
    initComponents();
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
//αρχικοποίηση συστατικών γραφικής διεπαφής (κώδικας παραγμένος από Netbeans)
private void initComponents() {
    //Δημιουργία συστατικών της φόρμας
    buttonGroup1 = new javax.swing.ButtonGroup();
    jPanel1 = new javax.swing.JPanel();
    TrainButton = new javax.swing.JButton();
    TrainCheck = new javax.swing.JCheckBox();
    DesiredTrainButton = new javax.swing.JButton();
    DesiredTrainCheck = new javax.swing.JCheckBox();
    TestButton = new javax.swing.JButton();
    TestCheck = new javax.swing.JCheckBox();
    DesiredTestButton = new javax.swing.JButton();
    DesiredTestCheck = new javax.swing.JCheckBox();
    TabbedAlgorithm = new javax.swing.JTabbedPane();
    jPanel2 = new javax.swing.JPanel();
    LRrbfLabel = new javax.swing.JLabel();
    LClabel = new javax.swing.JLabel();
    LSlabel = new javax.swing.JLabel();
    RBFRadio = new javax.swing.JRadioButton();
    RBFsomRadio = new javax.swing.JRadioButton();
    NOClabel = new javax.swing.JLabel();
    EpochsLabel = new javax.swing.JLabel();
    EPOCHSrbfSpinner = new javax.swing.JSpinner();
    LRrbfSpinner = new javax.swing.JSpinner();
    LCrbfSpinner = new javax.swing.JSpinner();
    LSrbfSpinner = new javax.swing.JSpinner();
    NOCrbfSpinner = new javax.swing.JSpinner();
    jPanel4 = new javax.swing.JPanel();
    LRbpttLabel = new javax.swing.JLabel();
    LRbpttSpinner = new javax.swing.JSpinner();
    EPOCHSBpttLabel = new javax.swing.JLabel();
    EPOCHSBpttSpinner = new javax.swing.JSpinner();
    NHUbpttLabel = new javax.swing.JLabel();
    NHUbpttSpinner = new javax.swing.JSpinner();
    TIMESTEPSbpttLabel = new javax.swing.JLabel();
    TIMESTEPSbpttSpinner = new javax.swing.JSpinner();
    CalculateButton = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    jMenuBar1 = new javax.swing.JMenuBar();
    FILEmenu = new javax.swing.JMenu();
    NewMenuItem = new javax.swing.JMenuItem();
    CalculateMenuItem = new javax.swing.JMenuItem();
    jSeparator1 = new javax.swing.JPopupMenu.Separator();
    ExitMenuItem = new javax.swing.JMenuItem();
    HELPmenu = new javax.swing.JMenu();
    HowMenuItem = new javax.swing.JMenuItem();
    jSeparator2 = new javax.swing.JPopupMenu.Separator();

```

```

AboutMenuItem = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jPanel1.setLayout(null);

TrainButton.setText("Insert Train Data");
TrainButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        TrainButtonActionPerformed(evt);
    }
});
jPanel1.add(TrainButton);
TrainButton.setBounds(20, 130, 190, 30);

TrainCheck.setText("Train Data");
TrainCheck.setEnabled(false);
TrainCheck.setHideActionText(true);
jPanel1.add(TrainCheck);
TrainCheck.setBounds(220, 130, 130, 23);

DesiredTrainButton.setText("Insert Desired Train Data");
DesiredTrainButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        DesiredTrainButtonActionPerformed(evt);
    }
});
jPanel1.add(DesiredTrainButton);
DesiredTrainButton.setBounds(20, 180, 190, 30);

DesiredTrainCheck.setText("Train Desired Data");
DesiredTrainCheck.setEnabled(false);
DesiredTrainCheck.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        DesiredTrainCheckActionPerformed(evt);
    }
});
jPanel1.add(DesiredTrainCheck);
DesiredTrainCheck.setBounds(220, 180, 130, 23);

TestButton.setText("Insert Test Data");
TestButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        TestButtonActionPerformed(evt);
    }
});
jPanel1.add(TestButton);
TestButton.setBounds(20, 230, 190, 30);

TestCheck.setText("Test Data");
TestCheck.setEnabled(false);
jPanel1.add(TestCheck);
TestCheck.setBounds(220, 230, 130, 23);

