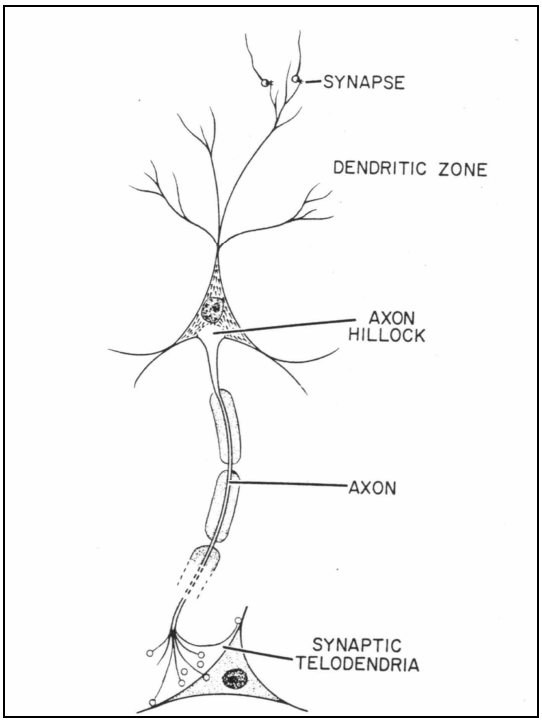
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΦΑΡΜΟΓΕΣ ΝΕΥΡΟΑΣΑΦΩΝ ΔΙΚΤΥΩΝ ΣΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΠΛΟΙΟΥ



από τον

ΜΠΟΜΠΟΡΗ ΣΤΑΥΡΟ

Καθηγητής: ΝΙΚΟΣ ΟΥΖΟΥΝΟΓΛΟΥ

ΠΕΡΙΛΗΨΗ

Το γνωστικό αντικείμενο αυτής της διπλωματικής είναι τα νευρωνικά δίκτυα τα ασαφή συστήματα, οι αλγόριθμοι τους καθώς και ο συνδυασμός τους: τα νευροασαφή συστήματα.

Παρουσιάζεται και αναπτύσσεται μια νέα τεχνική με την μορφή αλγορίθμου (MASMOD αλγόριθμος) η οποία εφαρμόζεται στο σύστημα του πλοίου.

Στο πρώτο μέρος της διπλωματικής παρουσιάζουμε την θεωρία νευρωνικών δικτύων. Ξεκινάμε με την πιο απλή (στοιχειώδη) μορφή νευρωνικού δικτύου το Perceptron και δίνουμε μια θεμελιώδη εφαρμογή του στην ταξινόμηση προτύπων. Στην συνέχεια ταξινομούμε όλα τα υπάρχοντα είδη νευρωνικών δικτύων και παρουσιάζουμε τα μονοεπίπεδα νευρωνικά δίκτυα (SLP) καθώς και τον αλγόριθμο μάθησης τους EMT (ελαχίστων μέσων τετραγώνων).

Στο 3ο κεφάλαιο έχουμε τα πολυεπίπεδα νευρωνικά δίκτυα (MLP) μαζί με την σημαντική ιδιότητα τους: της παγκόσμιας προσέγγισης και τον πιο γνωστό στην θεωρία NN (Neurals Networks) αλγόριθμο back - propagation. Στο 4ο κεφάλαιο παρουσιάζουμε τα δίκτυα ακτινικών συναρτήσεων βάσης (RBF) καθώς και τον τρόπο με τον οποίο εκπαιδεύονται.

Στο 5ο κεφάλαιο γίνεται η εισαγωγή και παρουσίαση αλγορίθμων που ανήκουν στην κατηγορία της μη επιβλεπόμενης μάθησης με σημαντικότερο από αυτούς τον LVQ.

Στο κεφάλαιο 6 αναφέρουμε το δίκτυο SOM (self organizing feature map - αυτοοργανούμενος χάρτης) ο οποίος χρησιμοποιεί ανταγωνιστικούς αλγορίθμους μη εποπτευόμενης δηλαδή μάθησης.

Στο 7ο κεφάλαιο αναπτύσσουμε τα ανατροφοδοτούμενα NN καθώς και τα αναδρομικά NN (RNN) με σημαντικότερο το δίκτυο Hopfield.

Στο 2ο μέρος της διπλωματικής αναπτύσσουμε την ασαφή λογική, τα ασαφή συστήματα και τα νευροασαφή συστήματα.

Ξεκινάμε (8ο κεφάλαιο) με την θεωρία ασαφών συνόλων τις πράξεις τους, τους τελεστές και το θεώρημα της επέκτασης καταλήγοντας με τον σημαντικό για την θεωρία ελέγχου κανόνα της ασαφούς σύνθεσης (max - min). Στην συνέχεια παρουσιάζονται τα ασαφή συστήματα ταξινομούνται και

αναλύονται με τους απαραίτητους μαθηματικούς ορισμούς. Τέλος εισάγουμε τα νευροασαφή συστήματα και τα ταξινομούμε.

Στο τρίτο μέρος της διπλωματικής αναλύεται εκτενώς η νέα τεχνική των Qiaug Gan και Chris J. Harris που βασίζεται σε αναδρομικά νευροασαφή δίκτυα και στον MASMOD αλγόριθμο. Γίνονται οι απαραίτητοι ορισμοί από την θεωρία συστημάτων και ελέγχου. Χρησιμοποιώντας τον MASMOD αλγόριθμο που αποτελεί το κομβικό σημείο της τεχνικής γραμμικοποιούμε και αναγνωρίζουμε το άγνωστο, μη-γραμμικό, δυναμικό και διακριτό σύστημα του πλοίου. Τα δεδομένα για την λειτουργία του αλγορίθμου (data pairs) τα παίρνουμε από μια μη γραμμική εξίσωση πλοίου η οποία είναι εκτενώς αναλυμένη.

Τέλος στο παράρτημα υπάρχει ο MASMOD αλγόριθμος με την μορφή MATLAB κώδικα.

SUMMARY

The subject of this diploma thesis is neural networks, fuzzy systems, their algorithms and their combination: neuro-fuzzy systems.

We present and elaborate a new technique in the form of an algorithm (MASMOD algorithm), which is applied on the vessel's system.

In the first part of this diploma thesis we present the theory of neural networks. We begin with the most simple (elementary) form of a neural network, Perceptron, and we provide one of its fundamental applications for pattern classification. Then we classify all existing types or neural networks and we present the single-level neural networks (SLP), as well as their EMT (least mean square) learning algorithm.

In the 3rd Chapter we have the multi-level neural networks (MLP) with their important property of global approach, and the most well-known back-propagation algorithm in the theory of neural networks. In the 4th Chapter we present the radial base function networks (RBF), and their training method.

In the 5th Chapter we introduce and present algorithms belonging to the non-supervised learning class, the most important of which is LVQ.

In Chapter 6 we discuss the SOM (self organizing feature map) network, which uses competitive algorithms, i.e. algorithms with non-supervised learning.

In the 7th Chapter we discuss the back-propagating NNs and the recursive NNs (RNN), the most important of which is the Hopfield network.

In the second part of this diploma thesis we discuss fuzzy logic, fuzzy systems, and neuro-fuzzy systems.

We begin (Chapter 8) with the theory of fuzzy sets, their operations, their operators, and the expansion theorem, rounding off with the important, for the control theory, rule of fuzzy composition (max-min). Then we present the fuzzy systems, and we classify and analyze them with all required mathematical terms and definitions. Finally, we introduce the neuro-fuzzy systems and we classify them.

In the third part of this diploma thesis we describe in detail the new technique of Qiaug Gan and Chris J. Harris, which is based on recursive neuro-fuzzy networks and on the MASMOD algorithm. We provide the required definitions from the systems and control theory. By using the MASMOD algorithm, which is the hub of this technique, we linearize and identify the unknown, non-linear, dynamic and distinct vessel system. The data pairs for the operation of the algorithm are obtained from a non-linear vessel equation, which is thoroughly analyzed.

Finally, in the Appendix we provide the MASMOD algorithm in the form of MATLAB code.

ΠΡΟΛΟΓΟΣ

Αντικείμενο αυτής της διπλωματικής είναι η αναγνώριση και γραμμικοποίηση του συστήματος του πλοίου με τη χρησιμοποίηση νευροασαφών συστημάτων και αλγορίθμων.

Στο πρώτο μέρος της διπλωματικής έχουμε μια εκτενή αναφορά στα νευρωνικά δίκτυα. Πρέπει να τονίσουμε ότι για τη συγγραφή αυτού του μέρους βασιστήκαμε ως επί το πλείστον στις σημειώσεις του Αριστείδη Λυκά: «Υπολογιστική Νοημοσύνη» προσθέτοντας πολλά άλλα στοιχεία.

Όλα τα βιβλία που χρησιμοποιήθηκαν για τη συγγραφή της Θεωρίας Νευρωνικών Δικτύων και των Ασαφών Συστημάτων αναφέρονται στη βιβλιογραφία.

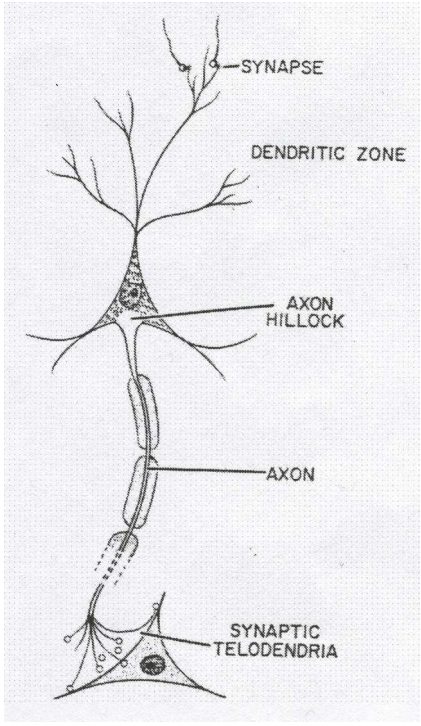
Στο δεύτερο μέρος έχουμε μια αναλυτική παρουσίαση της Ασαφούς λογικής, των Ασαφών συστημάτων και των νευροασαφών συστημάτων.

Τέλος, στο τρίτο μέρος της διπλωματικής αναλύουμε και εφαρμόζουμε την τεχνική που αναπτύχθηκε απ' τους Q. Jang και Harris, ενώ στο Παράρτημα μπορείτε να δείτε τον MASMOD αλγόριθμο σαν MATLAB κώδικα.

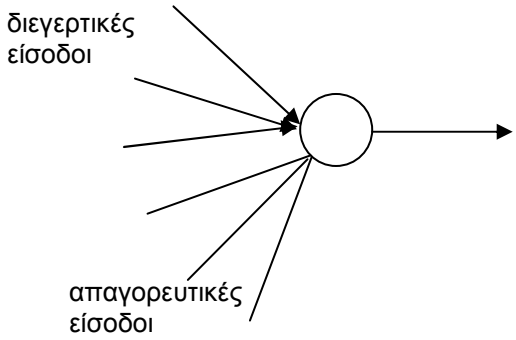
Μπόμπορης Σταύρος

ΕΙΣΑΓΩΓΗ

Τα τεχνητά νευρωνικά δίκτυα αποτελούν μια προσέγγιση της λειτουργίας του ανθρώπινου εγκεφάλου όσον αφορά τον τρόπο προσέγγισης της δομής των νευρώνων του. Όπως είναι γνωστό ο ανθρώπινος εγκέφαλος αποτελείται από πολλούς νευρώνες. Οι νευρώνες είναι εξειδικευμένα φυσικά κύτταρα τα οποία παρουσιάζουν προσαρμοστικότητα, ικανότητα αναγνώρισης, ανοχή στα λάθη, μεγάλη χωρητικότητα μνήμης όταν λειτουργούν ομαδικά, δηλαδή όταν δημιουργούν μία βιολογική αρχιτεκτονική. Στο παρακάτω σχήμα φαίνεται ο βιολογικός νευρώνας:



Όπως φαίνεται και στο σχήμα ο νευρώνας αποτελείται από το σώμα, τον άξονα, τους δενδρίτες και τις συνάψεις. Ένα μοντέλο του νευρώνα φαίνεται στο παρακάτω σχήμα:



Η δομή αυτή μπορεί να μην είναι ένα πλήρες και ακριβές μοντέλο ενός νευρικού κυττάρου αλλά προσεγγίζει τη λογική διαδικασία των δύο καταστάσεων που συμβαίνει στα νευρικά κύτταρα. Οι είσοδοι του νευρώνα μπορούν να χωριστούν σε δύο κατηγορίες: τις διεγερτικές και τις απαγορευτικές. Οι είσοδοι διέγερσης που συμβολίζονται με ένα βέλος τείνουν να διεγείρουν το κύτταρο κατά τρόπο αύξοντα μέχρι την τελική διέγερση. Οι απαγορευτικές είσοδοι που συμβολίζονται από μια σκέτη γραμμή μπορούν να απαγορεύσουν τη διέγερση του κυττάρου. Το μοντέλο αυτό εξελίχθηκε, και σήμερα υπάρχει μια ποικιλία αρχιτεκτονικών νευρωνικών δικτύων.

Οι τομείς εφαρμογής των NN (neural network) είναι οι εξής:

i) Στον αυτόματο έλεγχο: για αναγνώριση και γραμμικοποίηση συστημάτων καθώς και για έλεγχο.
ii) Στην τεχνητή νοημοσύνη (π.χ. υλοποίηση έμπειρων συστημάτων).
iii) Στην αναγνώριση φωνής, εικόνας, κειμένου, χειρονομιών, κ.ο.κ.
iv) Στην ιατρική (π.χ. ιατρική διάγνωση και επεξεργασία ιατρικής εικόνας).
v) Επικοινωνία ανθρώπου – υπολογιστή.
vi) Επεξεργασία εικόνας και μηχανική όραση.
vii) Στα δυναμικά εξελισσόμενα συστήματα.
viii) Αμυντικά συστήματα (π.χ. υποβρύχια ανίχνευση ναρκών).

και τέλος στην επεξεργασία σήματος π.χ. στην ανάλυση και μορφολογία του σεισμικού σήματος.

Πρέπει να τονίσουμε ότι τα NN έχουν υλοποιηθεί ηλεκτρονικά και έχει αναπτυχθεί βιβλιογραφία πάνω σ’ αυτόν τον τομέα (VLSI).

Οι πρώτοι που ασχολήθηκαν με τους νευρώνες είναι οι McCulloch και Pitts το 1943, οι οποίοι έφτιαξαν το αναπτυχθέν μοντέλο. Στη συνέχεια ο Von Neumman πήρε στοιχεία απ' την εργασία τους και τα χρησιμοποίησε στην κατασκευή του EDVAC, ενός από τους απογόνους του ENIAC. Το 1948 ο Wiener προσπάθησε να συνδέσει τη στατιστική μηχανική με τη μάθηση, κάτι που θα ολοκληρώσει αργότερα ο Hopfield. Το 1949 ο Hebb στο βιβλίο του «Οργάνωση της Συμπεριφοράς» υποστηρίζει ότι οι συνάψεις του εγκεφάλου τροποποιούνται καθώς ο άνθρωπος μαθαίνει. Το 1958 ο Rosenblat κάνει μια προσέγγιση στο πρόβλημα της αναγνώρισης προτύπων στη δουλειά του για το perceptron με το perceptron convergence theorem. Το 1960 οι Window και Hoff εισάγουν τον αλγόριθμο Least-Mem-Squares (LMS) που χρησιμοποιείται στην κατασκευή του Adaline και του Madaline. Στη δεκαετία του '80 επανήλθε το ενδιαφέρον πάνω στα NN με κορύφωση το 1986 όταν το Rumelhart διατύπωσε τον αλγόριθμο ανάστροφης διάδοσης σφάλματος (back-propagation) που έγινε ο δημοφιλέστερος αλγόριθμος εκπαίδευσης των MLP.

Κεφάλαιο 1
Perceptron

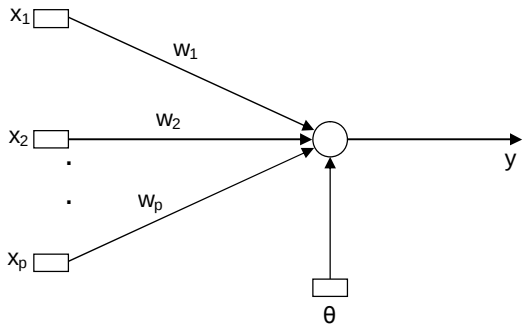
Εισαγωγή

Πολλές απ' τις επιστημονικές εφαρμογές του ανθρώπου βασίζονται σε φυσικά μοντέλα (αεροπλάνο, ρομπότ, ατμομηχανή κλπ.). Αυτή τη φορά μιμούμενος τη λειτουργία του ανθρώπινου εγκεφάλου που βασίζεται στους νευρώνες, ο άνθρωπος δημιούργησε τα νευρωνικά δίκτυα (Neural Networks).

Οι νευρώνες είναι εξειδικευμένα φυσικά κύτταρα του εγκεφάλου που διασυνδέονται μεταξύ τους. Έχουν εισόδους και εξόδους. Τα ηλεκτρικά σήματα που μεταφέρουν πληροφορίες μπαίνουν στις εισόδους των νευρώνων, υφίστανται επεξεργασία απ' αυτούς και παράγουν εξόδους. Οι έξοδοι του νευρώνα συνδέονται με τις εισόδους άλλων νευρώνων και το τελικό αποτέλεσμα είναι η σκέψη.

1.1. Perceptron

Βασιζόμενοι στον βιολογικό νευρώνα δημιουργήσαμε το Perceptron. Το Perceptron είναι ένα τεχνητό νευρωνικό δίκτυο (NN) που αποτελείται από έναν απλό νευρώνα ο οποίος δέχεται εξωτερικές εισόδους που συμμετέχουν με διαφορετικά βάρη και επιπλέον διεγείρονται από μία εξωτερική πόλωση. Η πόλωση μπορεί να θεωρηθεί σαν μια είσοδος σταθερής τιμής θ.



Σχήμα 1. Perceptron

Όπου x_1, x_2, … , x_p, θ είναι είσοδοι του νευρώνα. w_1, w_2, …, w_p τα βάρη και y η έξοδος. Η συνολική είσοδος του νευρώνα είναι $u = \sum_{i=1}^{p} w_1 x_i + \theta$ (υπολογισμός ενεργοποίησης).

Η έξοδος του νευρώνα (y) υπολογίζεται περνώντας την είσοδο (u) μέσα από μία συνάρτηση ενεργοποίησης (activation function) f: y=f(u).

Οι συναρτήσεις ενεργοποίησης που συνήθως χρησιμοποιούνται είναι μη γραμμικές ώστε να μπορούμε να προσομοιώσουμε μη γραμμικές διαδικασίες. Τέτοιες συναρτήσεις είναι οι σιγμοειδής. Υπάρχουν δύο είδη σιγμοειδών συναρτήσεων:

1) Η λογιστική: $f(u)=1/(1+\exp(-gu))$ που παρέχει τιμές στο (0,1). Η παράμετρος g είναι μια παράμετρος που καθορίζει την κλίση της σιγμοειδούς.

2) Η υπερβολική εφαπτομένη: f(u)=tanh(gu) η οποία παρέχει τιμές στο (-1,1).

Άλλες συναρτήσεις ενεργοποίησης είναι η γραμμική έξοδος $(f(u)=u)$, η συνάρτηση καμπάνας (bell function) και η βηματική συνάρτηση κατωφλίου φ.

Το εσωτερικό γινόμενο των εισόδων με τα βάρη (u) περνά από τη βηματική συνάρτηση φ(u) και παράγει μια έξοδο που είναι +1 αν η έξοδος είναι θετική και -1 αν είναι αρνητική.

Το ερώτημα που τίθεται σ' αυτό το σημείο είναι: Ποια πρέπει να είναι η τιμή των βαρών ώστε για δεδομένες εισόδους να έχουμε συγκεκριμένη έξοδο;

Ο τρόπος με τον οποίο μεταβάλλονται τα βάρη ενός ΝΝ καθορίζεται από έναν αλγόριθμο μάθησης. Ο αλγόριθμος μάθησης δεν είναι τίποτα άλλο από μια διαδοχή ενεργειών που επιδρούν στο ΝΝ ώστε να έχουμε επιθυμητό αποτέλεσμα (εκπαίδευση του ΝΝ).

Ο αλγόριθμος εκπαίδευσης του Perceptron αναπτύχθηκε απ' τον Rosenblatt και είναι γνωστός ως Αλγόριθμος Σύγκλισης του perceptron.

Παρακάτω δίνουμε αυτόν τον αλγόριθμο.

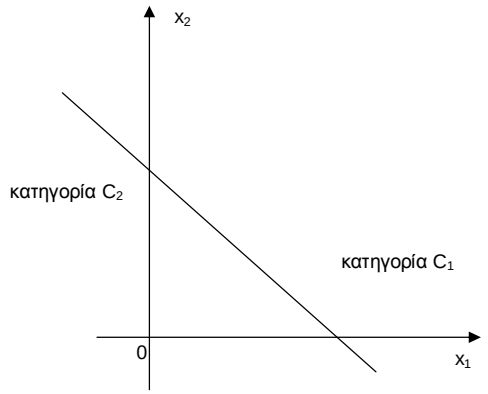
<u>Μεταβλητές και παράμετροι</u>

- $x(n)=(-1, x_1(n), x_2(n), \cdots xp(n))^T$

  Διάνυσμα εισόδου

- $w(n)= \left(\theta(n), w_1(n), w_2(n), \cdots, w_p(n)\right)^T$

  διάνυσμα παραμέτρων
- θ(n)=πόλωση
- y(n)=έξοδος
- d(n)=επιθυμητή έξοδος π.χ. $d(n) = \begin{cases} +1, \text{αν } x(n) \in C_1 \\ -1, \text{αν } x(n) \in C_2 \end{cases}$



Σχ. 2. Γραμμική διαχωρισιμότητα δύο διαστάσεων,
για πρόβλημα δύο κατηγοριών

- n = ρυθμός μάθησης
- α = θετική σταθερά μικρότερη της μονάδας

1. Αρχικοποίηση: θέτω w(0)=0.

Κατόπιν εκτελώ τους ακόλουθους υπολογισμούς για χρονικά βήματα n=1,2,…

2. Ενεργοποίηση: Τη χρονική στιγμή n ενεργοποιούμε το perceptron εφαρμόζοντας ως είσοδο σε αυτό κάποιο διάνυσμα εκπαίδευσης x(n) με επιθυμητή έξοδο d(n).

3. Υπολογισμός της εξόδου: Υπολογίζουμε την έξοδο του perceptron με την υπόθεση ότι έχουμε χρησιμοποιήσει σαν συνάρτηση ενεργοποίησης τη βηματική συνάρτηση $\text{sgon}(u) = \begin{cases} +1, \text{αν } u > 0 \\ -1, \text{αν } u < 0 \end{cases}$. Έχουμε $y(n) = \text{sgm}\left[w^T(n)x(n)\right]$.

4. Προσαρμογή των παραμέτρων: Μεταβολή της τιμής των βαρών σύμφωνα με την εξίσωση: $w(n+1) = w(n) + n\left[d(n) - y(n)\right]x(n)$

5. Αύξηση του χρόνου n κατά μια μονάδα, και άλμα στο Βήμα 2.

1.2. Εφαρμογή

Το perceptron χρησιμοποιείται για την ταξινόμηση προτύπων τα οποία πρέπει να είναι γραμμικά διαχωρίσιμα, δηλαδή να υπάρχει ένα υπερεπίπεδο που να διαχωρίζει τα πρότυπα δύο διαφορετικών κατηγοριών. Είναι περιορισμένο να εκτελεί ταξινομήσεις προτύπων που ανήκουν σε δύο μόνο κατηγορίες. Σκοπός του perceptron είναι να ταξινομεί την είσοδο $x=(x_1,x_2,\ldots,x_p)$ σε μια από τις δύο κατηγορίες (C_1ή C_2) (βλ. Σχ. 2). Ο κανόνας της ταξινόμησης είναι να απονεμηθεί η έξοδος x στην κατηγορία C_1 αν η έξοδος y είναι +1 και στην κατηγορία C_2 αν είναι –1.

Για την κατανόηση της συμπεριφοράς ενός ταξινομητή προτύπων, συνηθίζεται η γραφική αναπαράσταση των τελικών περιοχών απόφασης που προκύπτουν. Στην περίπτωση του perceptron, υπάρχουν δύο περιοχές απόφασης διαχωριζόμενες από ένα υπερεπίπεδο που ορίζεται από τη σχέση:

$$u = 0 \Rightarrow \sum_{i=1}^{p} w_i x_i - \theta = 0 .$$

Στην περίπτωση δύο μεταβλητών εισόδου x_1 και x_2, το διαχωριστικό όριο παίρνει τη μορφή ευθείας γραμμής (βλ. Σχ. 2). Ένα σημείο (x_1,x_2) το οποίο βρίσκεται άνω της διαχωριστικής γραμμής ανήκει στην κατηγορία C_1. Η επίδραση της πόλωσης θ είναι παράλληλη μερική μετακίνηση του διαχωριστικούς ορίου.

Κεφάλαιο 2
Η μηχανή Adaline

2.1. Ταξινόμηση ΝΝ

Τα σημαντικότερα είδη ΝΝ είναι έξι και μπορούν να διαιρεθούν ανάλογα με τα χαρακτηριστικά τους σε διάφορες κατηγορίες. Μια διαίρεση μπορεί να γίνει σε ΝΝ που δέχονται δυαδικές ή συνεχείς εισόδους. Στη συνέχεια μπορούν να διαιρεθούν ανάλογα με το αν μαθαίνουν με εποπτευόμενη μάθηση ή όχι κλπ. Έτσι έχουμε την παρακάτω ταξινόμηση των ΝΝ.

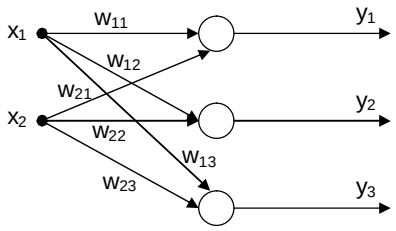
α) Δυαδική είσοδος
   1) Εποπτευόμενη μάθηση
      i) Δίκτυο Hopfield
      ii) Δίκτυο Hamming
   2) Μη εποπτευόμενη μάθηση
      i) Ταξινομητής Carpenter / Grossberg

β) Συνεχής είσοδος
   1) Εποπτευόμενη μάθηση
      i) Μόνο επίπεδα perceptron
         Single Layer Perceptron
      ii) Πολυεπίπεδα perceptron
         Multi Layer Perceptron (MLP)
   2) Μη εποπτευόμενη μάθηση
      i) NN SDM (Self Organizing Maps)

Ένα άλλο είδος ΝΝ είναι τα αναδρομικά νευρωνικά δίκτυα (Recurrent Neural Networks) RNN. Σ' αυτή την κατηγορία ανήκουν και τα δίκτυα Hopfield καθώς και ορισμένα από τα ανατροφοδοτούμενα (feedback) νευρωνικά δίκτυα.

2.2. Μονοεπίπεδα νευρωνικά δίκτυα (SLP)

Η γενική δομή ενός μονοεπίπεδου ΝΝ φαίνεται στο Σχήμα 4.



Σχήμα 3. SLP

Το παραπάνω NN αποτελείται από ένα επίπεδο 3 νευρώνων. Οι είσοδοι του NN x_1, x_2 συμμετέχουν με διαφορετικά βάρη (w_{11}, w_{12}, w_{13}) στις εισόδους των νευρώνων.

Ορίζουμε σαν ενέργεια E του NN: $E = -\frac{1}{2}\sum i \sum j w_{ij} x_i x_j$

2.3. Αλγόριθμος των ελαχίστων μέσων τετραγώνων (delta rule)

Ο αλγόριθμος των ΕΜΤ (least-mean square) εφαρμόζεται σε NN με ένα γραμμικό νευρώνα (Adaline). Η μηχανή Adaline είναι στην πραγματικότητα ένα perceptron με τη διαφορά ότι δεν υπάρχει βηματική συνάρτηση κατωφλίου στην έξοδο του νευρώνα. Η εξαγωγή του αλγόριθμου ΕΜΤ είναι μια επίπονη μαθηματική διαδικασία που αρχικά εξετάστηκε απ' τους Widrow και Hoff. (Βλέπε παράρτημα).

Ο αλγόριθμος των ΕΜΤ υλοποιείται με τα παρακάτω βήματα:

1. Αρχικοποίηση: θέτουμε

   $\hat{w}_\kappa(1) = 0$ για κ=1,2,…,p

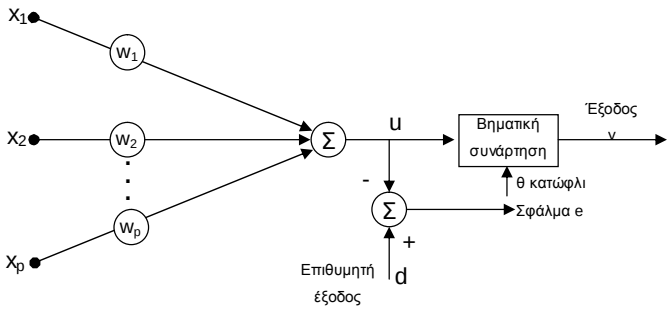
2. Σε κάθε βήμα n=1,2,… υπολόγισε

   $y(n) = \sum_{j=1}^{p} w_j(n) x_j(n)$

   $e(n)=d(n)-y(n)$

   $\hat{w}_\kappa(n+1) = \hat{w}_\kappa(n) + ne(n)x_\kappa(n)$ για κ=1,2,…,p

όπου $\hat{w}_\kappa$ το διάνυσμα βαρών του NN $x_j(n)$ οι είσοδοί του, y(n) η έξοδος, n ο ρυθμός μάθησης, d(n) η επιθυμητή έξοδος και e(n) το σφάλμα. Στο παρακάτω σχήμα δίνεται το διάγραμμα της μηχανής Adaline όπου φαίνονται και οι όροι του ΕΜΤ αλγόριθμου.



Σχήμα. 4. Μηχανή Adaline

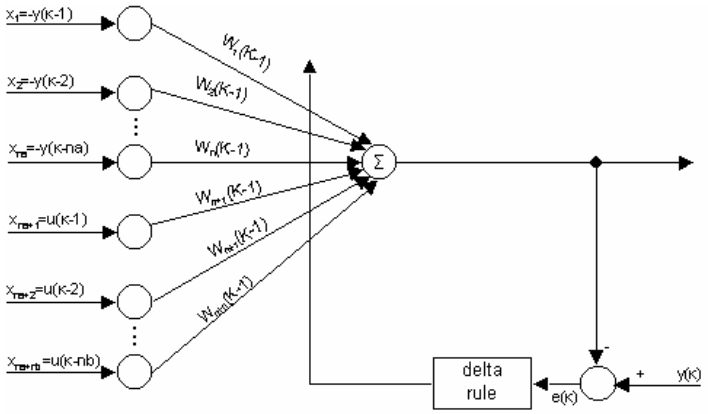
Η μηχανή Adaline χρησιμοποιεί τον αλγόριθμο ΕΜΤ για την εκπαίδευσή της. Μπορεί να χρησιμοποιηθεί για προσαρμοζόμενη ταξινόμηση προτύπων. Αποτελείται από μια μονάδα εσωτερικού γινομένου, μια βηματική συνάρτηση και έναν μηχανισμό ρύθμισης βαρών. Οι είσοδοι $x_1,x_2,\ldots,x_p$ της μονάδας εσωτερικού γινομένου έχουν τιμές είτε -1 είτε +1. Η βηματική συνάρτηση χαρακτηρίζεται από μια παράμετρο θ που ονομάζεται κατώφλι. Παίρνει τιμές μεταξύ 0 και 1. Για κάθε είσοδο x η Adaline εφοδιάζεται και με μια επιθυμητή απόκριση d η τιμή της οποίας είναι είτε -1 είτε +1. Τα βάρη $w_1,w_2,\ldots,w_p$ και το κατώφλι θ προσαρμόζονται σύμφωνα με τον αλγόριθμο ΕΜΤ. Συγκεκριμένα η έξοδος του γραμμικού αθροιστή u, που προκύπτει για εισόδους $x_1,x_2,\ldots,x_p$ αφαιρείται από την επιθυμητή απόκριση d και έτσι παράγεται το σφάλμα e. Το σφάλμα e χρησιμοποιείται για την υλοποίηση του αλγορίθμου ΕΜΤ. Η έξοδος της Adaline προκύπτει αφού περάσει η έξοδος u του γραμμικού αθροιστή από τη βηματική συνάρτηση. Δηλαδή $y = \begin{cases} +1 \text{ αν } u \geq \Theta \\ -1 \text{ αν } u < \Theta \end{cases}$. Ο στόχος της διαδικασίας μάθησης στην Adaline μπορεί να διατυπωθεί ως εξής:

Με δεδομένο ένα σύνολο προτύπων εισόδου με αντίστοιχες επιθυμητές εξόδους, βρείτε τις βέλτιστες τιμές βαρών $w_1,w_2,\ldots,w_p$ και κατωφλίου θ ώστε να ελαχιστοποιηθεί η πραγματική μέση τετραγωνική τιμή του σφάλματος en.

Εδώ πρέπει να σημειωθεί ότι οι διάφοροι αλγόριθμοι μάθησης υλοποιούνται με τον προγραμματισμό τους.

2.4. Εφαρμογή

Σ' αυτή την παράγραφο θα αναπτύξουμε μια εφαρμογή του αλγορίθμου ΕΜΤ για την αναγνώριση ενός συστήματος. Θα χρησιμοποιήσουμε ένα ΝΝ δύο επιπέδων και θα εφαρμόσουμε των ΕΜΤ αλγόριθμο. Στο σχήμα 5 φαίνεται το 2 επιπέδων ΝΝ που καλείται και νευρωνικός εκτιμητής του συστήματος (neural estimator).



Το επίπεδο εισόδου αποτελείται από na+nb στοιχεία. Αυτές οι na+nb είσοδοι είναι οι μετρούμενοι είσοδοι – έξοδοι του συστήματος που θέλουμε να αναγνωρίσουμε. Το επίπεδο εξόδου έχει μόνο ένα στοιχείο του οποίου η έξοδος $\hat{y}(\kappa)$ είναι η προβλεπόμενη έξοδος του συστήματος. Υ(κ) είναι η επιθυμητή έξοδος του συστήματος και η οποία αφαιρείται απ' την $\hat{y}(\kappa)$ και παράγεται το σφάλμα e(k). Μεταβάλλουμε τα βάρη του ΝΝ σύμφωνα με τον delta rule ώσπου το σφάλμα e(k) να ελαχιστοποιηθεί. Όταν συμβεί αυτό θα

έχουμε υπολογίσει τις τιμές των βαρών ώστε να έχουμε την επιθυμητή έξοδο (αναγνώριση).

Πιο αναλυτικά το διάνυσμα βαρών τη χρονική στιγμή κ ορίζεται ως εξής: $w(\kappa) = [w_1(\kappa) \ldots, w_{na}(\kappa), w_{n+1}(\kappa), \ldots, w_{na+nb}(\kappa)]$.

Ενημερώνεται χρησιμοποιώντας τον delta rule:

$w(\kappa+1) = w(\kappa) = \frac{ae(\kappa+1)x(\kappa)}{\varepsilon + x^T(\kappa)x(\kappa)}$ όπου x(κ) είναι διάνυσμα εισόδου του δικτύου και ορίζεται ως εξής: $x(\kappa) = [x_1(\kappa), \ldots, x_{na}(\kappa), x_{na+1}(\kappa), \ldots, x_{na+nb}(\kappa)]^T =$

$= [-y(\kappa), \ldots, -y(\kappa - na), u(\kappa), \ldots, u(k - nb)]^T$.

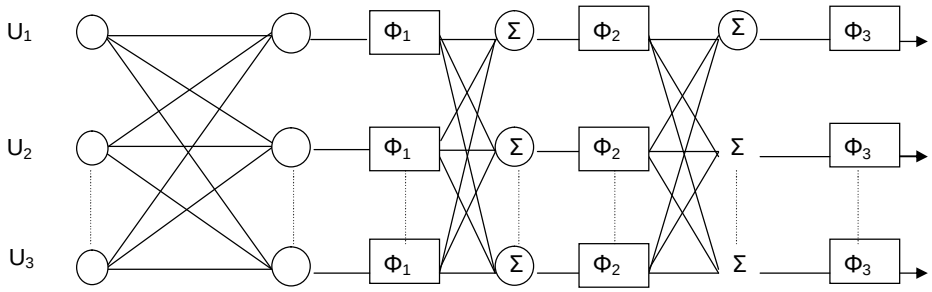
Το a $\in(0,2)$ είναι ένας απλοποιητικός παράγοντας. Η σταθερά ε έχει επιλεχθεί να είναι κοντά στο μηδέν και έχει συμπεριληφθεί για να αποφύγουμε τη διαίρεση με το μηδέν.

Όπως φαίνεται στο σχήμα 5 το σφάλμα e(κ) είναι η διαφορά μεταξύ πραγματικής απόκρισης του συστήματος y(κ) και της απόκρισης $\hat{y}(\kappa)$ του νευρωνικού μοντέλου. Το $\hat{y}(\kappa)$ δίνεται $\hat{y}(\kappa) = \sum_{i=1}^{na+nb} w_i(\kappa-1)x_i(\kappa-1)$ και ο προσαρμοστικός αλγόριθμος ελαχιστοποιεί το σφάλμα.

Κεφάλαιο 3
Πολυεπίπεδα Νευρωνικά Δίκτυα (MLP)

3.1. Δομή των MLP

Ένα πολυεπίπεδο NN (MLP) αποτελείται από ένα πλήθος νευρώνων που βρίσκονται ταξινομημένα σε επίπεδα τα οποία συνδέονται μεταξύ τους όπως φαίνεται στο σχήμα 7.



Σχήμα 7

Όπως φαίνεται στο παραπάνω σχήμα τα MLP συνήθως αποτελούνται από τρία επίπεδα: i) επίπεδο εισόδου ii) ένα ή περισσότερα κρυμμένα επίπεδα iii) επίπεδο εξόδου.

Τα MLP μπορούν να χαρακτηρισθούν και ως δίκτυα προσοτροφοδότησης (Feed forward Neural Networks).

Η εξίσωση που δίνει την έξοδο του MLP είναι η εξής:

$$y = \Phi_n(W_n\ \Phi_{n-1}... \Phi_1(W=1_u+b_1)+...+b_n)+b_n)$$

όπου W_i είναι η μήτρα βαρών του i επιπέδου, φ_i είναι η συνάρτηση ενεργοποίησης του i επιπέδου και b_i είναι οι τιμές κατωφλίου ή πόλωσης που αναφέρονται στον κάθε κόμβο του i επιπέδου. Η συνάρτηση φ_i είναι συνήθως μη γραμμική και πιο συγκεκριμένα σιγμοειδής: $\varphi(u) = \frac{1}{1+\exp(-u)}$

Άλλες συναρτήσεις που χρησιμοποιούνται είναι συνάρτηση κατωφλίου, η συνάρτηση υπερβολικής εφαπτομένης καθώς και η Gaussian συνάρτηση (Γκαουσιανή). Το γεγονός αυτό δίνει την δυνατότητα στα MLP να μπορούν να προσομοιάζουν μη γραμμικές διαδικασίες.

Στο σχήμα 7 τα MLP είναι πλήρους σύνδεσης δηλαδή όλες οι έξοδοι των νευρώνων του ενός επιπέδου συνδέονται σαν είσοδοι στο επόμενο επίπεδο. Υπάρχει δυνατότητα να συνδεθούν μόνο μία ομάδα εξόδων των νευρώνων του ενός επιπέδου με ορισμένες εισόδους του επόμενου.