DesiredTestButton.setText("Insert Desired Test Data");
DesiredTestButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        DesiredTestButtonActionPerformed(evt);
    }
});

```

```

});
jPanel1.add(DesiredTestButton);
DesiredTestButton.setBounds(20, 280, 190, 30);

DesiredTestCheck.setText("Test Desired Data");
DesiredTestCheck.setEnabled(false);
DesiredTestCheck.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        DesiredTestCheckActionPerformed(evt);
    }
});
jPanel1.add(DesiredTestCheck);
DesiredTestCheck.setBounds(220, 280, 130, 23);

LRrbfLabel.setText("LR");

LClabel.setText("LC");

LSlabel.setText("LS");

buttonGroup1.add(RBFRadio);
RBFRadio.setSelected(true);
RBFRadio.setText("RBF");
RBFRadio.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        RBFRadioActionPerformed(evt);
    }
});

buttonGroup1.add(RBFsomRadio);
RBFsomRadio.setText("RBF SOM");
RBFsomRadio.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        RBFsomRadioActionPerformed(evt);
    }
});

NOClable.setText("Number of centers");

EpochsLabel.setText("Epochs");

EPOCHSRbfSpinner.setModel(new javax.swing.SpinnerNumberModel(100, 10, 5000, 1));

LRrbfSpinner.setModel(new javax.swing.SpinnerNumberModel(0.1d, 0.1d, 2.0d, 0.1d));

LCrbfSpinner.setModel(new javax.swing.SpinnerNumberModel(0.1d, 0.1d, 2.0d, 0.1d));

LSrbfSpinner.setModel(new javax.swing.SpinnerNumberModel(0.1d, 0.1d, 2.0d, 0.1d));

NOCrbfSpinner.setModel(new javax.swing.SpinnerNumberModel(10, 5, 25000, 1));

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(34, 34, 34)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addGap(34, 34, 34)
                    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

G)

```

        .addGroup(jPanel2Layout.createSequentialGroup())
        .addComponent(LRrbfLabel)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(LRrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(22, 22, 22)
        .addComponent(NOCLabel)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(NOCrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE, 79,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(jPanel2Layout.createSequentialGroup())
        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(LCLabel)
            .addComponent(LSLabel))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup())
            .addComponent(LCrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(77, 77, 77)
            .addComponent(EpochsLabel)
            .addGap(18, 18, 18)
            .addComponent(EPOCHSrbfSpinner, javax.swing.GroupLayout.DEFAULT_SIZE,
71, Short.MAX_VALUE))
            .addGroup(jPanel2Layout.createSequentialGroup())
            .addComponent(LSrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(66, 66, 66)
            .addComponent(RBFRadio)
            .addGap(8, 8, 8)
            .addComponent(RBFsomRadio))))
        .addGap(58, 58, 58))
    );
    jPanel2Layout.setVerticalGroup(
        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup())
        .addGap(57, 57, 57)
        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(LRrbfLabel)
                .addComponent(LRrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(NOCLabel)
                .addComponent(NOCrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup())
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(EpochsLabel)

```

```

        .addComponent(EPOCHSRbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(RBFRadio)
            .addComponent(RBFsomRadio)))
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(LCrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(LCLabel))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(LSrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(LSLabel)))
            .addGap(18, 18, 18))
    );

    TabbedAlgorithm.addTab("RBF/RBF SOM", jPanel2);

    LRbpttLabel.setText("LR");

    LRbpttSpinner.setModel(new javax.swing.SpinnerNumberModel(0.1d, 0.1d, 2.0d, 0.1d));

    EPOCHSbpttLabel.setText("Epochs");

    EPOCHSbpttSpinner.setModel(new javax.swing.SpinnerNumberModel(100, 10, 5000, 1));

    NHUbpttLabel.setText("Number Of Hidden Units");

    NHUbpttSpinner.setModel(new javax.swing.SpinnerNumberModel(15, 10, 50, 1));

    TIMESTEPSbpttLabel.setText("Time Steps");

    TIMESTEPSbpttSpinner.setModel(new javax.swing.SpinnerNumberModel(10, 5, 5000, 1));

    javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
    jPanel4.setLayout(jPanel4Layout);
    jPanel4Layout.setHorizontalGroup(
        jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addComponent(LRbpttLabel)
                        .addComponent(NHUbpttLabel)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(NHUbpttSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(LRbpttSpinner, 0, 0, Short.MAX_VALUE))
                        .addGap(6, 6, 6))
                    .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(LRbpttSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(LSLabel)))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(LCrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(LCLabel))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(LSrbfSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(LSLabel)))
            .addGap(18, 18, 18))
    );

```

```

        .addComponent(TIMESTEPSbpttLabel)
        .addComponent(EPOCHSbpttLabel))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
G)
        .addComponent(TIMESTEPSbpttSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
57, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(EPOCHSbpttSpinner, javax.swing.GroupLayout.PREFERRED_SIZE, 57,
Short.MAX_VALUE))
        .addGap(72, 72, 72))
    );
    jPanel4Layout.setVerticalGroup(
        jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel4Layout.createSequentialGroup()
            .addGap(48, 48, 48)
            .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
G)
                .addGroup(jPanel4Layout.createSequentialGroup()
                    .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                        .addComponent(LRbpttSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(LRbpttLabel))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                            .addComponent(NHUbpttSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(NHUbpttLabel)))
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addComponent(TIMESTEPSbpttSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS
ELINE)
                            .addComponent(EPOCHSbpttSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
20, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(EPOCHSbpttLabel)))
                        .addComponent(TIMESTEPSbpttLabel))
                    .addContainerGap(53, Short.MAX_VALUE))
                );
        TabbedAlgorithm.addTab("BPTT", jPanel4);

    jPanel1.add(TabbedAlgorithm);
    TabbedAlgorithm.setBounds(360, 130, 360, 180);

    CalculateButton.setText("CALCULATE AND EXPORT REPORT");
    CalculateButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            CalculateButtonActionPerformed(evt);
        }
    });
    jPanel1.add(CalculateButton);
    CalculateButton.setBounds(260, 370, 260, 30);

    jLabel1.setIcon(new javax.swing.ImageIcon(("gas.jpg"))); // NOI18N
    jPanel1.add(jLabel1);
    jLabel1.setBounds(0, -20, 750, 580);

```



```

FILEmenu.setText("FILE");

NewItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_
N, java.awt.event.InputEvent.CTRL_MASK));
NewItem.setText("NEW");
NewItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        NewMenuItemActionPerformed(evt);
    }
});
FILEmenu.add(NewMenuItem);

CalculateMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.
VK_Q, java.awt.event.InputEvent.CTRL_MASK));
CalculateMenuItem.setText("CALCULATE");
CalculateMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        CalculateMenuItemActionPerformed(evt);
    }
});
FILEmenu.add(CalculateMenuItem);
FILEmenu.add(jSeparator1);

ExitMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_E
, java.awt.event.InputEvent.CTRL_MASK));
ExitMenuItem.setText("EXIT");
ExitMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ExitMenuItemActionPerformed(evt);
    }
});
FILEmenu.add(ExitMenuItem);

jMenuBar1.add(FILEmenu);

HELPmenu.setText("HELP");

HowMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_
H, java.awt.event.InputEvent.CTRL_MASK));
HowMenuItem.setText("HOW TO USE?");
HowMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        HowMenuItemActionPerformed(evt);
    }
});
HELPmenu.add(HowMenuItem);
HELPmenu.add(jSeparator2);

AboutMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_
_A, java.awt.event.InputEvent.CTRL_MASK));
AboutMenuItem.setText("ABOUT");
AboutMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        AboutMenuItemActionPerformed(evt);
    }
});
HELPmenu.add(AboutMenuItem);