Η ομοιότητα του MLP με τους βιολογικούς νευρώνες έγκειται στην μη γραμμικότητα της συνάρτησης ενεργοποίησης και η διαφορά τους είναι ότι οι βιολογικοί νευρώνες παρουσιάζουν διαφορετική συμπεριφορά μεταξύ τους ενώ οι τεχνητοί είναι όλοι όμοιοι μεταξύ τους και χρησιμοποιούν τον ίδιο τύπο μη γραμμικότητας.

Ένα ακόμα χαρακτηριστικό των MLP είναι ότι δεν υπάρχει σύνδεση μεταξύ νευρώνων του ίδιου επιπέδου καθώς επίσης δεν υπάρχει διασύνδεση μεταξύ νευρώνων που δεν ανήκουν σε διαδοχικά επίπεδα.

Τα ερωτήματα που πρέπει να απαντηθούν κατά την σχεδίαση ενός MLP είναι τα εξής:

i. Πόσα επίπεδα θα χρησιμοποιηθούν στο δίκτυο;
ii. Πόσους νευρώνες θα έχει το κάθε επίπεδο;
iii. Τι μη γραμμικές συναρτήσεις θα χρησιμοποιηθούν στους κόμβους

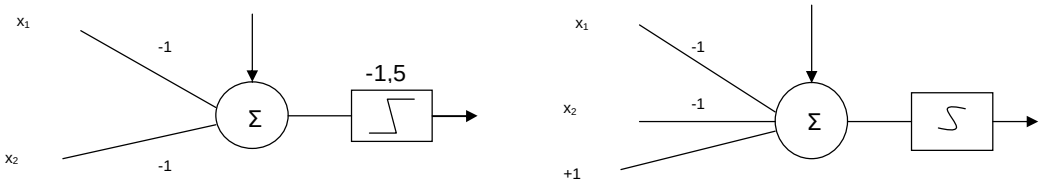
Απ' τη στιγμή που θα απαντηθούν οι παραπάνω ερωτήσεις πρέπει να γίνει επιλογή των βαρών του NN καθώς και του αλγόριθμου μάθησης.

Τα βάρη αρχικά επιλέγονται τυχαία ενώ ο αλγόριθμος μάθησης έχει ως στόχο να προσαρμόσει τα βάρη έτσι ώστε να μειωθεί το σφάλμα μεταξύ πραγματικής εξόδου του νευρωνικού και της επιθυμητής, σύμφωνα με κάποιο κριτήριο κόστους που έχει ορισθεί προηγουμένως. Το MLP εκπαιδεύεται με ένα σύνολο δεδομένων που προέρχεται συνήθως από το σύστημα που πρόκειται να ελεγχθεί. Απ' τη στιγμή που η δομή του δικτύου έχει επιλεγεί μεταβάλλουμε τα βάρη ώστε η σχέση εισόδου εξόδου του δικτύου να προσεγγίσει καλύτερα αυτή του συστήματος. Το MLP μετά απ' αυτή την διαδικασία μπορεί να χρησιμοποιηθεί σαν μοντέλο του συστήματος.

Ο αλγόριθμος που συνήθως επιλέγεται για την εκπαίδευση των MLP είναι ο αλγόριθμος της οπισθοδρομικής διάδοσης σφάλματος που είναι γνωστός σαν backpropagation. Πριν αναπτύξουμε και παρουσιάσουμε τον αλγόριθμο backpropagation ας δούμε μερικές εφαρμογές των MLP.

3.2. Εφαρμογή

Τα NN μπορούν να χρησιμοποιηθούν για ταξινόμηση (classification). Μπορεί εύκολα να αποδειχθεί ότι η πύλη nand υλοποιείται από μία γραμμική μονάδα κατωφλίου καθώς επίσης και από μία γραμμική μονάδα που χρησιμοποιεί σαν συνάρτηση ενεργοποίησης την σιγμοειδή (βλέπε σχήμα 8)



Σχήμα 8. Μια γραμμική μονάδα κατωφλίου που μπορεί να υλοποιήσει την πύλη nand.

Στον παρακάτω πίνακα βλέπουμε την υλοποίηση της nand με γραμμική μονάδα κατωφλίου και με την σιγμοειδή συνάρτηση.

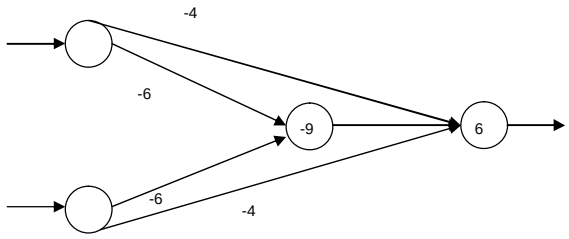
Input		Threshold unit		Σιγμοειδής συνάρτηση	
X_1	X_2	Άθροισμα	Output	Άθροισμα	Output
-1	-1	2	+1	3	+1
-1	+1	0	+1	1	+1
+1	-1	0	+1	1	+1
+1	-1	-2	-1	-1	-1

Η τιμή των βαρών είναι –1 ενώ η πόλωση παίρνει την τιμή –2/3. Στην υλοποίηση με την σιγμοειδή συνάρτηση οι τιμές των βαρών είναι πάλι –1 ενώ η πόλωση bias = 1.

Επειδή όλες οι λογικές συναρτήσεις μπορούν να συνθεθούν από την nand πύλη φτάνουμε στο ακόλουθο συμπέρασμα.

Όλες οι λογικές συναρτήσεις μπορούν να υλοποιηθούν με MLP που αποτελείται από γραμμικές μονάδες ή από σιγμοειδών συναρτήσεων μονάδες. Πιο συγκεκριμένα κάθε λογική συνάρτηση μπορεί να υλοποιηθεί

από ένα 3 επιπέδων MLP. Παρακάτω δίνουμε ένα παράδειγμα μιας τέτοιας υλοποίησης:



Σχήμα 8: ένα NN με ένα κρυμμένο επίπεδο υλοποιεί την XOR λογική λειτουργία. Οι τιμές μέσα στους κύκλους είναι η πόλωση ενώ οι λοιπές είναι βάρη.

Input		Threshold unit		Σιγμοειδής συνάρτηση	
X_1	X_2	Άθροισμα	Output	Άθροισμα	Output
-1	-1	14	+1	5	+1
-1	+1	2	+1	-3	-1
+1	-1	2	+1	-3	-1
+1	+1	-10	-1	+7	+1

Πίνακας 1: Η υλοποίηση της XOR από ένα NN με ένα κρυμμένο επίπεδο.

3.3. Προσέγγιση συναρτήσεων

Μία πολύ σημαντική ιδιότητα των πολυεπίπεδων προσοτροφοδοτούμενων νευρωνικών δικτύων (MLP) είναι η παγκόσμια προσέγγιση (universal approximation). Αυτό σημαίνει ότι ένα MLP μπορεί να προσεγγίσει οποιαδήποτε συνάρτηση με οποιαδήποτε ακρίβεια. Αυτό προϋποθέτει την ύπαρξη ενός ή περισσότερων κρυμμένων επιπέδων καθώς και την χρήση μη γραμμικών συναρτήσεων σαν συναρτήσεις ενεργοποίησης των νευρώνων.

Ας θεωρήσουμε τα σημεία του n-διάστατου Ευκλείδειου χώρου R^n με το $x=(x_1, x_2, ...x_n)$ και ορίζουμε την νόρμα του χ me thn $|x| = \left(\sum_{i=o}^{n} x_o^2 \right)^{\frac{1}{2}}$

Θεώρημα

Έστω φ(x) μία μη σταθερή, περιορισμένη και μονότονα αυξανόμενη συνεχής συνάρτηση, K είναι ένα συμπαγές υπομοντέλο του R^n και $f(x_1, x_2,...,x_n)$ είναι μία πραγματικής τιμής συνεχής συνάρτηση K. κατόπιν, για ένα $\in>0$, υπάρχει ένας ακέραιος N και πραγματικές σταθερές c_i, θ_i (i=1,2,...N), W_{ij} (i=1,2...N, j=1,2,...N) έτσι ώστε:

Εξίσωση 1^A $\hat{f}(x_1, x_2,..., x_n) = \sum_{i=1}^{N} ci\varphi\left(\sum_{i=1}^{N} w_{ij} x_j - \theta_i\right)$ να ικανοποιεί

$\max_{x\in K}|f(x_1, x_2,...x_n) - \hat{f}(x_1,x_2,...x_n)| < \in$.

Στο δεύτερο μέλος της εξίσωσης 1 η φ(·) είναι η συνάρτηση ενεργοποίησης των κρυμμένων μονάδων. Αν c_i είναι ένα βάρος της κρυμμένης μονάδας I προς την μονάδα εξόδου τότε η $\hat{f}(\cdot)$ αναπαριστά την απεικόνιση απ' την είσοδο στην έξοδο. Τα παραπάνω μας λένε ότι ένα τριών επιπέδων δίκτυο με σιγμοειδείς κρυμμένες μονάδες μπορεί να προσεγγίσει τις συνεχείς συναρτήσεις με μεγάλη ακρίβεια.

3.3. Ο αλγόριθμος back-propagation

Η εκπαίδευση των MLP γίνεται με τον αλγόριθμο οπισθοδρομικής διάδοσης σφάλματος (error back propagation).

Μπορεί να θεωρηθεί σαν μία γενίκευση του αλγορίθμου EMT (least mean squares). Η διαδικασία εκπαίδευσης στον back propagation περιλαμβάνει υπολογισμούς που υλοποιούνται σε δύο περάσματα μέσω των επιπέδων του δικτύου: ένα κατά την ευθεία φορά και ένα κατά την αντίστροφη φορά (από την έξοδο προς την είσοδο).

Στην εφαρμογή του αλγόριθμου back propagation η εκπαίδευση επιτυγχάνεται η εκπαίδευση επιτυγχάνεται με την παρουσίαση στο δίκτυο ενός συνόλου παραδειγμάτων εκπαίδευση. Η εκπαίδευση μπορεί να είναι ανά πρότυπο (on line) ή off-line. Στην on-line εκπαίδευση έχουμε μεταβολή των βαρών του δικτύου για κάθε νέο πρότυπο που παρουσιάζεται. Στην off-line εκπαίδευση τα βάρη ενημερώνονται (μεταβάλλονται) έπειτα από την παρουσίαση στο δίκτυο όλου του συνόλου προτύπων.

Τα βήματα που ακολουθούμε για την υλοποίηση του αλγορίθμου back-propagation (on line) είναι τα εξής:

1. Αρχικοποίηση: Ορισμός της αρχιτεκτονικής του δικτύου και αρχικοποίηση των βαρών των συνδέσεων με μικρές τυχαίες τιμές ομοιόμορφα κατανεμημένες. (συνήθως στο διάστημα [-1,1]).
2. Παρουσίαση των προτύπων εκπαίδευσης. Παρουσίασε στο δίκτυο ένα σύνολο (εποχή) προτύπων εκπαίδευσης. Για κάθε πρότυπο του συνόλου εκτέλεσε τα στάδια 3 και 4 που ακολουθούν.
3. Υπολογισμοί ευθείας φοράς. Θεωρούμε το πρότυπο εκπαίδευσης [x(n), d(n)] όπου x(n) είναι το διάνυσμα εφαρμογής στην είσοδο του δικτύου και d(n) είναι το επιθυμητό διάνυσμα εξόδου του δικτύου. Υπολόγισε τα σήματα που προκύπτουν ανά επίπεδο. Το σημείο ενεργοποίησης του νευρώνα j του επιπέδου 1 είναι:

$$u_m^{\ell}(n) = \sum_{i=0}^{P} W_{ji}^{(\ell)}(n) y_i^{(\ell-1)}(n)$$

όπου $y_i^{(P-1)}(n)$ είναι το σήμα εξόδου του νευρώνα i του προηγούμενου επιπέδου ℓ-1, κατά την επανάληψη n και $W_{ji}(n)$ είναι το βάρος της σύναψης του νευρώνα j στο επίπεδο ℓ που τροφοδοτείται από το νευρώνα i του επιπέδου ℓ-1. Για i=0 έχουμε y_0=-1 και w_{j0}=θj όπου θj είναι η τιμή πόλωσης. Χρησιμοποιούμε την λογιστική συνάρτηση για τον νευρώνα J του επιπέδου ℓ

$$j_j^{\ell}(n) = \frac{1}{1+\exp(-u_j^{\ell}(n))}$$

Αν ο νευρώνας j ανήκει στο πρώτο κρυμμένο επίπεδο (ℓ=1) θέστε

$$y_j^0(n)) = x_j(n)$$

όπου $x_j(n)$ είναι το j-οστό στοιχείο του διανύσματος εισόδου x(n). Αν ο νευρώνας j ανήκει στο επίπεδο εξόδου (ℓ=L) θέστε

$$y_j^L(n)) = O_j(n)$$

Έτσι υπολογίστε το σήμα σφάλματος

$$e_j(n) = d_j(n) - O_j(n)$$

όπου $d_j(n)$ είναι το j-οστό στοιχείο του επιθυμητού διανύσματος d(n).

4. Υπολογισμοί αντίστροφης φοράς. Υπολογίστε τα δ (τοπικές κλίσεις) προχωρώντας αντίστροφα στο δίκτυο ανά επίπεδο: (L)

$$\delta_j(n) = e_j(n)O_j(n)[1-O_j(n)]$$

$$\delta_j^{\ell}(n) = y_j^{\ell}(n)[1-y_j^{\ell}(n)]\sum_k \delta_k^{(\ell+1)}(n)w_{kj}^{(\ell+1)}(n)$$

Προσαρμόστε τα βάρη του επιπέδου ℓ σύμφωνα με τον κανόνα δέλτα

$$w_{ji}^{(\ell)}(n+1) = w_{ji}^{(\ell)}(n) + \alpha[1-w_{ji}^{(\ell)}(n) - w_{ji}^{(\ell)}(n-1)] + n\delta^{(\ell)}(n)y_j^{(\ell-1)}(n)$$

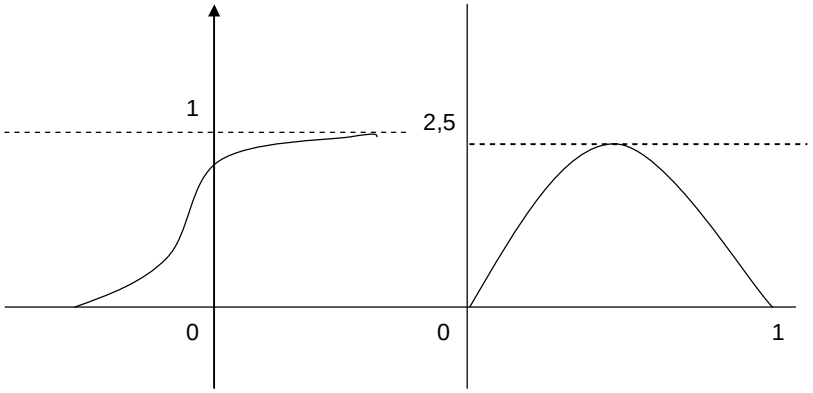
όπου n είναι ο ρυθμός μάθησης και α είναι σταθερά Momentum (όρος ορμής συνήθως ένας θετικός σταθερός αριθμός).

5. Επανάληψη. Επανάλαβε τους υπολογισμούς για νέες εποχές προτύπων έως ότου οι παράμετροι του δικτύου σταθεροποιηθούν και η μέση τιμή σφάλματος γίνει ελάχιστη. Η σειρά παρουσίασης των προτύπων σε κάθε εποχή πρέπει να είναι τυχαία. Η σταθερά α και ο ρυθμός μάθησης n συνήθως μειώνονται με το πέρασμα των εποχών.

Όπως αναφέραμε παραπάνω η πιο συχνά χρησιμοποιημένη συνάρτηση ενεργοποίησης είναι η σιγμοειδής. Είναι μη ελαττούμενη και μπορεί να περιορίσει την έξοδο στην περιοχή (0,1). Προκειμένου να εφαρμόσουμε τον αλγόριθμο back propagation πρέπει να ξέρουμε την παράγωγο της σιγμοειδούς. Αυτή είναι:

$$\frac{dO_j(n)}{du_j(n)} = O_j(n)(1-O_j(n))$$

Στο σχήμα 1D παρουσιάζουμε γραφικά την σιγμοειδή και την παράγωγό της. Η παράγωγος της σιγμοειδούς εμφανίζεται στο 4^{ο} βήμα κατά τον υπολογισμό του $\delta_j(n)$.



α) Η σιγμοειδής α) Η παράγωγός της

Είναι άξιο να σημειωθεί ότι η παράγωγος της σιγμοειδούς παίρνει την μέγιστη τιμή της όταν η σιγμοειδής είναι γύρω στο 0,5. Το ποσό μεταβολής των βαρών είναι ανάλογο της παραγώγου. Γι' αυτό κατά τη διάρκεια της μάθησης τα βάρη αλλάζουν περισσότερο όταν η σχετική μονάδα έχει τιμή εξόδου 0,5. Αυτό συμβάλλει στην σταθερότητα της μάθησης του νευρωνικού συστήματος.

Ας δώσουμε μερικά στοιχεία για τον ρυθμό (rate) μάθησης n. Μικρές τιμές του n σημαίνουν πολλούς κύκλους μάθησης (επαναλήψεις) ενώ μεγάλες τιμές του n μπορεί να προκαλέσουν ταλάντωση (oscillation). Δεν είναι εύκολο να βρεις τον κατάλληλο n με τον οποίο η μάθηση να είναι όσο το δυνατό πιο γρήγορη χωρίς ταυτόχρονα να παρουσιάζεται ταλάντωση. Για να ομαλοποιήσουμε την ταλάντωση εισάγουμε τον όρο ορμής α (momentum term). Έτσι δίνουμε την δυνατότητα για μεγάλα βήματα αλλαγής βαρών.

Η μη γραμμική ιδιότητα των MLP μας δίνει τη δυνατότητα όπως αναλύσαμε στην παράγραφο 3.2 της προσέγγισης μη γραμμικών συναρτήσεων. Απ' την άλλη μεριά η μη γραμμικότητα των MLP που οφείλεται στις μη γραμμικές σιγμοειδείς συναρτήσεις μπορεί να προκαλέσει τον εγκλωβισμό του αλγορίθμου μάθησης σε τοπικά ελάχιστα. Το πιο σοβαρό πρόβλημα του back-propagation είναι ο ρυθμός σύγκλισης (convergence rate): για πολλές εφαρμογές η back-propagation διαδικασία μπορεί να πάρει μέρες για να συγκλίνει σε ένα σύνολο σωστών βαρών. Γι' αυτό το λόγο έχουν αναπτυχθεί πολλοί γρηγορότεροι αλγόριθμοι μάθησης που δεν διαφέρουν από τον back-propagation.

Πριν προχωρήσουμε στις εφαρμογές των NN θα εξετάσουμε διάφορους άλλους τύπους NN. Υπάρχουν και άλλοι τύποι MLP όπως τα regularations δίκτυα (κανονικοποιημένα δίκτυα) που είναι 3-επιπέδων και τα οποία δεν χρειάζονται αλγόριθμο μάθησης.

Κεφάλαιο 4

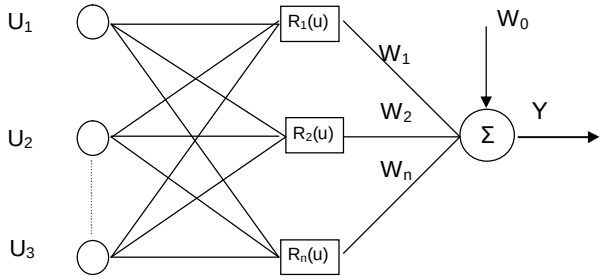
Δίκτυα ακτινικών συναρτήσεων Βάσης (RBF)

4.1. Δομή και ορισμός των RBF

Μία άλλη κατηγορία NN είναι τα δίκτυα ακτινικών συναρτήσεων βάσης (RBF). Ανήκουν στην γενική κατηγορία των προσοτροφοδοτούμενων NN και προτάθηκαν ως βελτίωση των MLP. Με μία πρώτη ματιά τα RBF φαίνονται πολύ διαφορετικά απ' τους προγόνους (MLP) τους αλλά αποκλίνουν πολύ και από την γενική έννοια του νευρωνικού συστήματος. Πράγματι οι μονάδες που συγκροτούν ένα δίκτυο RBF δεν έχουν καμία σχέση με το πρότυπο νευρώνα. Η μάθηση επίσης των δικτύων RBF είναι μία ολότελα διαφορετική διαδικασία. Παρ' όλα αυτά θεωρούμε ότι τα δίκτυα που χρησιμοποιούν τις ακτινικές συναρτήσεις βάσεις αποτελούν εξέλιξη των MLP γιατί αναπτύσσονται πάνω στα πρότυπα και στην συλλογιστική αυτών.