```

```

jMenuBar1.add(HELPmenu);

setJMenuBar(jMenuBar1);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 749,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 414,
javax.swing.GroupLayout.PREFERRED_SIZE)
);
this.setResizable(false);
this.setTitle("GAS PREDICTION SOFTWARE");
this.setLocation(100, 100);
pack();

} // </editor-fold>
//Υλοποίηση των μεθόδων για τους ακροατές συμβάντων
//Εισαγωγή Αρχείου Train
private void TrainButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FileNameTrain=this.selectFile();
    this.TrainCheck.setSelected(true);
}
//Εισαγωγή Αρχείου Desired Train
private void DesiredTrainButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    FileNameDesiredTrain=this.selectFile();
    this.DesiredTrainCheck.setSelected(true);
}
//Εισαγωγή Αρχείου Desired Test
private void DesiredTestButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    FileNameDesiredTest=this.selectFile();
    this.DesiredTestCheck.setSelected(true);
}
//Εισαγωγή Αρχείου Train
private void TestButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FileNameTest=this.selectFile();
    this.TestCheck.setSelected(true);
}
private void DesiredTrainCheckActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
}
private void DesiredTestCheckActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
}

```

```

//Η λειτουργία του κουμπιού Calculate
private void CalculateButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    //ελεγχος για εισαγωγή αρχείων
    if(this.TestCheck.isSelected() && this.TrainCheck.isSelected()&& this.DesiredTestCheck.isSelected()
&& this.DesiredTrainCheck.isSelected()) {
        ps1.progressBar.setVisible(false);
        if(this.TabbedAlgorithm.getSelectedIndex()==0) {
            double LRrbf= Double.parseDouble(this.LRrbfSpinner.getValue().toString());
            double LCrbf= Double.parseDouble(this.LCrbfSpinner.getValue().toString());
            double LSrbf= Double.parseDouble(this.LSrbfSpinner.getValue().toString());
            int NOCrbf=Integer.parseInt(this.NOCrbfSpinner.getValue().toString());
            int ITNUM=Integer.parseInt(this.EPOCHSrbfSpinner.getValue().toString());
            //RBFsom
            if(RBFsomRadio.isSelected()) {

                ReadingData rE1 = new
                ReadingData(NOCrbf,this.FileNameTrain,this.FileNameTest,this.FileNameDesiredTrain,this.FileNameD
esiredTest);
                if(rE1.check()) {
                    try {
                        rE1.Read();
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                    RbfGas r21 = new RbfGas(rE1,LRrbf,LCrbf,LSrbf,ITNUM);
                    ps1.pb.setVisible(true);
                    WritingOut wr1= new WritingOut("RBF-SOM");
                    rep.setVisible(false);
                    r21.startRBF(wr1);
                    rep.setVisible(true);
                    rep.ReadAndShow(wr1.getFilename());
                }
            }

            //RBF
            else if(RBFRadio.isSelected()){
                ReadingData rE2 = new
                ReadingData(NOCrbf,this.FileNameTrain,this.FileNameTest,this.FileNameDesiredTrain,this.FileNameD
esiredTest);
                if(rE2.check()) {
                    try {
                        rE2.Read();
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    RbfGas r22 = new RbfGas(rE2,LRrbf,LCrbf,LSrbf,ITNUM);
                    ps1.pb.setVisible(true);
                    WritingOut wr2 = new WritingOut("RBF");
                    r22.startRBFforiginal(wr2);
                    rep.setVisible(true);
                    rep.ReadAndShow(wr2.getFilename());

                }
            }
        }
    }
}