Τα RBF έχουν τις ρίζες τους στη μαθηματική θεωρία προσέγγισης συναρτήσεων και στην ουσία υλοποιούν μία συνάρτηση παρεμβολής (interpolation function) η οποία προσεγγίζει την τιμή μιας συνάρτησης σε κάποιο σημείο ως το μέσο όρο των τιμών της συνάρτησης σε κοντινά σημεία.

Η δομή ενός RBF φαίνεται στο σχήμα 10.



Σχήμα 10

Το πρόβλημα της μη γραμμικά διαχωριζόμενης ταξινόμησης τα (MLP) το αντιμετωπίζουν με την επίδραση ενός μετασχηματισμού πάνω στις μη – γραμμικά διαχωριζόμενες εισόδους ώστε να μετατραπούν σε γραμμικά διαχωριζόμενες. Αυτό το μετασχηματισμό τον ασκεί το κρυμμένο επίπεδο το

οποίο τροφοδοτεί το επίπεδο εξόδου με ένα νέο γραμμικά χωριζόμενο πρόβλημα που είναι εύκολο να λυθεί.

Το RBF του σχημ. 10 αποτελείται από τρία επίπεδα:

i. Το επίπεδο εισόδου όπως και στα MLP όπου δεν γίνεται καμία επεξεργασία στα δεδομένα εισόδου.
ii. Το κρυμμένο επίπεδο που αποτελείται από μη γραμμικές μονάδες επεξεργασίας και το οποίο μετασχηματίζει το διανυσματικό χώρο σε γραμμικά χωριζόμενο ώστε
iii. Το τρίτο επίπεδο (επίπεδο εξόδου) να επιλύσει ένα εύκολο γραμμικά χωριζόμενο πρόβλημα. Όπως φαίνεται στο σχήμα το επίπεδο εξόδου αποτελείται από γραμμικά στοιχεία που απλώς αθροίζουν τις εισόδους που λαμβάνουν.

Οι συναρτήσεις R_1, R_2...R_n δεν είναι μονότονες συναρτήσεις όπως οι σιγμοειδείς αλλά μη μονότονες, μη γραμμικές που εκτείνονται στο διάστημα [0,1].

Οι μονάδες του κρυμμένου επιπέδου (j=1,...,n) υπολογίζουν μία ειδική συνάρτηση $h_j\left(\vec{x}\right)=R_j\left(\vec{x}\right)$ του διανύσματος εισόδου. Αν $\left(\vec{x}\right)=(x_1,...,x_d)^T$ είναι το διάνυσμα εισόδου η έξοδος του RBF δίνεται από την εξίσωση:

$$y_j\left(\vec{x}\right)=\sum_{j=1}^{M} h_j\left(\vec{x}\right)W_{ji}+W_{oi}$$

i=1,...K W_{ji} είναι το βάρος της σύνδεσης από την κρυμμένη μονάδα j στην μονάδα εξόδου i και W_{oi} είναι η πόλωση της μονάδας i.

Η μορφή των $h_j\left(\vec{x}\right)=R_j\left(\vec{x}\right)$ που ονομάζονται ακτινικές συναρτήσεις βάσης είναι: $h_j\left(\vec{x}\right)=\exp\left(-\frac{\left|1\vec{x}-\vec{\mu}_j\right|^2}{2\delta_j^2}\right)$ (4.1.1.)

όπου $\vec{\mu}_j=(\mu_{j1},...,\mu_{j,d})^T$ διάνυσμα που καθορίζει το κέντρο της συνάρτησης βάσης και η παράμετρος σ_j καθορίζει την ακτίνα της συνάρτησης βάσης. Η

συνάρτηση βάσης παίρνει την μέγιστη τιμή της στο κέντρο $\left(\vec{x} = \vec{\mu_j}\right)$ και η τιμή ελαττώνεται εκθετικά καθώς απομακρυνόμαστε από το κέντρο.

4.2. Εκπαίδευση RBF

Η ανανέωση των βαρών στα δίκτυα RBF πραγματοποιείται με βάση τον παρακάτω αλγόριθμο:

1. Τα πρότυπα μάθησης διαμερίζονται σε κ υποσύνολα χρησιμοποιώντας οποιοδήποτε αλγόριθμο ομαδοποίησης. Στη συνέχεια τα κέντρα αυτά των επιμέρους υποσυνόλων ορίζονται ως τα κέντρα των κ κρυμμένων μονάδων.
2. Τα πλάτη και οι διασπορές των μονάδων αυτών καθορίζονται μέσω κάποιας ευριστικής μεθόδου. Αυτή η ευριστική μέθοδος είναι συνήθως η ευριστική των p-πλησιέστερων γειτόνων και χαρακτηριστικό της είναι ότι μεταβάλλει τα εύρη των μη-γραμμικών συναρτήσεων που χρησιμοποιούνται ώστε να επιτευχθεί μία ορισμένη επικάλυψη αποκρίσεων ανάμεσα σε κάθε κρυμμένη μονάδα και τη γειτονική της έτσι ώστε να εξασφαλίζουν την συνολική απόκριση, συνέχεια και ομαλότητα όπως βέβαια και αξιόπιστη παρεμβολή στα σημεία ανάμεσα στις γειτονικές υποπεριοχές του χώρου των εισόδων. Μια τέτοια κατάλληλη διασπορά μπορεί να οριστεί με την παρακάτω σχέση:

$$\sigma_j = \left(\left(\frac{1}{p}\right)\sum_{i=1}^{P}\left\|C_j - C_i\right\|^2\right)^{\frac{1}{2}}$$

όπου C_i είναι ο πλησιέστερος γείτονας του C_j. Συνήθως χρησιμοποιείται p=2

3. Βρίσκουμε τις τιμές ενεργοποίησης από την (4.1.1.)
4. Υπολογίζονται τα βάρη του στρώματος εξόδου χρησιμοποιώντας γραμμική παρεμβολή ελαχίστων τετραγώνων. Ο στόχος στην προκειμένη περίπτωση είναι η ελαχιστοποίηση της τετραγωνικής νόρμας των υπολοίπων:

$$\min_w E = \frac{1}{2}\sum_{p=1}^{N}\left\|y_p - a_p W\right\|^2 = \frac{1}{2}(T - AW)'(T - AW)$$

όπου W είναι ο πίνακας των βαρών, T είναι ο πίνακας των επιθυμητών εξόδων και A είναι πίνακας των τιμών ενεργοποίησης.

Τα βέλτιστα βάρη ορίζονται ως εκείνα που ελαχιστοποιούν το σφάλμα και το ελάχιστο σφάλμα είναι εκείνο του οποίου η κλίση είναι μηδενική.

$$\forall E = A'(T - AW) = -AT + A'AW = 0$$

Λύνοντας την παραπάνω εξίσωση παίρνουμε

$$\underline{W} = (A'A)^{-1}A'T = A^{+}T$$

οπότε το πρόβλημα ανάγεται στον καθορισμό του αντίστροφου πίνακα του A.

Η παραπάνω τεχνική εκπαίδευσης καλείται εκπαίδευση δύο σταδίων.

Ενιαία Εκπαίδευση

Εναλλακτικά είναι δυνατή η ταυτόχρονη ενημέρωση όλων των παραμέτρων του δικτύου χρησιμοποιώντας την τεχνική gradient descent ακριβώς με τον ίδιο τρόπο που ισχύει στο MLP και η οποία παρουσιάζεται στο παράρτημα. Το μόνο που διαφέρει είναι οι σχέσεις που δίνουν τις μερικές παραγώγους των εξόδων ως προς τις παραμέτρους του δικτύου και οι οποίες βρίσκονται εύκολα ότι είναι:

$$\frac{dy_i}{dW_{oi}} = 1$$

$$\frac{dy_i}{u_{ji}} = h_j\left(\vec{x}\right)$$

$$\frac{dy_i}{d\mu_{jk}} = 2u_{ji}h_j\left(\vec{x}\right)\frac{\left(x_k - \mu_{jk}\right)}{\delta_j^2}$$

$$\frac{dy_i}{d\sigma_j} = u_{ji}h_j\left(\vec{x}\right)\frac{\left|\vec{x} - mu_j\right|^2}{\sigma_j^3}$$

Φυσικά μπορεί να χρησιμοποιηθεί είτε ομαδική (off line) είτε σειριακή (on-line) ενημέρωση σε πλήρη αναλογία με το δίκτυο MLP.

4.3. Σύγκριση MLP και RBF

Η μεγάλη διαφορά μεταξύ MLP και RBF είναι ότι τα NN RBF είναι πιο εξειδικευμένα να κάνουν μία συγκεκριμένη δουλειά απ' ότι τα MLP που χρησιμοποιούνται για πιο γενικές χρήσεις.

Αυτό οφείλεται στο γεγονός ότι τα RBF αξιοποιούν στο μέγιστο δυνατό την πρωθύστερη (προϋπάρχουσα) γνώση γύρω απ' το πρόβλημα που θέλουμε να λύσουμε. Έτσι τα RBF είναι πιο αποτελεσματικά σαν ταξινομητές.

Απ' την άλλη μεριά όμως τα RBF στερούνται γενικότητος. Δηλαδή ένα NN που προκύπτει απ' τον $1^{ο}$ αλγόριθμο που αναπτύξαμε δεν είναι κατάλληλο παρά μόνο για τα δεδομένα για τα οποία σχεδιάστηκε. Επίσης άλλο ένα μειονέκτημα των RBF είναι ότι η εκ των προτέρων γνώση δεν είναι πάντα δυνατή.

Μια άλλη διαφορά μεταξύ των δύο τύπων δικτύου είναι ότι τα MLP μαθαίνουν χρησιμοποιώντας κατανεμημένη αναπαράσταση, δηλαδή η γνώση σχετικά με την έξοδο που αντιστοιχεί σε κάποια είσοδο κατανέμεται στις τιμές των βαρών όλων των κρυμμένων μονάδων. Έτσι τα MLP έχουν ανοχή σε σφάλματα (fault tolerence) που σημαίνει ότι η καταστροφή κάποιων συνδέσεων δεν είναι καταστροφική για το δίκτυο λόγω του ότι η γνώση είναι κατανεμημένη σε όλες τις συνδέσεις.

Αντιθέτως στα RBF ο χώρος προτύπων διαιρείται σε περιοχές που κάθε μία απ' αυτές αντιστοιχεί σε μία μονάδα του κρυμμένου επιπέδου. Στα MLP η κάθε περιοχή ορίζεται απ' την τομή των υπερεπιπέδων που ορίζονται από όλες τις κρυμμένες μονάδες.

ΚΕΦΑΛΑΙΟ 5

Άλλοι αλγόριθμοι μάθησης

Εισαγωγή

Τα μέχρι τώρα μοντέλα ΝΝ που παρουσιάσαμε χρησιμοποιούν για την εκπαίδευσή τους αλγόριθμους που ανήκουν στην κατηγορία της επιβλεπόμενης μάθησης. Ωστόσο υπάρχουν αλγόριθμοι που ανήκουν στην κατηγορία της ανταγωνιστικής μάθησης (μάθηση χωρίς επίβλεψη) και χρησιμοποιούνται από άλλους τύπους ΝΝ.

Μάθηση χωρίς επίβλεψη

Σε αντίθεση με ότι συμβαίνει στην μάθηση με επίβλεψη όπου τα δεδομένα εισόδου απεικονίζονται στα δεδομένα εξόδου στη μάθηση χωρίς επίβλεψη έχουμε στην διάθεσή μας μόνο το σύνολο δεδομένων εισόδου $\vec{x}^{i}$ και δεν υπάρχει κάποιο επιθυμητό διάνυσμα εξόδου $\vec{y}^{i}$ όπως συμβαίνει στους μέχρι τώρα αναλυθέντες αλγορίθμους.

Σκοπός της μάθησης χωρίς επίβλεψη είναι η αυτοοργάνωση και η ανακάλυψη διάφορων χαρακτηριστικών ιδιοτήτων των δεδομένων $\vec{x}^{i}$.

Τα προβλήματα που εμπίπτουν στην κατηγορία της μάθησης χωρίς επίβλεψη είναι:

i. Ομαδοποίηση (clustering)
ii. Ανάλυση βασικών συνιστωσών (Principal component analysis)
iii. Μείωση της διάστασης των δεδομένων (προβολή τους σε χώρο μικρότερης διάστασης από τον αρχικό).

Για κάθε μία από τις παραπάνω κατηγορίες προβλημάτων έχουν αναπτυχθεί μοντέλα νευρωνικών δικτύων και αντίστοιχες τεχνικές εκπαίδευσης. Εδώ θα παρουσιάσουμε αλγόριθμους ομαδοποίησης καθώς και μία άλλη κατηγορία μάθησης την ανταγωνιστική (competitive learning) η οποία ανήκει στην γενική κατηγορία μάθησης χωρίς επίβλεψη.

5.3. Αλγόριθμοι ομαδοποίησης (Cluster algorithms)

Όπως αναφέραμε παραπάνω οι αλγόριθμοι αυτοί ανήκουν στην κατηγορία της μάθησης χωρίς επίβλεψη. Οι αλγόριθμοι ομαδοποίησης ταξινομούν πρότυπα σε ομάδες σύμφωνα με κάποιες μετρούμενες ομοιότητες. Με την κατάλληλη ομαδοποίηση που πετυχαίνουμε με αυτούς τους αλγόριθμους το ποσοστό ομοιότητας μεταξύ των σχεδίων (προτύπων) pattern: πρότυπο, σχέδιο, μοντέλο που ανήκουν στις ίδιες ομάδες μεγιστοποιείται ενώ το ποσοστό ομοιότητας μεταξύ μοντέλων (σχεδίων) διαφορετικών ομάδων ελαχιστοποιείται. Με άλλα λόγια, οι αλγόριθμοι ομαδοποίησης προσπαθούν να μεγιστοποιήσουν τις αποστάσεις των μοντέλων που ανήκουν στην ίδια ομάδα και να ελαχιστοποιήσουν τις μεταξύ των ομάδων αποστάσεις. Πώς μετράμε όμως την ομοιότητα δύο μοντέλων (patterns); (προτύπων).

Για να ορίσουμε έναν ομαδοποιητή δεδομένων, είναι απαραίτητο να ορίσουμε πρώτα μία μέτρηση της ομοιότητας ώστε να μπορούμε να κατατάξουμε τα δεδομένα σε κάποιες ειδικές ομάδες. Έστω x είναι ένα χαρακτηριστικό διάνυσμα των μοντέλων και z ένα διάνυσμα αναφοράς. Η Ευκλείδεια απόσταση μεταξύ x και z είναι:

$$d=||x-z|| \quad \text{ή} \quad d=\sum_j W_i(x_j - Z_j)^2 \tag{5.1}$$

όπου W_j είναι το βάρος της j-οστής συντεταγμένης του διαστήματος του μοντέλου. Όσο πιο μικρή είναι η απόσταση αυτή τόσο μεγαλύτερη είναι η ομοιότητα. Η εξίσωση 5.1 δεν είναι αποτελεσματική καθώς μετράει την απόσταση μεταξύ ενός διανύσματος μοντέλου x και μιας ομάδας (cluster) η οποία αναπαριστάται από το διάνυσμα αναφοράς z. Αυτό συμβαίνει γιατί το διάνυσμα μοντέλου είναι ένα σημείο χωρίς μέγεθος ενώ μία ομάδα έχει ένα μέγεθος το οποίο περιγράφεται από κάποιες διαχωρισμένες παραμέτρους. Η απόσταση από μία ομάδα μπορεί να οριστεί από την απόσταση είτε απ' το κέντρο της είτε απ' τις άκρες της αλλά και οι δύο ορισμοί δεν είναι σημαντικοί (ουσιαστικοί) για την ομαδοποίηση. Η Μαχαλανόμπις απόσταση είναι μια καλή μέτρηση της απόστασης μεταξύ ενός διανύσματος και μιας ομάδας. Η Μαχαλανόμπις απόσταση δίνεται απ' την εξίσωση:

$$d_n=(x-m)^1 C^{-1} (x-m) \tag{5.2}$$

Όπου θεωρούμε ότι μία ομάδα μπορεί προσεγγιστικά να περιγραφεί από μία διανεμημένη πυκνότητα με μέσο διάνυσμα m και μία μήτρα C.

Άλλες μετρήσεις (όχι της απόστασης) είναι επίσης χρήσιμες για κάποιες εφαρμογές. Για παράδειγμα η κανονικοποιημένη συσχέτιση είναι μια συνάρτηση ομοιότητας.

$$S(x, z) = \frac{x' z}{\|x\|\|z\|}$$

η οποία είναι το εσωτερικό γινόμενο (inner product) μεταξύ των διανυσμάτων x και z και παίρνει την τιμή 1 όταν τα x και z είναι παράλληλα.

5.3.1. Ένας απλός αλγόριθμος ομαδοποίησης

Ο αλγόριθμος ομαδοποίησης που θα αναπτύξουμε παρακάτω δεν απαιτεί πρωθύστερη (a priori) γνώση των ομάδων. Έστω ότι έχουμε ένα σύνολο από N απλά μοντέλα και T είναι ένα θετικό κατώφλι. Ακολουθούμε τα εξής βήματα:

1. Πάρε ένα σημείο σαν ένα κέντρο της πρώτης ομάδας (τάξης).
2. Για κάθε μη ταξινομημένο σημείο, υπολόγισε την απόσταση μεταξύ του σημείου και όλων των κέντρων των κλάσεων (ομάδα, τάξη, class). Αν όλες οι αποστάσεις είναι μεγαλύτερες απ' το κατώφλι T, πάρε το σημείο σαν το κέντρο μιας νέας κλάσης. Διαφορετικά το σημείο θα ανήκει στην τάξη της οποίας το κέντρο είναι κοντύτερα στο σημείο.
3. Ο αλγόριθμος τερματίζεται όταν όλα τα σημεία είναι ταξινομημένα.

Αυτός ο αλγόριθμος έχει περιορισμούς όπως το αποτέλεσμα εξαρτάται απ' το κατώφλι (threshold).

5.3.2. Μέγιστης απόστασης αλγόριθμος

Ο αλγόριθμος αυτός βασίζεται στην Ευκλείδεια απόσταση. Δεν σχηματοποιούμε νέες ομάδες κατά την διάρκεια της ταξινόμησης, όπως γίνεται με τον προηγούμενο αλγόριθμο, αλλά ο αλγόριθμος αποτελείται από δύο φάσεις. Η πρώτη φάση ορίζει όλες τις πιθανές ομάδες και η δεύτερη φάση υλοποιεί την ταξινόμηση. Τα βήματα που ακολουθούμε είναι τα εξής:

1. Πάρε το πρώτο σημείο σαν το κέντρο της πρώτης ομάδας, και δήλωσέ το με z_1.

2. Βρες το δείγμα που είναι μακρύτερα από το z_1 και θεώρησέ το σαν το κέντρο της δεύτερης ομάδας z_2. Πάρε την απόσταση μεταξύ των z_1 και z_2 σαν μία σταθερή απόσταση D_s.
3. Υπολόγισε τις αποστάσεις για κάθε εναπομείναν σημείο από τα κέντρα των ομάδων z_1,. z_2,..., και βρες σημεία με την ελάχιστη απόσταση απ' τα κέντρα αυτών των ομάδων. Στη συνέχεια βρες τη μέγιστη απ' τις ελάχιστες αποστάσεις για όλα αυτά τα σημεία. Αν η αναλογία αυτών των μεγίστων ξεπερνούν την σταθερή απόσταση D_s (είναι μεγαλύτερες από ένα κατώφλι) το αντίστοιχο σημείο παίρνεται σαν ένα άλλο κέντρο ομάδα. Αυτή η διαδικασία τελειώνει όταν δεν μπορούν να βρεθούν νέα κέντρα ομάδων.
4. Ανάθεσε κάθε δείγμα σημείου στην κοντινότερη ομάδα όπως αναπαριστάται απ' τα κέντρα τους.

5.3.3. Ο αλγόριθμος των Κ-μέσων

Στους παραπάνω αναλυθέντες αλγόριθμους τα κέντρα των ομάδων ορίζονται ακτινικά (π.χ. το κατώφλι επιλέγεται ακτινικά). Θεωρώντας την πραγματικότητα ότι η ακτινική επιλογή των κέντρων των ομάδων δεν είναι συνήθως η σωστή, οι ομάδες που σχηματίζονται συνήθως δεν αντικατοπτρίζουν την φύση των δεδομένων. Παρακάτω δίνουμε τα βήματα του k-mean (κ-μέσων) αλγόριθμου σε γενικές γραμμές καθώς και μια πιο λεπτομερή ανάλυσή του. Ο κ-μέσων αλγόριθμος εκσυγχρονίζει δυναμικά τα κέντρα των ομάδων για κάθε νέο δείγμα μέχρι την στιγμή που τα κέντρα των ομάδων φτάσουν σε μία σταθερή κατάσταση. Αποτελεί τον πιο απλό αλγόριθμο σημειακής ομαδοποίησης και είναι κατάλληλος για την εύρεση συμπαγών ομάδων. Χρησιμοποιεί ένα σημείο $\vec{w}_j = (w_{k1}..., w_{jd})^T$ ως αντιπρόσωπο για κάθε ομάδα j. Προϋποθέτει ότι ο αριθμός των ομάδων M είναι σταθερός και καθορίζεται εξαρχής.

Τα βήματα που ακολουθούμε για την υλοποίηση των Κ-μέσων αλγορίθμου είναι:

4. Επίλεξε Κ αρχικά κέντρα z_1, z_2, ..., z_k.
5. Διαχώρισε τα δείγματα μεταξύ των κ ομάδων χρησιμοποιώντας το κριτήριο ελάχιστης απόστασης.

6. Πάρε το κεντροειδές από κάθε ομάδα σαν ένα καινούργιο κέντρο αυτής της ομάδας.
7. Σύγκρινε τα καινούργια κέντρα με τα προηγούμενα. Αν η αλλαγή είναι κάτω από ένα κατώφλι (T) ολοκλήρωσε (τελείωσε) τον αλγόριθμο. Αν όχι πήγαινε πίσω στο βήμα 2.
8. Παρακάτω δίνουμε μία πιο αναλυτική περιγραφή του αλγορίθμου (πιο μαθηματική). Με Ω_j συμβολίζουμε το σύνολο των προτύπων της ομάδας (j(j=1, ..., M). Έστω $\vec{x}^{i}$ (i=1,...N) τα πρότυπα εκπαίδευσης.

- Αρχικοποίηση των M κέντρων $\vec{\omega}^{(\theta\theta}_{i}$ (j=1,...,M)
- t=0
- Repeat

1. Για κάθε πρότυπο εκπαίδευσης $\vec{x}^{i}$ υπολογισμός των αποστάσεων $d(\vec{x}^{i}), \vec{w}_j$ και ταξινόμηση του $\vec{x}^{i}$ στην ομάδα $\Omega_K^{(\pi\pi}$ με τη μικρότερη απόσταση: $d(\vec{x}^{i}), \vec{w}_K = \min_j d(\vec{x}^{i}, \vec{w}_j)$.
2. Για κάθε ομάδα j υπολογισμός των νέων κέντρων:

$$\vec{W}_j^{(t+1)} = \frac{1}{N_j^{(t)}} \sum_{\vec{x} \in \Omega_j^{(t)}} \vec{x}^{(i)}$$

   όπου N_j είναι ο αριθμός των προτύπων που έχουν αποδοθεί στην ομάδα j.
3. t: = t+1

- until $\left| \vec{W}_j^{(t)} - \vec{W}_j^{(t-1)} \right| < \in \forall j$ ή υπέρβαση ενός μέγιστου αριθμού επαναλήψεων.

5.4. Ο αλγόριθμος LVQ

Εισαγωγή

Ο αλγόριθμος αυτός ανήκει στην κατηγορία της ανταγωνιστικής μάθησης η οποία υλοποιείται με τη χρήση ενός επιπέδου από ανταγωνιστικούς νευρώνες (ανταγωνιστικό επίπεδο). Κάθε φορά που εμφανίζεται κάποιο διάνυσμα εισόδου στο ανταγωνιστικό επίπεδο έχουμε ανταγωνισμό μεταξύ των νευρώνων του ανταγωνιστικού επιπέδου για το

ποιος θα βγει νικητής για το συγκεκριμένο πρότυπο με βάση κάποιο κριτήριο επίδοσης. Ο νικητής νευρώνας είναι αυτός ο οποίος συνήθως μεταβάλλει τις τιμές των βαρών του περισσότερο σε σχέση με τους υπόλοιπους νευρώνες. Με τον τρόπο αυτό επιτυγχάνεται αυτοοργάνωση (self-organization). Το κριτήριο που χρησιμοποιείται για την εύρεση του νικητή νευρώνα είναι η απόσταση (συνήθως Ευκλείδεια) του διανύσματος εισόδου $\vec{x}^{i}=(x_1,...,x_d)^T$ από το διάνυσμα των βαρών $w_i=(w_{i1}...,w_{id})^T$ κάθε νευρώνα i. Ο νικητής νευρώνας j είναι αυτός με τη μικρότερη απόσταση $|\vec{x} - \vec{w}_j|$. Η αυτοοργάνωση επιτυγχάνεται ση συνέχεια μετακινώντας το διάνυσμα βαρών του νικητή νευρώνα j προς το διάνυσμα εισόδου $\vec{x}^{i}$. Αυτό επαναλαμβάνεται για κάθε νέο διάνυσμα εισόδου. Στην κατηγορία της ανταγωνιστικής μάθησης ανήκει ο αλγόριθμος LVQ (learning vector quantization) που αποτελεί μία προσαρμοζόμενη παραλλαγή του αλγορίθμου κ-μέσων. Σύμφωνα με αυτόν τον αλγόριθμο σε κάθε βήμα οι διάφορες ομάδες ανταγωνίζονται για την απόκτηση του προτύπου εκπαίδευσης που χρησιμοποιούμε στο συγκεκριμένο βήμα. Στη συνέχεια έχουμε μετακίνηση των κέντρων των ομάδων ανάλογα με το ποια ομάδα βγαίνει νικήτρια. Ο αλγόριθμος LVQ μπορεί εύκολα να υλοποιηθεί με ένα νευρωνικό δίκτυο που περιλαμβάνει ένα ανταγωνιστικό επίπεδο με τόσους ανταγωνιστικούς νευρώνες όσες και οι ομάδες. Μπορεί να χρησιμοποιηθεί τόσο για ομαδοποίηση όσο και για ταξινόμηση.

5.4.1. Αλγόριθμος LVQ για ομαδοποίηση

Τα βήματα που ακολουθούμε είναι τα εξής:

- Καθορισμός του αριθμού M των ομάδων
- Αρχικοποίηση των M κέντρων $\vec{w}_j^{(0)}$ (j=1,...,M)
- Αρχικοποίηση του ρυθμού μάθησης n
- t=0

Repeat

1. Τυχαία επιλογή ενός προτύπου $\vec{x}^{i}$

2. Εύρεση του νικητή νευρώνα K, δηλαδή της ομάδας της οποίας το κέντρο έχει τη μικρότερη Ευκλείδεια απόσταση από το πρότυπο $\vec{x}^{i}$.
3. Υπολογισμός του νέου διανύσματος βαρών του νευρώνα K, δηλ. της νέας θέσης $\vec{W}_{kj}^{(t+1)}$ για το κέντρο μόνο της νικήτριας ομάδας K:!

$$W_{kj}^{(t+1)} = W_{kj}^{(t)} + n\left(x_j^i - W_{kj}^{(t)}\right)$$

για j=1,...,d. Τα βάρη των υπολοίπων ομάδων δεν μεταβάλλονται:

$$\vec{W}_{kj}^{(t+1)} = \vec{W}_i^{(t)} \text{ για } i \neq K$$

4. Βαθμιαία αργή μείωση του n
5. t:=t+1

Until τα βάρη W_j συγκλίνουν τις τελικές τους τιμές.

Η μείωση της παραμέτρου n γίνεται αργά παρόλο που ο αλγόριθμος δουλεύει ικανοποιητικά και όταν ο ρυθμός μάθησης παραμένει σταθερός σε μία μικρή τιμή (π.χ. n=0,1). Στο σημείο αυτό θα αναφερθούμε σε ένα πρόβλημα που αφορά τους αλγορίθμους μάθησης των NN. Με δεδομένη την δομή και τον αριθμό των μονάδων του NN καθώς και ενός αλγορίθμου μάθησής του το πρόβλημα που καλούμαστε να λύσουμε έχει πάντα λύση; Δηλαδή η λύση που μας προσφέρει ο αλγόριθμος είναι κατάλληλη για το πρόβλημα; Απάντηση σ' αυτά τα ερωτήματα θα δώσουμε σε άλλα κεφάλαια. Άλλο ένα πρόβλημα που έχουμε να αντιμετωπίσουμε είναι ότι ο αλγόριθμος (π.χ. LVQ) δεν μπορεί να εγγυηθεί την εύρεση του ολικού ελαχίστου και συνήθως καταλήγει σε τοπικά ελάχιστα.

Ο αλγόριθμος LVQ για ταξινόμηση

Με κατάλληλη τροποποίηση του αλγόριθμου που αναπτύχθηκε στην προηγούμενη παράγραφο ο LVQ μπορεί να χρησιμοποιηθεί για ταξινόμηση. Στην αρχή αναθέτουμε κάποια κατηγορία σε καθένα από τους ανταγωνιστικούς νευρώνες. Η αρχικοποίηση γίνεται θέτοντας τα βάρη ενός νευρώνα που τον αντιστοιχίζουμε στην κατηγορία C_i ίσο με κάποιο πρότυπο εκπαίδευσης της κατηγορίας C_i. Η αρχικοποίηση των βαρών των νευρώνων γίνεται και με τυχαίο τρόπο.

Σε κάθε βήμα του αλγόριθμου LVQ για ταξινόμηση θεωρούμε ένα πρότυπο εκπαίδευσης (έστω x^i) και βρίσκουμε τον νικητή νευρώνα j. Τα βάρη των υπολοίπων νευρώνων i δεν μεταβάλλονται. Τα βάρη του νικητή νευρώνα j μεταβάλλονται ως εξής:

- Αν η κατηγορία του προτύπου $\vec{x}^i$ και του νευρώνα j συμπίπτουν, τότε το διάνυσμα βαρών της $\vec{w}_j$ μετακινείται προς το πρότυπο $\vec{x}^i$.

$$\vec{W}_{kj}^{(t+1)} = \vec{W}_i^{(t)} + n\left(\vec{x}^i - \vec{w}_j^{(t)}\right)$$

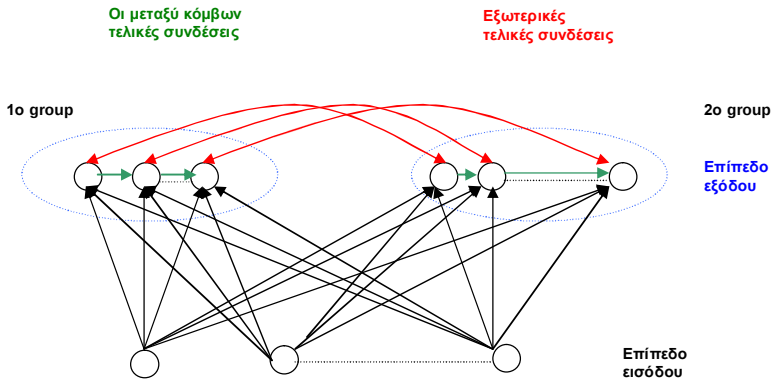
- Αν η κατηγορία του προτύπου $\vec{x}^i$ και της ομάδας j είναι διαφορετικές τότε το κέντρο $\vec{w}_j$ απομακρύνεται από το πρότυπο $\vec{x}^i$

$$\vec{W}_{kj}^{(t+1)} = \vec{W}_i^{(t)} - n\left(\vec{x}^i - \vec{w}_j^{(t)}\right)$$

Ο ρυθμός μάθησης (n) ελαττώνεται σταδιακά κατά τη διάρκεια της εκπαίδευσης και ο αλγόριθμος σταματά όταν τα κέντρα δεν έχουν μετακινηθεί σημαντικά μεταξύ δύο ομάδων προτύπων.

Η ταξινόμηση ενός νέου διανύσματος εισόδου γίνεται βρίσκοντας το νικητή νευρώνα και θεωρώντας ως απόφαση ταξινόμησης την κατηγορία του νευρώνα αυτού. Υπάρχουν πολλές παραλλαγές του αλγορίθμου LVQ για ταξινόμηση (LVQ2, LVQ3 κ.λ.π.) και οι οποίες προσπαθούν να βελτιώσουν μερικά απ' τα μειονεκτήματά του.

Για να γίνουν πιο κατανοητά τα όσα αναφέραμε ως εδώ για την ανταγωνιστική μάθηση δίνουμε στο σχήμα 11 ένα ανταγωνιστικό δίκτυο.



Σχήμα 11

Στο παραπάνω ανταγωνιστικό δίκτυο υπάρχουν μία ή περισσότερες ομάδες στις μονάδες εξόδου. Οι μεταξύ κόμβων συνδέσεις είναι μεταξύ μονάδων μέσα στην ίδια ομάδα (πράσινο χρώμα) ενώ οι εξωτερικές συνδέσεις (excitatory) είναι μεταξύ αντίστοιχων μονάδων διαφορετικών γκρουπ (κόκκινο χρώμα). Μεταξύ των επιπέδων υπάρχουν μόνο προσοτροφοδοτούμενες συνδέσεις (μαύρο χρώμα).

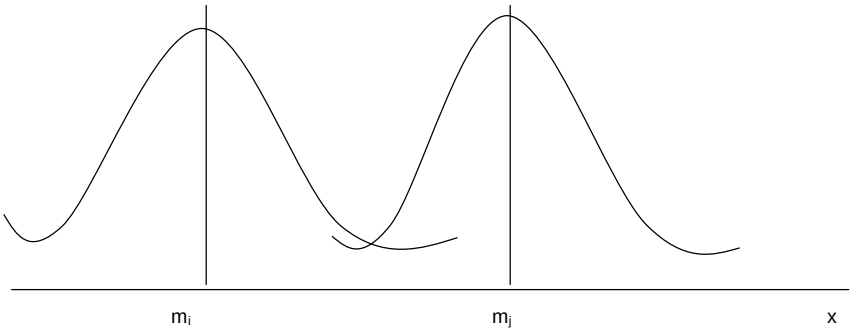
5.4.3. Ο αλγόριθμος LVQ2

Ένας τύπος NW που βασίζεται στην ανταγωνιστική μάθηση και που θα τη παρουσιάσουμε στο επόμενο κεφάλαιο είναι το δίκτυο SOM(self-organizing map→ αυτοοργανούμενος χάρτης).

Ο αλγόριθμος μάθησης ενός SOM είναι ο LVQ. Εδώ θα παρουσιάσουμε τον LVQ2.

Όταν ένα SOM χρησιμοποιείται για ταξινόμηση, οι μονάδες εξόδου ομαδοποιούνται σε υποσύνολα καθένα αντίστοιχο σε μία ομάδα. Θεωρήστε ότι δύο templates m_i και m_j που ανήκουν σε διαφορετικές κλάσεις είναι το πρώτο και το δεύτερο κοντινότερο template στο διάνυσμα εισόδου x. Αλλά, από συγκεκριμένες a priori πληροφορίες ξέρουμε ότι το x δεν είναι στην ομάδα i αλλά στην j, έτσι ώστε mi_i και m_j είναι σε λάθος θέση. Επίσης θεωρούμε ότι η διακριτική επιφάνεια ορίζεται σαν το μέσο των m_i και m_j. Τώρα θα ορίσουμε ένα συμμετρικό παράθυρο γύρω απ' το μέσο του σχεδίου και θα

βάλουμε τον όρο ότι οι διορθώσεις στα m_i και m_j θα γίνονται αν το x πέφτει μέσα στο παράθυρο στην λάθος πλευρά του σχεδίου (σχήμα 12).



Σχήμα 12

Γι' αυτό, αν x ανήκει στην ομάδα $w_i \neq w_j$ και πέφτει μέσα στο παράθυρο, αναβάθμισε τα m_i και m_j ως ακολούθως:

$$m_i(t+1) = m_i(t)-\alpha(t)[x(t)- m_i(t)]$$

$$m_j(t+1) = mj(t)+\alpha(t)[x(t)- m_j(t)]$$

Σε όλες τις άλλες περιπτώσεις τα templates αφήνονται άθικτα. Το διάνυσμα εισόδου x καθορίζεται να είναι μέσα στο παράθυρο αν $\min\left(\frac{d_i}{d_j},\frac{d_j}{d_i}\right) > S$

Όπου d_i και d_j είναι οι αποστάσεις απ' το πρότυπο εισόδου x των m_i και m_j αντίστοιχα. Σημειώστε ότι το στενότερο σχετικό πλάτος του παραθύρου από το w είναι όταν έχουμε $S = \frac{1-w}{1+w}$

Η βέλτιστη επιλογή του w εξαρτάται από τον αριθμό των διαθέσιμων δειγμάτων. Αν έχουμε ένα μεγάλο αριθμό από δείγματα, ένα στενό παράθυρο θα εγγυηθεί την πιο ακριβή τοποθεσία της οριακής απόφασης. Απ' την άλλη μεριά για καλή στατιστική ακρίβεια, ο αριθμός των δειγμάτων που πέφτουν μέσα στο παράθυρο πρέπει να είναι επαρκής.

Κεφάλαιο 6
Το Νευρωνικό Δίκτυο SOM

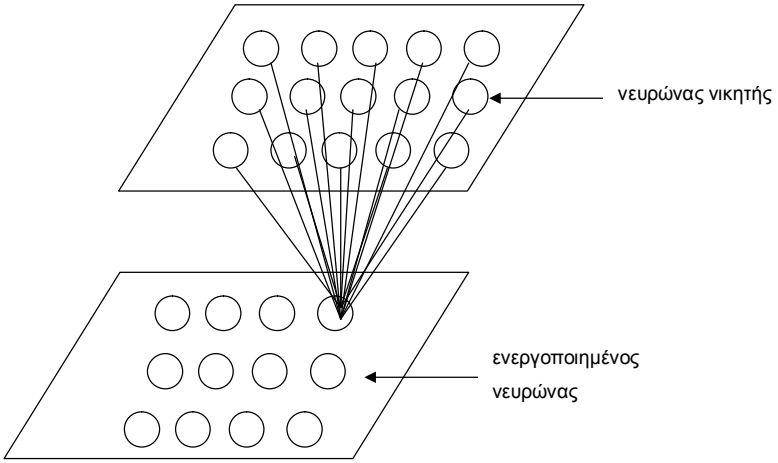
Εισαγωγή

Αυτός ο τύπος ΝΝ προτάθηκε απ' τον Kohonen στα μέσα της δεκαετίας του 1980 και αποτελεί ένα αυτοοργανούμενο χάρτη χαρακτηριστικών που βασίζεται στην έννοια της ανταγωνιστικής μάθησης. Οι νευρώνες του ανταγωνιστικού επιπέδου δηλαδή μάχονται μεταξύ τους για το ποιος θα επικρατήσει (θα βγει νικητής) για κάποιο συγκεκριμένο ερέθισμα (πρότυπο εισόδου).