```

```

    }
    //BACKPROP
    else if(this.TabbedAlgorithm.getSelectedIndex()==1) {
        double LRbptt= Double.parseDouble(this.LRbpttSpinner.getValue().toString());
        int NHUbptt=Integer.parseInt(this.NHUbpttSpinner.getValue().toString());
        int TIMESTEPSbptt=Integer.parseInt(this.TIMESTEPSbpttSpinner.getValue().toString());
        int ITNUMbptt=Integer.parseInt(this.EPOCHSbpttSpinner.getValue().toString());
        ReadingData rebptt = new
ReadingData(NHUbptt,1,this.FileNameTrain,this.FileNameTest,this.FileNameDesiredTrain,this.FileName
eDesiredTest);
        if(rebptt.check()) {
            try {
                rebptt.Read();
            } catch (IOException e) {
                JOptionPane.showMessageDialog(null,"Please Correct files");
                e.printStackTrace();
            }
            WritingOut wr3= new WritingOut("BPTT");
            RNNNetwork rn = new RNNNetwork(rebptt,LRbptt,TIMESTEPSbptt,ITNUMbptt);
            ps1.pb.setVisible(true);
            rn.execute(wr3);
            rep.setVisible(true);
            rep.ReadAndShow(wr3.getFilename());

        }
    }
    }
    else JOptionPane.showMessageDialog(null,"Please insert 4 files");
}

private void RBFRadioActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
}

private void RBFsomRadioActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
}

private void NewMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.LRrbfSpinner.setValue(0.1);
    this.LSrbfSpinner.setValue(0.1);
    this.LCrbfSpinner.setValue(0.1);
    this.NOCrbfSpinner.setValue(10);
    this.EPOCHSrbfSpinner.setValue(100);
    this.LRbpttSpinner.setValue(0.1);
    this.NHUbpttSpinner.setValue(15);
    this.TIMESTEPSbpttSpinner.setValue(10);
    this.EPOCHSbpttSpinner.setValue(100);
    this.DesiredTestCheck.setSelected(false);
    this.TestCheck.setSelected(false);
    this.TrainCheck.setSelected(false);
    this.DesiredTrainCheck.setSelected(false);
    this.FileNameDesiredTest="";
    this.FileNameDesiredTrain="";
    this.FileNameTest="";
    this.FileNameTrain="";
}

```

```

    }
    //Υλοποίηση των μενου
    private void CalculateMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.CalculateButton.doClick();
    }
    //εξοδος menu
    private void ExitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        System.exit(0);
    }
    //how to menu
    private void HowMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        HelpWindow h = new HelpWindow(this,true);
        h.setVisible(true);
        h.setTitle("HELP WINDOW");
    }
    //about menu
    private void AboutMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        JOptionPane.showMessageDialog(null,"This program created by Kyprianos Hatzidimitriou");
    }
    //Μέθοδος επιλογής αρχείου
    public String selectFile() {
        String fileName;
        JFileChooser chooser = new JFileChooser();
        TxtFileFilter filter = new TxtFileFilter();
        chooser.setFileFilter(filter);
        chooser.showOpenDialog(getParent());
        fileName = chooser.getSelectedFile().getAbsolutePath();
        System.out.println(fileName);
        return fileName;
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MainWindow().setVisible(true);
            }
        });
    }
    //δήλωση ιδιοτήτων MainWindow
    // Variables declaration - do not modify
    private javax.swing.JMenuItem AboutMenuItem;
    private javax.swing.JButton CalculateButton;
    private javax.swing.JMenuItem CalculateMenuItem;
    private javax.swing.JButton DesiredTestButton;
    private javax.swing.JCheckBox DesiredTestCheck;
    private javax.swing.JButton DesiredTrainButton;
    private javax.swing.JCheckBox DesiredTrainCheck;
    private javax.swing.JLabel EPOCHSbpttLabel;
    private javax.swing.JSpinner EPOCHSbpttSpinner;
    private javax.swing.JSpinner EPOCHSrbfSpinner;

```

```

private javax.swing.JLabel EpochsLabel;
private javax.swing.JMenuItem ExitMenuItem;
private javax.swing.JMenu FILEmenu;
private javax.swing.JMenu HELPmenu;
private javax.swing.JMenuItem HowMenuItem;
private javax.swing.JLabel LCLabel;
private javax.swing.JSpinner LCrbfSpinner;
private javax.swing.JLabel LRbpttLabel;
private javax.swing.JSpinner LRbpttSpinner;
private javax.swing.JLabel LRrbfLabel;
private javax.swing.JSpinner LRrbfSpinner;
private javax.swing.JLabel LSLabel;
private javax.swing.JSpinner LSrbfSpinner;
private javax.swing.JLabel NHUbpttLabel;
private javax.swing.JSpinner NHUbpttSpinner;
private javax.swing.JLabel NOCLabel;
private javax.swing.JSpinner NOCrbfSpinner;
private javax.swing.JMenuItem NewMenuItem;
private javax.swing.JRadioButton RBFRadio;
private javax.swing.JRadioButton RBFsomRadio;
private javax.swing.JLabel TIMESTEPSbpttLabel;
private javax.swing.JSpinner TIMESTEPSbpttSpinner;
private javax.swing.JTabbedPane TabbedAlgorithm;
private javax.swing.JButton TestButton;
private javax.swing.JCheckBox TestCheck;
private javax.swing.JButton TrainButton;
private javax.swing.JCheckBox TrainCheck;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JLabel jLabel1;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel4;
private javax.swing.JPopupMenu.Separator jSeparator1;
private javax.swing.JPopupMenu.Separator jSeparator2;
private String FileNameTest,FileNameTrain,FileNameDesiredTest,FileNameDesiredTrain;
private ProgressWindow ps1;
private ReportWindow rep;
    // End of variables declaration
}

```

HelpWindow.java

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * Help.java
 *
 * Created on Aug 18, 2010, 4:34:21 PM
 */

```

```
package GUI;
```

```
//Η κλάση HelpWindow θα είναι ένα παράθυρο διαλόγου γι αυτό και κληρονομεί την JDialog
```

```

public class HelpWindow extends javax.swing.JDialog {

    /** Creates new form Help */
    public HelpWindow(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    //αρχικοποίηση συστατικών γραφικής διεπαφής (κώδικας παραγμένος από Netbeans)
    private void initComponents() {
        //Δημιουργία συστατικών της φόρμας
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        OkButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

        jTextArea1.setColumns(20);
        jTextArea1.setEditable(false);
        jTextArea1.setFont(new java.awt.Font("Arial", 0, 12)); // NOI18N
        jTextArea1.setRows(5);
        jTextArea1.setText("In this section will discuss the usage of GPS tool. First you must insert two
pairs of train and test files.\n\nThis could be done by selecting:\n\n· Insert Train Data – for train
data\n\n· Insert Desired Train – for actual output of train data\n\n· Insert Test Data – for Test
data\n\n· Insert Desired Test – for actual output of Test data\n\nThis software can be train with three
different architectures:\n\nRBF Network\n\nParameters for this network:\n\nLR – Learning rate for
centres\n\nLC – Learning rate for coefficient\n\nLS – Learning rate for sigma\n\nNumber of
centres\n\nEpochs – Number of iterations\n\nHybrid SOM - RBF Network\n\nParameters for this
network:\n\nLR – Learning rate for centres\n\nLC – Learning rate for coefficient\n\nLS – Learning rate
for sigma\n\nNumber of centres\n\nEpochs – Number of iterations\n\nRecurrent Neural
Network\n\nParameters for this network:\n\nLR – Learning rate\n\nNumber of hidden units\n\nNumber
of time steps\n\nEpochs – Number of iterations");
        jScrollPane1.setViewportView(jTextArea1);

        OkButton.setText("OK");
        //προσθήκη ακροατή συμβάντων για το κουμπί Ok
        OkButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                OkButtonActionPerformed(evt);
            }
        });

        jLabel1.setFont(new java.awt.Font("Tahoma", 3, 48)); // NOI18N
        jLabel1.setText("HELP");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .add(jScrollPane1)
                        .add(OkButton)
                    )
                )
        );
    }
}

```

```

        .addGroup(layout.createSequentialGroup())
        .addGap(302, 302, 302)
        .addComponent(OkButton))
        .addGroup(layout.createSequentialGroup())
        .addGap(34, 34, 34)
        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 591,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup())
        .addGap(238, 238, 238)
        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 135,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(32, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
        .addGap(33, 33, 33)
        .addComponent(jLabel1)
        .addGap(18, 18, 18)
        .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 365,
Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(OkButton)
        .addContainerGap())
    );

    pack();
} // </editor-fold>

private void OkButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

/**
 * @param args the command line arguments
 */

//δήλωση ιδιοτήτων του HelpWindow
// Variables declaration - do not modify
private javax.swing.JButton OkButton;
private javax.swing.JLabel jLabel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextArea jTextArea1;
// End of variables declaration
}

```


Παράρτημα Ε

Παράρτημα Ε: Δεδομένα εκπαίδευσης και επαλήθευσης

E-1

Παράρτημα Ε:

Τα δεδομένα ελέγχου και επαλήθευσης που χρησιμοποιήθηκαν στην έρευνα αυτή λόγω του μεγάλου όγκου τους, βρίσκονται μόνο σε ηλεκτρονική μορφή.