Σε ένα δίκτυο SOM (self organizing feature map) οι νευρώνες τοποθετούνται στη θέση των κόμβων ενός πλέγματος μιας ή δύο διαστάσεων. Χάρτες μεγαλύτερης διάστασης είναι δυνατό να δημιουργηθούν, αλλά δε συνηθίζεται. Οι νευρώνες συντονίζονται επιλεκτικά σε διάφορα πρότυπα εισόδου στα πλαίσια της ανταγωνιστικής διαδικασίας εκπαίδευσης. Οι νευρώνες διατάσσονται με τέτοιο τρόπο ώστε η θέση τους στο πλέγμα να συσχετίζεται με συγκεκριμένα χαρακτηριστικά των προτύπων εισόδου. Έτσι προήλθε και το όνομα Αυτοοργανούμενος Χάρτης: οι τοπογραφικές θέσεις των νευρώνων είναι ενδεικτικές των χαρακτηριστικών των προτύπων εισόδου.

Η ανάπτυξη του ΝΝ SOM έγινε με βάση ένα χαρακτηριστικό του ανθρώπινου εγκεφάλου: η οργάνωσή του σε πολλές περιοχές έτσι ώστε διαφορετικά αισθητήρια να διεγείρουν διαφορετικές περιοχές του εγκεφάλου. Πιο συγκεκριμένα, αισθήσεις όπως της αφής, της όρασης και της ακοής απεικονίζονται σε διαφορετικές περιοχές του εγκεφαλικού φλοιού με τοπολογική διάταξη. Επομένως, ο υπολογιστικός χάρτης αποτελεί ένα βασικό δομικό στοιχείο επεξεργασίας πληροφοριών του νευρωνικού συστήματος. Υπολογιστικός χάρτης είναι ένας πίνακας νευρώνων που αναπαριστούν επεξεργαστές ή φίλτρα που είναι μεταξύ τους ελαφρώς διαφορετικά συντονισμένα. Με τον τρόπο αυτό οι νευρώνες αντιστοιχίζουν τα σήματα εισόδου σε μία κατανομή της ενεργοποίησης στα διάφορα σημεία του χάρτη, ανάλογα με τα χαρακτηριστικά των ερεθισμάτων εισόδου. Σε κάθε επίπεδο π.χ. του ακουστικού μονοπατιού, τα νευρωνικά κύτταρα και οι ίνες τοποθετούνται ανατομικά σε σχέση με την συχνότητα που δημιουργεί την

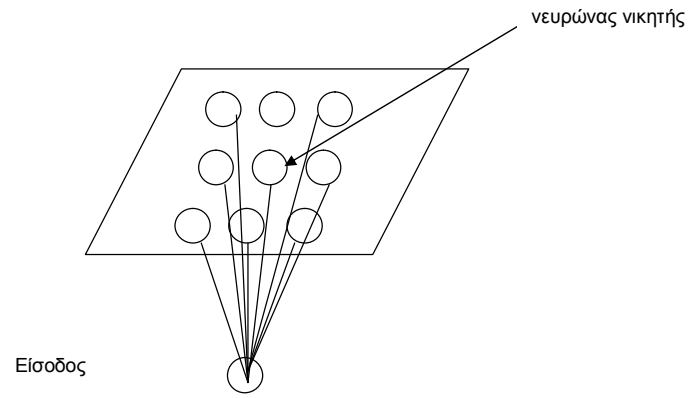
ισχυρότερη απόκριση σε κάθε νεύρο. Παρατηρείται έτσι μία «τονοτοπική» οργάνωση. Αν και οι περισσότερες από τις χαμηλού επιπέδου οργανώσεις του εγκεφάλου είναι γενετικά προκαθορισμένες, κάποιες υψηλότερου επιπέδου οργανώσεις δημιουργούνται με τη βοήθεια της εκμάθησης από διάφορους αλγορίθμους παρατηρείται δηλαδή αυτοοργάνωση. Ο Kohonen, με βάση τα χαρακτηριστικά αυτά, παρουσίασε έναν αλγόριθμο που παράγει αυτό που εκείνος ονομάζει: «αυτοοργανούμενες απεικονίσεις χαρακτηριστικών». Υπάρχουν δύο βασικά μοντέλα απεικόνισης χαρακτηριστικών: i) το μοντέλο του Σχήματος 12 που προτάθηκε απ' τους Willshaw και von der Malsburg (1976) και ii) το μοντέλο του Σχήματος 13 που προτάθηκε απ' τον Kohonen (1982) και δεν προσπαθεί να εξηγήσει νευροβιολογικές λεπτομέρειες. Το 1^{ο} μοντέλο των Willshaw και Malsburg βασιζόταν σε βιολογικές αρχές και προσπαθούσε να εξηγήσει τη σύνδεση του αμφιβληστροειδή με το οπτικό τμήμα του φλοιού του εγκεφάλου. Συγκεκριμένα υπάρχουν δύο ξεχωριστά δισδιάστατα πλέγματα νευρώνων που συνδέονται μεταξύ τους και το ένα προβάλλεται πάνω στο άλλο. Το ένα πλέγμα αναπαριστά νευρώνες εισόδου και το άλλο εξόδου. Το πλέγμα εισόδου χρησιμοποιεί ένα διεγερτικό μηχανισμό μικρής εμβέλειας και έναν μηχανισμό αναστολής μεγάλης εμβέλειας. Αυτοί οι μηχανισμοί είναι τοπικής φύσης και ιδιαίτερα σημαντικοί για την αυτοοργάνωση. Η βασική ιδέα του μοντέλου είναι η κωδικοποίηση της γεωμετρικής εγγύτητας σε μορφή συσχετίσεων ηλεκτρικής δραστηριότητας και η επίτευξη αυτοοργάνωσης ώστε γειτονικοί νευρώνες εισόδου να απεικονίζονται σε γειτονικούς νευρώνες εξόδου. Η αυτοοργάνωση λοιπόν παράγει μια τοπολογικά διατεταγμένη απεικόνιση. Στο σχήμα 13 φαίνεται το μοντέλο των Willshaw και Malsburg.



Σχήμα 13: Μοντέλο Willshaw-von der Malsburg

Το δεύτερο μοντέλο (του Kohonen) ενσωματώνει τα βασικά χαρακτηριστικά των υπολογιστικών χαρτών του εγκεφάλου χωρίς να προσπαθεί να εξηγήσει νευροβιολογικές λεπτομέρειες. Είναι περισσότερο γενικευμένο από το προηγούμενο, με την έννοια ότι είναι ικανό να διενεργεί συμπίεση δεδομένων (μείωση των διαστάσεων εισόδου). Ανήκει στην κατηγορία των αλγορίθμων διανυσματικού κβαντισμού (vector quantization). Μπορεί να προσεγγιστεί με δύο τρόπους: είτε με βάση τις αρχές της βιολογικής αυτοοργάνωσης είτε ακολουθώντας τη θεωρία πληροφορίας (κωδικοποίηση πληροφορίας).

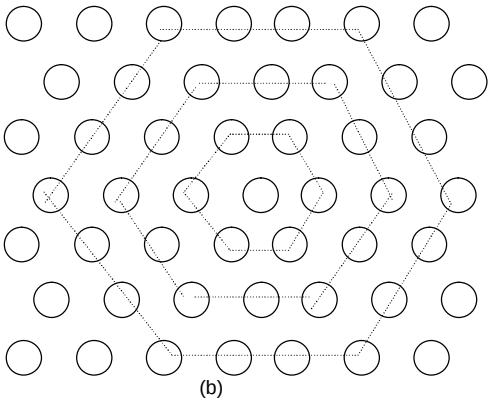
Τα NN SOM έχουν ποικιλία εφαρμογών όπως στην αναγνώριση φωνής, στην ρομποτική, στην παραγωγή ελέγχου, στις τηλεπικοινωνίες, στην επεξεργασία εικόνας (image processing) και στην όραση υπολογιστών που θα παρουσιαστούν στο 2^{ο} τμήμα της διπλωματικής.



Σχήμα 14. Μοντέλο Kohonen

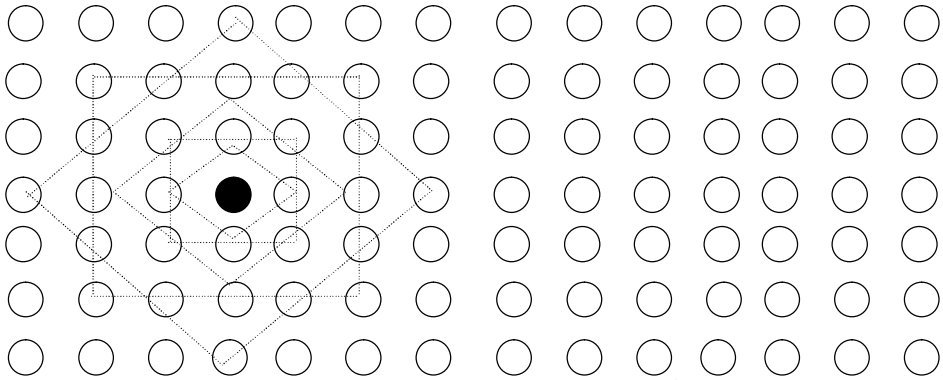
6.1. Το Δίκτυο SOM

Ένα τυπικό δίκτυο SOM αποτελείται από ένα επίπεδο μονάδων εισόδου και ένα επίπεδο από μονάδες εξόδου, συνήθως τακτοποιημένες σαν μία σειρά δύο διαστάσεων όπως φαίνεται στο σχήμα 14. Υποθέτουμε ότι υπάρχουν M μονάδες εισόδου και N μονάδες εξόδου στο δίκτυο.



Σχήμα 15(b)

Κάθε μονάδα εισόδου συνδέεται με κάθε μονάδα εξόδου με μια συγκεκριμένη σύναψη βαρών $\{w_{mn,}\ m=1,2,...,M,\quad n=1,2,...,N\}$.



Σχήμα 15: Οι μονάδες εξόδου του Kohonen SOM NN είναι τακτοποιημένες σαν 2-διάστατη σειρά. Οι γείτονες που ορίζονται απ' την απόσταση διασύνδεσης είναι ένας κύκλος (α) ορθογώνια, σειρά και πιθανοί γείτονες (β) εξαγωνική σειρά και πιθανοί γείτονες.

Για κάθε μονάδα εξόδου ορίζεται ένα διάνυσμα (template) βαρών $m_i(t)=(W_{1j}(t), ...,W_{mi}(t))^T$ (i=1,2,...N).

Αυτές οι templates είναι σαν τα κωδικοποιημένα διανύσματα του διανυσματικού κβαντιστή (Vector Quantization). Αυτά θα ταιριαστούν με τα διανύσματα εισόδου κατά την διάρκεια της μάθησης. Σαν αντικείμενα μακριών όρων μνήμης τα templates ή τα κωδικοποιημένα διανύσματα θα πρέπει να αποθηκευτούν σαν συνδέσεις βαρών απ' τις μονάδες εισόδου στις μονάδες εξόδου.

Ας πούμε ότι $x(t) = (x_1(t), x_2(t),...,x_n(t))^T$ είναι το Μ-διαστάσεων πραγματικό διάνυσμα εισόδου παρουσιαζόμενο στη σειρά μονάδας εισόδου τους χρόνους t=1,2,3... (για αποτελεσματικότητα της διαδικασίας μάθησης και μαθηματικής ευκολίας, όλα τα διανύσματα εισόδου είναι πάντα κανονικοποιημένα στο μήκος της μονάδας). Κατόπιν οι μονάδες αρχίζουν να ανταγωνίζονται η μία την άλλη. Η μονάδα νικητής c επιλέγεται βασιζόμενοι στην ελάχιστη απόσταση βάσης:

$$||m_c(t)-x||= \min||m_o(t)-x||$$

όπου $m_c(t)$ είναι η αντίστοιχη template της νικήτριας μονάδας c. Μετά την επιλογή της νικήτριας μονάδας, όλες οι μονάδες μέσα στην δική της γειτονιά ενημερώνονται (αναβαθμίζονται). Έστω $N_c(t)$ είναι η γειτονιά γύρω απ' τη μονάδα C το χρόνο t. Η N_c συνήθως τοποθετείται πολύ πλατιά στην αρχή για να αποκτήσει μία σκληρή σφαιρική διάταξη, και μετά συστέλλεται μονοτονικά με το χρόνο με σκοπό να βελτιώσει την αποτελεσματικότητα του χάρτη (κοίτα σχήμα 15). Αυτή η διαδικασία είναι κρίσιμη για την τοπολογική διάταξη του SOM. Η φόρμουλα ενημέρωσης του διανύσματος βαρών είναι:

$$m_i(t+1) = \begin{cases} m_i(t) + a(t)[x(t) - m_i(t)] & \text{Αν η μονάδα i είναι μέσα στη } N_o(t) \\ m_i(t) & \text{αλλιώς} \end{cases} \quad \text{(Εξ. 6.1)}$$

όπου $0 \leq a(t) \leq 1$ είναι ένα κέρδος προσαρμογής και μειώνεται με το χρόνο. Η σύγκλιση της παραπάνω σχέσης έχει αποδειχθεί μαθηματικά.

Μία εναλλακτική σημείωση στην εξίσωση είναι να θεωρήσουμε ότι a(t) είναι μία συνάρτηση του d_i της απόστασης μεταξύ της μονάδας i και της νικήτριας μονάδας c.

Τότε έχουμε:

$$m_i(t) = m_i(t) + a(t_i d_i)[x(t) - m_i(t)$$

$$a(t_i d_i) = \begin{cases} \text{aoexp}\left(d_i^2 \big/ \sigma^2(t) \right) & \text{αν } i \in N_i t \\ 0 & \text{αλλιώς} \end{cases} \quad \text{(Εξ. 6.2)}$$

Εδώ το κέρδος προσαρμογής α είναι μία συνάρτηση • και του χρόνου t και της απόστασης d_i. Όπως ορίστηκε, το κέρδος προσαρμογής α έχει το σχήμα της (Gaussian) Γκαουσιανής συνάρτησης. Η παράμετρος σ ορίζει το πλάτος του Gaussian σχήματος. Αυτό είναι η ακτίνα του παραθύρου της γειτονιάς. Η νικήτρια μονάδα συνήθως ενημερώνεται σύμφωνα με την εξίσωση ενημέρωσης βαρών Εξ. 6.2.

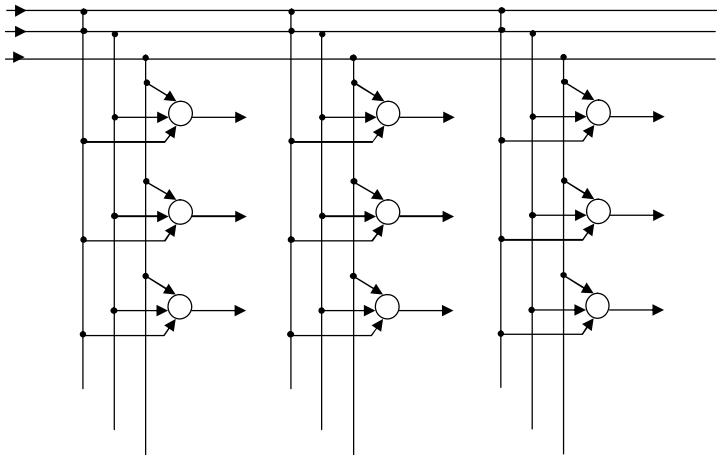
Καθώς η απόσταση από την νικήτρια μονάδα γίνεται μεγαλύτερη, το ποσοστό της αναβάθμισης βαρών μειώνεται. d_o είναι μία παράμετρος να ρυθμίσουμε τον ρυθμό μάθησης.

Σαν αποτέλεσμα της ανταγωνιστικής μάθησης με ένα δυναμικό παράθυρο γειτονιάς, το διάνυσμα βαρών (templates) είναι να προσεγγίσει την

συνάρτηση πυκνότητας πιθανότητας του διανύσματος εισόδου σε μία διαστημικά τοποθετημένη εκδοχή. Αυτό μπορεί να εξηγηθεί ως ακολούθως:

Στην αρχή τα templates είναι ακτινικά αρχικοποιημένα και ταυτόχρονα διασκορπισμένα ακτινικά στο Μ-διαστάσεων χώρο. Κατά την διάρκεια κάθε μάθησης, η ενημέρωση των βαρών δεν συμβαίνει μόνο στη νικήτρια μονάδα. Οι γειτονικές μονάδες επίσης υφίστανται τον ίδιο τύπο της ενημέρωσης βαρών. Γι' αυτό λέμε ότι το template νικητής μαζί με κάθε τοπολογικά γειτονικά templates ενημερώνονται για να είναι κοντύτερα στο διάνυσμα εισόδου.

Στο σχήμα 16 φαίνεται το σχηματικό διάγραμμα ενός δισδιάστατου πλέγματος νευρώνων που συνήθως χρησιμοποιείται ως χάρτης. Κάθε νευρώνας του πλέγματος συνδέεται πλήρως με όλους τους κόμβους του επιπέδου της εισόδου. Το δίκτυο αναπαριστά μία δομή πρόσθιας τροφοδότησης όπου οι νευρώνες διατάσσονται σε γραμμές και στήλες.



Σχήμα 15

Ο αλγόριθμος με τον οποίο εκπαιδεύεται ένα NN SOM είναι ο LVQ (LVQ_1, LVQ_2) που παρουσιάστηκε στο προηγούμενο κεφάλαιο. Παρακάτω δίνουμε μία περίληψη του LVQ που ονομάζεται και αλγόριθμος του Kohonen.

Βήμα 1: Αρχικοποίησε τα βάρη: Τα βάρη συνδέσεων των Ν κόμβων εισόδου με τους Μ κόμβους εξόδου αρχικοποιούνται σε μικρές, τυχαίες τιμές.

Βήμα 2: Εφάρμοσε μία νέα είσοδο.

Βήμα 3: Υπολόγισε την απόσταση της εισόδου από όλους τους κόμβους: Υπολόγισε τις αποστάσεις d_j μεταξύ της εισόδου και κάθε κόμβου εξόδου j σύμφωνα με την σχέση:

$$d_j = \sum_{i=0}^{N-1} \{x_i(t) - w_{ij}(t)^2$$

όπου $x_i(t)$ είναι η είσοδος στον κόμβο i τη χρονική στιγμή t και $W_{ij}(t)$ είναι το βάρος της σύνδεσης του κόμβου εισόδου i με τον κόμβο εξόδου j την χρονική στιγμή t.

Βήμα 4: Επέλεξε τον κόμβο εξόδου με την ελάχιστη απόσταση:

Επίλεξε τον κόμβο j, εκείνον με την ελάχιστη απόσταση d_j.

Βήμα 5: Προσάρμοσε τα βάρη προς τον κόμβο j και τους γειτονικούς του. Τα βάρη προς τον κόμβο j και όλης της γειτονιάς $NE_j(t)$ προσαρμόζονται σύμφωνα με την σχέση:

$$w_{ij}(t+1) = w_{ij}(t) + n(t)\,(x_i(t) - w_{ij}(t))$$

$$\text{για } j \in Nej(t) \qquad 0 \le i \le N-1$$

όπου n(t) είναι ένας όρος κέρδους $0<n(t)<1$ όπου ελαττώνεται με το χρόνο.

Βήμα 6: Επανάλαβε από το βήμα 2.

Κεφάλαιο 7
Αναδρομικά Νευρωνικά δίκτυα και Αυτοσυσχετιστικές μνήμες

Εισαγωγή

Στην κατηγορία των αναδρομικών (recurrent) νευρωνικών δικτύων ανήκουν τα NN Hopfield. Το δίκτυο Hopfield είναι δυαδικών εισόδων και χρησιμοποιείται όταν είναι δυνατή η δυαδική παράσταση στων μεγεθών του προβλήματος (ασπρόμαυρες εικόνες, ACII παράσταση χαρακτήρων κ.λ.π.) ενώ δεν είναι κατάλληλο για προβλήματα όπου η ψηφιοποίηση δεν ενδείκνυται. Ορισμένες χρήσεις του δικτύου Hopfield είναι η συσχετιστική μνήμη και η λύση προβλημάτων μεγιστοποίησης. Πριν παρουσιάσουμε την δομή και τους αλγόριθμους μάθησης του Hopfield θα αναφερθούμε στην συσχετιστική μνήμη.

7.1. Συσχετιστική μνήμη

Τα ΝΝ μας προσφέρουν ένα νέο μηχανισμό αποθήκευσης προτύπων (εικόνες, λέξεις και ότι μπορεί να περιγραφεί από ένα διάνυσμα) παρόλο που το συνηθισμένο αποθηκευτικό μέσο μέχρι σήμερα είναι τα μαγνητικά υλικά και οι ημιαγωγοί. Ο νέος αποθηκευτικός μηχανισμός έχει εμπνευσθεί απ' το μυαλό μας. Έχει διαπιστωθεί ότι το μυαλό αποθηκεύει πληροφορίες τροποποιώντας τις συνδέσεις μεταξύ των νευρώνων. Τέτοιες τροποποιημένες συνάψεις κόμβων είναι διανεμημένες. Δεν μπορούμε να αναγνωρίσουμε ποιες τροποποιήσεις αντιστοιχούν σε κάποιο κομμάτι πληροφορίας.

Η τεχνητή μνήμη (Artificial memory) είχε την μορφή της τοπικής μνήμης πριν την αρχή της έρευνας των ΝΝ. Σκεφτείτε τον τρόπο που φτιάχνουμε αρχεία ή την μέθοδο με την οποία γράφουμε το ημερολόγιό μας. Στο ημερολόγιο όλα τα μηνύματα τοποθετούνται με χρονολογική σειρά. Τοποθετούμε αυτά τα μηνύματα συγκεκριμένων συμβάντων στην ημερομηνία που έγιναν. Όταν ψάχνουμε για ένα μήνυμα πρέπει πρώτα να βρούμε την ημερομηνία, η οποία λειτουργεί σαν διεύθυνση. Στις μέρες μας η αποθήκευση πληροφορίας σε ένα Η/Υ είναι ακόμα στην μορφή της τοπικής μνήμης. Δηλαδή η ανάκτηση ή το γράψιμο πληροφορίας γίνεται μέσω διευθύνσεων. Οι

διευθύνσεις σχηματοποιούνται σε όρους μιας ελάχιστης μονάδας όπως το byte για την μνήμη ημιαγωγών και τον τομέα (sector) για μαγνητική αποθήκευση. Όταν ερευνούμε για ένα ειδικό αντικείμενο δεδομένου, πρέπει πρώτα να βρούμε την σωστή διεύθυνσή του. Δεν υπάρχει καμία απόδειξη ότι στο μυαλό μας η αποθήκευση πληροφορίας είναι τοπική. Ανακαλούμε πληροφορίες απ' το περιεχόμενό τους. Η μνήμη που λειτουργεί μ' αυτό τον τρόπο αναφέρεται σαν περιεχόμενη – διευθυνσιακή μνήμη. Αυτό το είδος μνήμης μπορούμε να πούμε ότι χρησιμοποιούν οι βιβλιοθήκες και οι βάσεις δεδομένων επειδή χρησιμοποιούν key-words (λέξεις – κλειδιά) για την εύρεση ενός βιβλίου ή αντικειμένου των δεδομένων οι οποίες αναφέρονται στο περιεχόμενο των βιβλίων και των (data item) στοιχείων και δεδομένων.

Στην πραγματικότητα όμως είναι μία εικονική περιεχόμενη – διευθυνσιακή μνήμη και αυτό γιατί οι λέξεις-κλειδιά αντιστοιχούν σε διευθύνσεις.

Η περιεχόμενη-διευθυνσιακή μνήμη (content-addreassable memory) αναφέρεται και σαν συσχετιστική μνήμη (associative) ή διανεμημένη (distributed) μνήμη. Η συσχετιστική μνήμη έχει το ίδιο νόημα με την περιεχόμενη-διευθυνσιακή μνήμη. Η ικανότητα της content-addressable μνήμης πετυχαίνεται μέσω συσχέτισης. Υπάρχουν δύο τύποι συσχέτισης: η αυτοσυσχέτιση και η ετεροσυσχέτιση. Αυτοσυσχέτιση σημαίνει ότι ένα κομμάτι πληροφορίας συσχετίζεται με τον εαυτό του στην διαδικασία της αποθήκευσής του. Στην φάση επανάκτησης, κάποιος μπορεί να ανακαλέσει την πληροφορία μέσω ενός μέρους ή μιας υποβαθμισμένης εκδοχής αυτού του κομματιού πληροφορίας. Με την ετεροσυσχέτιση ένα κομμάτι πληροφορίας συσχετίζεται με ένα άλλο κομμάτι πληροφορίας. Αυτά τα δύο κομμάτια πληροφορίας μπορούν να έχουν κάποια σχέση όπως ίδια ή αντίθετη σημασία/ εμφάνιση, να είναι στενά συσχετισμένα κ.τ.λ. Αφού τα δύο κομμάτια πληροφορίας αποθηκεύονται με τον τρόπο της ετερο-συσχέτισης, το ένα κομμάτι πληροφορίας μπορεί να ανακαλέσει το άλλο.

Η συσχετιστική μνήμη έχει ανυψώσει τον μηχανισμό μνήμης από ένα απλό μηχανικό επίπεδο σε ένα σημασιολογικό (semantic) επίπεδο. Προσφέρει αποτελεσματικά εργαλεία για αποθήκευση πληροφορίας και ανάκτησή της. Είναι αναμενόμενο ότι με την συσχετιστική μνήμη κάποιες συναρτήσεις του ανθρώπινου εγκεφάλου μπορούν να προσομοιωθούν.

Πολλά μοντέλα ΝΝ για συσχετιστική μνήμη έχουν προταθεί απ' τους Amari, Kohonen και Hopfield. Τα προσοτροφοδοτούμενα δίκτυα και τα περισσότερα απ' τα αυτοοργανούμενα δίκτυα που έχουν περιγραφεί στα προηγούμενα κεφάλαια είναι γραμμικά, αφού δεν υπάρχει ανατροφοδότηση στα δίκτυα. Το δίκτυο Hopfield είναι αντιπροσωπευτικό της μη γραμμικής συσχετιστικής μνήμης. Η διανεμητική ιδιότητα του Hopfield δικτύου είναι πιο φανερή: Κάποιος δεν μπορεί να ξέρει ποιο αποθηκευμένο πρότυπο σχετίζεται με μία συγκεκριμένη θέση σύνδεση και καταστροφή μέρους των συνδέσεων μειώνει τον αριθμό των αποθηκευμένων προτύπων.

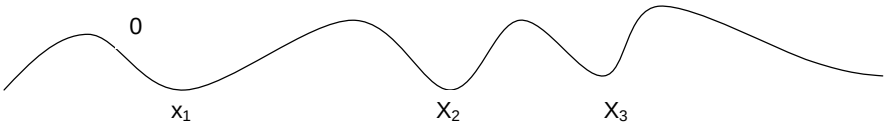
7.2. Φυσικό ανάλογο της μνήμης

Με σκοπό την ευκολότερη κατανόηση της λειτουργίας των δικτύων που υλοποιούν αυτοσυσχέτιση θεωρούμε το ακόλουθο φυσικό ανάλογο. Έστω ότι έχουμε ένα κοίλο δοχείο όπως αυτό του σχήματος 17.

Σχήμα 17

Αν τοποθετήσουμε μία σφαίρα στο εσωτερικό του δοχείου αυτή θα ταλαντευτεί και τελικά θα ισορροπήσει στον πάτο του. Η αρχική κατάσταση της σφαίρας είναι ασταθής και η τελική ευσταθής. Αξιοσημείωτο είναι ότι η σφαίρα ισορροπεί πάντα στο ίδιο σημείο το οποίο καθορίζεται από το ελάχιστο της συνολικής ενέργειας του συστήματος δοχείου – σφαίρας. Η τελική κατάσταση λέγεται ευσταθής διότι η σφαίρα παραμένει σε αυτή για πάντα. Αν μετακινήσουμε την σφαίρα κατά Δx απ' την θέση ισορροπίας x_o τότε αυτή θα ισορροπήσει στην θέση x_o. Ομοίως αν μεταβάλλουμε το διάνυσμα κατάστασης του δικτύου δηλαδή τα βάρη του κατά ΔW τότε το δίκτυο μετά από μερικές ταλαντεύσεις θα ισορροπήσει στην ευσταθή κατάσταση του x_o. Το x_o είναι μία αποθηκευμένη κατάσταση ισορροπίας.

Αν τώρα χρησιμοποιήσουμε μία κυματοειδή επιφάνεια αντί για ένα δοχείο μπορούμε να αποθηκεύσουμε πολλές μνήμες (καταστάσεις ισορροπίας), όπως στο σχήμα 18.



Σχήμα 18: {x_1, x_2,x_3...} είναι οι αποθηκευμένες μνήμες

Σε οποιοδήποτε μέρος της επιφάνειας και αν τοποθετήσουμε την σφαίρα, αυτή θα καταλήξει σε μία θέση ισορροπίας και μάλιστα στο κοντινότερο τοπικό κοίλο (τοπικό ελάχιστο). Επομένως ανακαλεί το πλησιέστερο αποθηκευμένο πρότυπο. Υπάρχουν λοιπόν δύο αλληλένδετοι τρόποι αντιμετώπισης του θέματος. Σύμφωνα με τον πρώτο, λέμε ότι το σύστημα οδηγείται σε ενεργειακό ελάχιστο ενώ σύμφωνα με το δεύτερο αποθηκεύουμε ένα σύνολο προτύπων και ανακαλούμε αυτό που βρίσκεται πλησιέστερα στην αρχική θέση. Το κοινό σημείο των δύο περιγραφών είναι ότι και στις δύο περιπτώσεις το σύστημα ή το δίκτυο καταλήγει στην θέση ισορροπίας.

7.3. Ανατροφοδοτούμενα Νευρωνικά Δίκτυα

Εισαγωγή

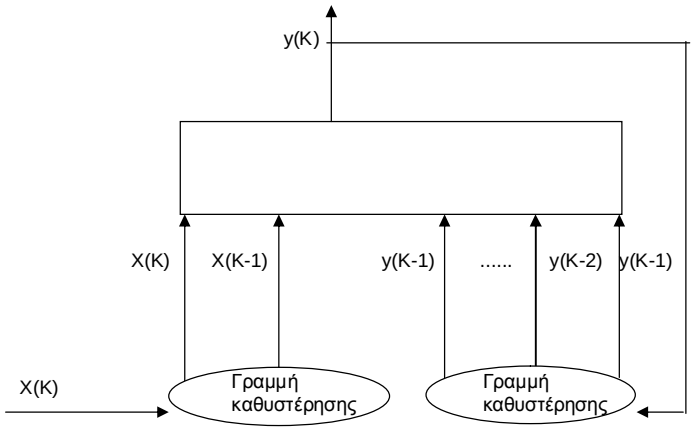
Τα ανατροφοδοτούμενα Νευρωνικά δίκτυα (Feedback Neural Networks FNN) χωρίζονται σε δύο κατηγορίες:

i. NN με ανατροφοδότηση εξόδου
ii. NN με ανατροφοδότηση κατάστασης (state feedback)

Ανήκουν στην γενική κατηγορία των δυναμικών (dynamic) NN και διαθέτουν ως συνάρτηση μεταφοράς μία εξίσωση διαφορών ή μία διαφορική εξίσωση. Εφαρμογές αυτών των δικτύων συναντάμε σε συστήματα ελέγχου αεροπλάνων, πυραύλων, διαστημοπλοίων, ρομπότ κ.λ.π.

7.3.1. FNN εξόδου

Τα FNN εξόδου προτάθηκαν από τον Narendra και αποτελούν έναν απλό τρόπο για να ενσωματώσουμε κάποιο σύστημα ανατροφοδότησης εξόδου σε ένα NN, ανατροφοδοτώντας ορισμένες χρονικά καθυστερημένες εξόδους του NN στην είσοδό του (Σχήμα 19).

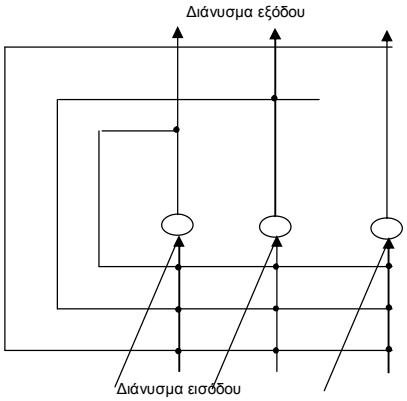


Σχήμα 19. Γενικό Παράδειγμα FNN

Οι δύο γραμμές καθυστέρησης μπορούν να τροφοδοτούν διαφορετικά NN οι έξοδοι των οποίων να τροφοδοτούν κάποιο άλλο NN.

7.3.2. FNN κατάστασης

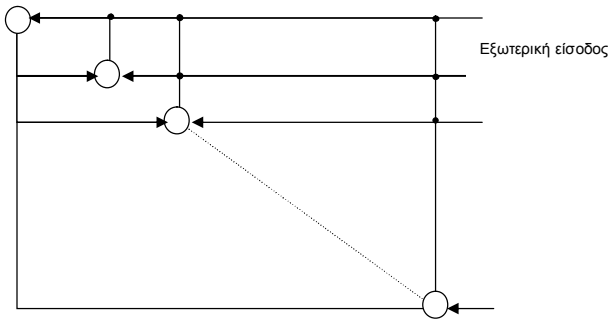
Αυτός ο τύπος FNN αποτελείται συνήθως από ένα μόνο επίπεδο. Στη γενική περίπτωση, η έξοδος κάθε νευρώνα ανατροφοδοτείται σε όλους τους άλλους νευρώνες αλλά και στην δική του είσοδο. Επομένως η έξοδος κάθε νευρώνα μπορεί να είναι τμήμα του διανύσματος κατάστασης. Τέλος οι νευρώνες μπορεί να δέχονται εισόδους απ' το εξωτερικό περιβάλλον Σχήμα 20.



Σχήμα 20. FNN κατάστασης

7.3.3. Δίκτυο Hopfield

Ο τύπος αυτός ΝΝ προτάθηκε από τον Hopfield και ανήκει στην κατηγορία των ανατροφοδοτούμενων ΝΝ με ανατροφοδότηση καταστάσεων. Στο σχήμα 21 παρουσιάζεται ένα δίκτυο Hopfield αποτελούμενο από Ν δυαδικές μονάδες.



Σχήμα 21: Δίκτυο Hopfield. Δεν υπάρχει διαχωρισμός μεταξύ των μονάδων εισόδου, κρυμμένων και εξόδου.

Όπως φαίνεται στο παραπάνω σχήμα κάθε νευρώνας ανατροφοδοτεί την «έξοδό» του σε όλους τους άλλους νευρώνες εκτός από τον εαυτό του. Όταν παρουσιασθεί ένα πρότυπο εισόδου, όλες οι μονάδες εξασφαλίζουν την αρχική τους κατάσταση απ' το πρότυπο εισόδου. Όταν το δίκτυο αντιληφθεί

μία σταθερή κατάσταση, η έξοδος αναπαριστάται απ' τις καταστάσεις των N μονάδων σαν μία δυαδική λέξη Nbits. Όπως είδαμε προηγουμένως η μελέτη των φυσικών συστημάτων μας λεει ότι τα αποθηκευμένα πρότυπα είναι τα ελάχιστα της συνάρτησης ενέργειας. Η συνάρτηση ενέργειας χαρακτηρίζει το δίκτυο και είναι η εξής:

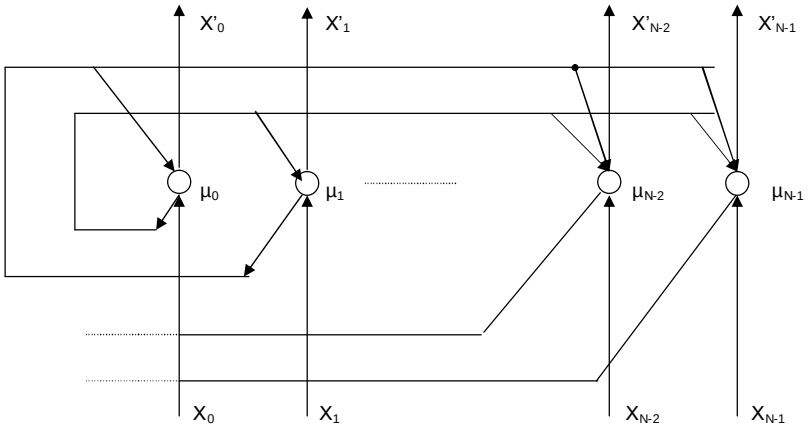
$$E(t) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j-1}^{n} y_i(t)y_j(t)w_{ij} + i = 1\sum^{n} I_i y_i(t)$$

όπου I_i = i-οστή μοναδιαία μήτρα

y_i = i-οστή είσοδος

w_{ij} = Βάρη των συνδέσεων

Τα βάρη των συνδέσεων είναι συμμετρικά δηλαδή για δύο οποιουσδήποτε κόμβους i και j τα βάρη είναι ίσα και προς τις δύο κατευθύνσεις ($w_{ij}=w_{ji}$ και w_{ii} =0). Επίσης οι πολώσεις θ_i θεωρούμε ότι είναι μηδενικές (θ_i = 0). Παρακάτω δίνουμε μία άλλη μορφή του Hopfield που δεν διαφέρει απ' την προηγούμενη απλώς εδώ δείχνουμε και την έξοδο του δικτύου που είναι έγκυρη μετά την σύγκλιση του αλγορίθμου εκπαίδευσης του Hopfield.



Σχήμα 22: Δίκτυο Hopfield

Παρακάτω παρουσιάζουμε τον αλγόριθμο μάθησης του δικτύου Hopfield:

Βήμα 1: Καθόρισε τα βάρη στις συνδέσεις

$$w_{ij} = \begin{cases} \sum_{s=0}^{M-1} x_i^s x_j^s, i \neq j \\ 0, i = j, 0 \leq i \leq N-1 \end{cases}$$

όπου w_{ij} είναι το βάρος της σύνδεσης του κόμβου i με τον κόμβο J, x_i^s είναι το στοιχείο i του παραδειγματικού προτύπου της κλάσης s.

Βήμα 2: Αρχικοποίησε ως προς το άγνωστο πρότυπο εισόδου

$$\mu_i(0)=x_i, 0 \leq i \leq N-1$$

όπου $\mu_i(t)$ είναι η έξοδος του κόμβου i στο χρόνο t και x_i είναι το στοιχείο i της εισόδους (+1 ή –1).

Βήμα 3: Επανάλαβε ως την σύγκλιση

$$\mu_i(t+1) = f_h\left(\sum_{i=0}^{N-1} w_{ij}\mu_i(t)\right), 0 \leq j \leq M-1$$

Η διαδικασία επαναλαμβάνεται ώσπου να παραμείνει αμετάβλητη η έξοδος των κόμβων για περισσότερες επαναλήψεις. Τότε η έξοδος αυτή παριστάνει το παραδειγματικό πρότυπο που ταιριάζει περισσότερο στην άγνωστη είσοδο.

Βήμα 4: Επανάλαβε από το βήμα 2.

7.3.4. Ικανότητα αποθήκευσης

Πόσο καλή είναι η μέθοδος καθορισμού των βαρών ώστε τα πρότυπα που θέλουμε να αποθηκεύσουμε να αποτελούν ευσταθείς καταστάσεις του δικτύου; Όσο αυξάνεται ο αριθμός m των προτύπων τόσο μειώνεται η πιθανότητα για ακριβή αποθήκευση. Ο Mc Cliece το 1987 έδειξε θεωρητικά ότι, στην περίπτωση που τα πρότυπα εκπαίδευσης είναι γραμμικώς ανεξάρτητα η χωρητικότητα ενός δικτύου Hopfield με N κόμβους είναι τάξης 0(N/zlogN). Μπορούμε να αποδείξουμε ότι ο μέγιστος αριθμός αποθηκευμένων προτύπων σε ένα δίκτυο Hopfield είναι 2^{N-2}.

Η σχέση βγαίνει με τον εξής συλλογισμό: 2^{N-1} είναι το σύνολο των καταστάσεων (ευσταθών και ασταθών) που μπορεί να αποθηκευτεί σε ένα Hopfield N κόμβων. Μόνο οι μισές απ' αυτές μπορούν να είναι τοπικά ελάχιστα δηλαδή ευσταθείς καταστάσεις ισορροπίας και αυτό θα συμβεί όταν δίπλα από ένα ελάχιστο βρίσκεται ένα μέγιστο (στην καλύτερη περίπτωση).

Άρα το σύνολο των αποθηκευμένων προτύπων δηλαδή των τοπικών ελαχίστων είναι $\frac{2^{N-1}}{2} = 2^{N-2}$

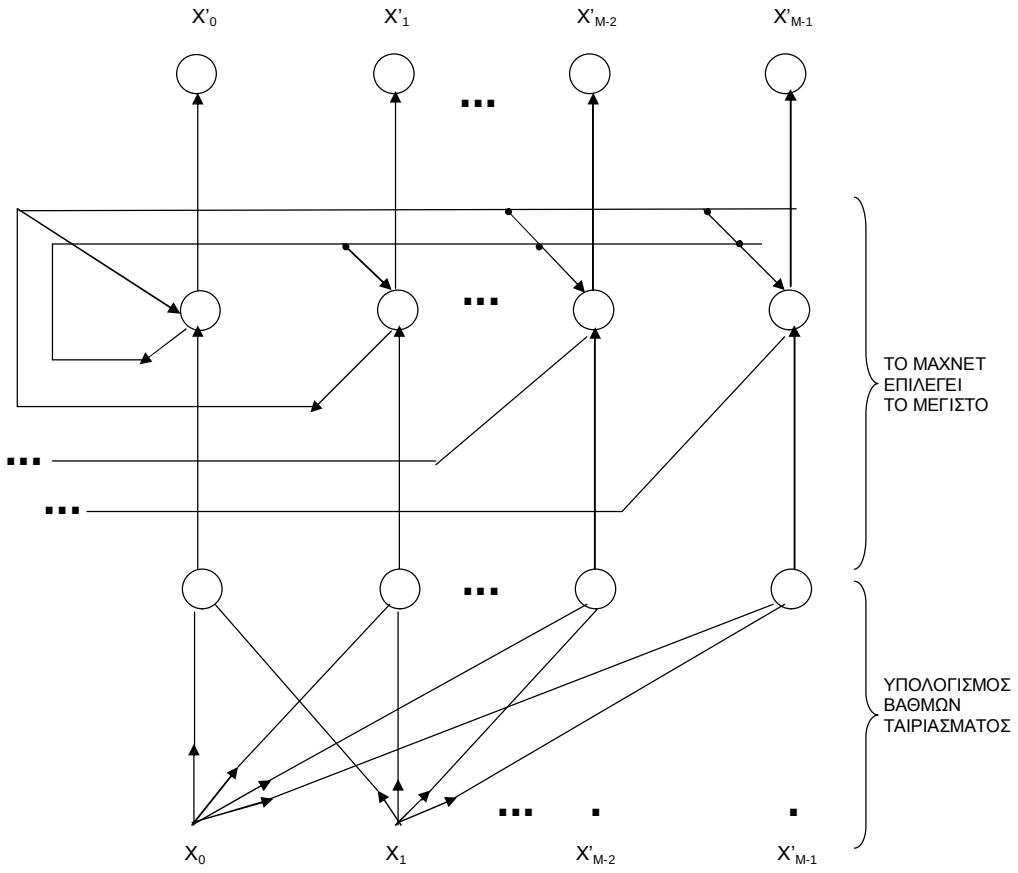
7.4. Άλλοι τύποι Νευρωνικών Δικτύων

Εισαγωγή

Παρακάτω παρουσιάζουμε συνοπτικά δύο άλλους τύπους ΝΝ: το δίκτυο Hamming και το δίκτυο Caprenter-Grossberg. Ένα κλασικό πρόβλημα της θεωρίας τηλεπικοινωνιών είναι όταν οι είσοδοι έχουν πολλά bits που αντιστρέφονται τυχαία και ανεξάρτητα με δεδομένη πιθανότητα και λαμβάνει χώρα όταν σήματα σταθερού μεγέθους στέλνονται μέσω καναλιών χωρίς μνήμη.

7.4.1. Δίκτυο Hamming

Στην παραπάνω περίπτωση ο βέλτιστος ταξινομητής ελαχίστου σφάλματος υπολογίζει την απόσταση Hamming της εισόδου από το παραδειγματικό πρότυπο της κάθε κλάσης και εκλέγει αυτό με την μικρότερη απόσταση. Η απόσταση Hamming ορίζεται ως ο αριθμός των bits εισόδου που δεν συμφωνούν με το εξεταζόμενο παραδειγματικό πρότυπο. Το δίκτυο Hamming υλοποιεί τον αλγόριθμο αυτό με εφαρμογή του στα νευρωνικά δίκτυα όπως φαίνεται στο σχήμα 22.



Σχήμα 22. Δίκτυο Hamming

Παρακάτω δίνουμε τον αλγόριθμο μάθησης τους δικτύου Hamming.

Βήμα 1: Καθόρισε τα βάρη στις συνδέσεις και τα κατώφλια.

Στο κάτω υποδίκτυο:

$$w_{ij} = \frac{x_i^j}{2}, \theta_j = \frac{N}{2}, 0 \le N-1, 0 \le j \le M-1$$

Στο πάνω υποδίκτυο

$$t_{KL} = \begin{cases} 1, K = \ell \\ -\epsilon, K \ne \ell \end{cases} \quad \varepsilon < \frac{1}{M}, \quad 0 \le K, \ell \le M-1$$

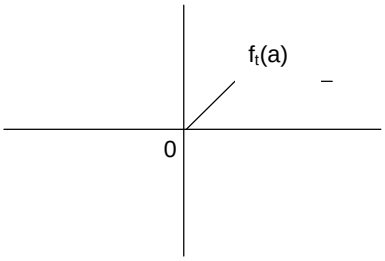
όπου w_{ij} είναι τα βάρη της σύνδεσης της εισόδου i με τον κόμβο j στο κάτω υποδίκτυο και θ_i το κατώφλι στον κόμβο αυτό, $t_{K\ell}$ το βάρος σύνδεσης του

κόμβου K με τον κόμβο ℓ στο πάνω υποδίκτυο, όπου όλα τα κατώφλια είναι μηδέν, x_i είναι το i-οστό στοιχείο του παραδειγματικού προτύπου της j κλάσης.

Βήμα 2: Αρχικοποίησε ως προς το άγνωστο πρότυπο εισόδου:

$$\mu_j(0) = f_t\left(\sum_{i=0}^{N-1} w_{ij}x_i - \theta_i\right), \qquad 0 \le j \le M-1$$

όπου μ_j η έξοδος του κόμβου j στο πάνω υποδίκτυο τη χρονική στιγμή t, x_i είναι το i-οστό στοιχείο της εισόδου και f_t είναι η threshold logic μη γραμμικότητα του παρακάτω σχήματος.



THRESHOLD LOGIC

Βήμα 3: Επανάλαβε ως της σύγκλιση:

$$\mu_\xi(\tau+1) = f_t\left(\mu_j(t) - \in \sum_{k\neq j}\mu_k(t)\right), \qquad 0 \le j,k \le M-1$$

Η διαδικασία αυτή επαναλαμβάνεται ως τη σύγκλιση, που σημαίνει μόνο μία από τις εξόδους να παραμείνει θετική.

Βήμα 4: Επανάλαβε από το Βήμα 2.

Το δίκτυο Hamming έχει σειρά πλεονεκτημάτων σε σχέση με το δίκτυο Hopfield. Το πιο σημαντικό είναι ο πολύ μικρότερος αριθμός κόμβων και συνδέσεων που απαιτεί.

7.4.2. Δίκτυο Carpenter-Grossberg

Το δίκτυο Carpenter-Grossberg ενώ μοιάζει δομικά με αυτό του Hamming παρουσιάζει τρία νέα στοιχεία: είναι ανατροφοδοτούμενο, κάθετη σύνδεση (μεταξύ των y_0,y_1) που ενισχύει την διαφορά των y_0,y_1 και τα βάρη από τα x προς τα y και αντίστροφα προσαρμόζονται, δηλαδή το δίκτυο μαθαίνει.

Ποιοτικά το δίκτυο λειτουργεί ως εξής: όταν έρχεται μία είσοδος εξετάζεται αν ταιριάζει με τις ήδη υπάρχουσες κλάσεις. Ο έλεγχος αυτός πραγματοποιείται με την βοήθεια της σύγκρισης του βαθμού ταιριάσματος με έναν αριθμό από 0 έως 1 που καλείται κατώφλι εγρήγορσης (vigilance threshold) και καθορίζεται από τις απαιτήσεις του δικτύου. Ο αλγόριθμος μάθησης του δικτύου παρουσιάζεται παρακάτω:

Βήμα 1: Αρχικοποίηση

- $t_{ij}(0)=1, \quad 0 \le i \le N-1, \quad 0 \le j \le M-1$
- $b_{ij}(0)=\frac{1}{1+N}, 0 \le i \le N-1, \qquad 0 \le j \le M-1$
- Δώσε τιμή στο p, $0 \le p \le 1$

Στις εξισώσεις, $b_{ij}(t)$ είναι τα προσοτροφοδοτούμενα βάρη (feed forward) και $t_{ij}(t)$ τα ανατροφοδοτούμενα (feedback) βάρη στο χρόνο t. Τα βάρη αυτά παριστάνουν το παραδειγματικό πρότυπο. Επίσης p είναι το κατώφλι εγρήγορσης που εκφράζει το πόσο κοντά πρέπει να είναι η είσοδος στο παραδειγματικό πρότυπο για να πούμε ότι ταιριάζουν.

Βήμα 2: Εφάρμοσε την νέα είσοδο

Βήμα 3: Υπολόγισε τους βαθμούς ταιριάσματος

$$\mu_j = \sum_{i=0}^{N-1} b_{ij}(t)x_i, \quad 0 \le j \le M-1$$

όπου μ_j η έξοδος του κόμβου εξόδου j

Βήμα 4: Επίλεξε το παραδειγματικό πρότυπο που ταιριάζει περισσότερο:

$$\mu_{j^*} = \max_j \{\mu_j\}$$

που βρίσκεται με οποιαδήποτε από τις δομές που περιγράφηκαν στην προηγούμενη παράγραφο και υπολογίζουν το μέγιστο από N στοιχεία.

Βήμα 5: Εξέταση με το κατώφλι εγρήγορσης:

$$\|x\| = \sum_{i=0}^{N-1} x_i$$

$$\|T \cdot X\| = \sum_{i=0}^{N-1} t_{ij} x_i$$

Ισχύει ότι $\frac{\|T \cdot X\|}{\|X\|} > p$?

Αν ναι, πήγαινε στο βήμα 7

Αν όχι, πήγαινε στο βήμα 6

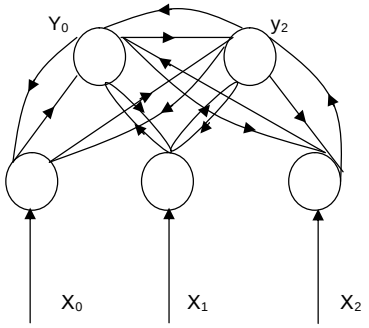
Βήμα 6: Απενεργοποίησε το παραδειγματικό πρότυπο που ταιριάζει πιο πολύ στην είσοδο. Η έξοδος του καλύτερου ταιριάσματος που βρέθηκε στο βήμα 4, τίθεται προσωρινά ίση με το μηδέν και πηγαίνουμε στο Βήμα 3.

Βήμα 7: Προσάρμοσε τα βάρη του παραδειγματικού προτύπου:

$$t_{ij}(t+1) = t_{ij}(t)\cdot x_i$$

$$b_{ij}(t+1)=\frac{t_{ij}(t)\cdot x_i}{0{,}5+\sum_{i=0}^{N-1} t_{ij}(t)x_i}$$

(επανενεργοποίησε πρώτα όλους τους κόμβους που απενεργοποιήθηκαν στο Βήμα 6).



Σχήμα 23. Δίκτυο Carpenter-Grossberg

Στο παραπάνω σχήμα φαίνεται το δίκτυο του Carpenter-Grossberg. Για λόγους απλότητας δεν παρουσιάζεται όλο το δίκτυο που περιέχει επιπρόσθετα στοιχεία.

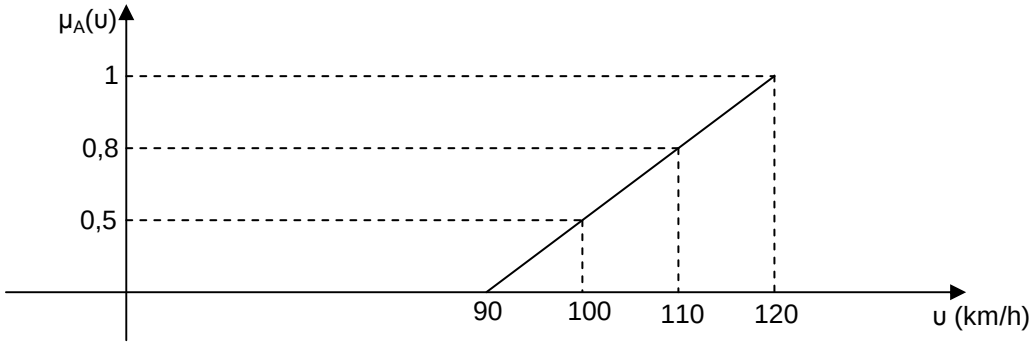
ΕΙΣΑΓΩΓΗ

Σ’ αυτό το τμήμα της διπλωματικής θα ασχοληθούμε με την ασαφή λογική, τα ασαφή συστήματα και τον ασαφή έλεγχο (fuzzy logic, fuzzy system, fuzzy control αντίστοιχα). Αφού παρουσιάσουμε την ασαφή συλλογιστική καθώς και τις εφαρμογές της θα κάνουμε σύγκριση των ασαφών και νευρωνικών συστημάτων και θα δούμε τα αποτελέσματα της «συνεργασίας» τους που μας δίνουν τα νευροασαφή συστήματα (neurofuzzy system).

Τα ασαφή συστήματα (εκμεταλλεύονται) «κάνουν χρήση» ασαφών κανόνων. Οι κανόνες αυτοί λέγονται ασαφής γιατί δεν είναι ποσοτικά ακριβής. Αυτούς τους κανόνες μπορούμε να τους πάρουμε από έναν έμπειρο χρήστη του μηχανήματος, συστήματος που θέλουμε να ελέγξουμε. Ας πάρουμε για παράδειγμα τον τρόπο που κάποιος μαθαίνει να οδηγεί. Ανάβουμε τη μηχανή (κανόνας σαφώς ορισμένος), πατάμε το συμπλέκτη και βάζουμε ταχύτητα (κανόνας σαφώς ορισμένος). Αφήνουμε σιγά σιγά το συμπλέκτη ενώ πατάμε λίγο το γκάζι. Αυτός ο κανόνας είναι ασαφώς ορισμένος. Πόσο «σιγά» και πόσο «λίγο» θα αφήσουμε το συμπλέκτη και θα πατήσουμε το γκάζι;

Με αυτόν τον τρόπο είμαστε αναγκασμένοι να ορίσουμε τις γλωσσικές μεταβλητές που θα αντιστοιχούν στο «σιγά» και στο «λίγο». Ο τρόπος με τον οποίο συνδυάζονται οι ασαφείς κανόνες με τις γλωσσικές μεταβλητές δηλαδή με τις εισόδους του συστήματος ώστε να παράγουμε αποτελέσματα (έξοδος) μας δίνεται απ’ την ασαφή λογική. Με την ασαφή λογική ποσοτικοποιούμε (μαθηματικοποιούμε) την υποκειμενικότητα των ασαφών κανόνων δηλαδή την υποκειμενικότητα της ανθρώπινης σκέψης («λίγο», «σιγά»). Αυτό γίνεται με την εισαγωγή της θεωρίας των ασαφών συνόλων (fuzzy sets). Στην κλασική θεωρία συνόλων το κάθε σύνολο είναι σαφώς ορισμένο π.χ. το σύνολο που ορίζεται απ’ τη σχέση $60 \le \upsilon \le 100$ όπου υ ταχύτητα, έχει συγκεκριμένα στοιχεία, τιμές της ταχύτητας. Η τιμή $\upsilon = 70\frac{\text{km}}{\text{h}}$ ανήκει στο σύνολο , ενώ η τιμή υ=120 km/h δεν ανήκει. Ξέρουμε δηλαδή για κάθε τιμή της μεταβλητής υ αν ανήκει ή όχι στο σύνολο $60 \le \upsilon \le 100$.

Το παραπάνω σύνολο είναι σαφώς ορισμένο. Αν ορίσουμε όμως το σύνολο των «μεγάλων» ταχυτήτων τότε αυτό το σύνολο είναι ασαφώς ορισμένο. Για τις τιμές της ταχύτητας για τις οποίες ισχύει $\upsilon = 70\frac{\text{km}}{\text{h}}$ ξέρουμε ότι ανήκουν στο σύνολο, ενώ για υ ≤ 90 δεν ανήκουν. Έτσι είμαστε υποχρεωμένοι να ορίσουμε συναρτήσεις συμμετοχής οι οποίες θα καθορίζουν πόσο συμμετέχει κάθε τιμή της ταχύτητας στο σύνολο αυτό. Έτσι για τις τιμές υ ≥ 120 η συνάρτηση συμμετοχής θα είναι μ(υ)=1 και για υ ≤ 90 μ(υ)=0. Η τιμή υ=100 km/h θα συμμετέχει στο σύνολο κατά ένα δεδομένο μικρό ποσό δηλαδή μ(100)=0,5. Προκειμένου να γίνουμε πιο σαφής, το ποσό που συμμετέχει η κάθε τιμή της ταχύτητας στο σύνολο (ασαφές) φαίνεται στο παρακάτω σχήμα.



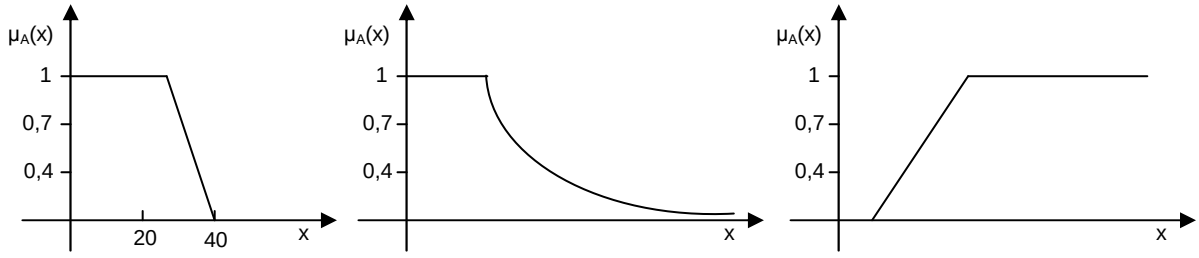
Είναι φανερό ότι η υ=110 km/h συμμετέχει στο σύνολο με συνάρτηση συμμετοχής $\mu_A(110)=0,8$. Το υπόλοιπο 0,2 δεν συμμετέχει στο σύνολο.

Η θεωρία των ασαφών συνόλων εισήχθη από τον Lofti Zadeh και αποτελεί όπως είδαμε επέκταση της κλασικής θεωρίας και αντίληψης των συνόλων.

ΚΕΦΑΛΑΙΟ 8
ΑΣΑΦΗΣ ΛΟΓΙΚΗ

8.1. Ασαφή σύνολα

Τα κλασικά σύνολα είναι υποσύνολα των ασαφών συνόλων όπως ακριβώς το σύνολο των πραγματικών αριθμών είναι υποσύνολο των μιγαδικών. (Αν θέσουμε Imz=0). Έστω λοιπόν ότι έχουμε ένα κλασικό σύνολο Α. Πάνω στο σύνολο αυτό μπορούμε να ορίσουμε μια συνάρτηση συμμετοχής $\mu_A(x)=\begin{cases}1, & x\in A\\ 0, & x\notin A\end{cases}$ η οποία θα παίρνει την τιμή μ·1 αν το $x\in A$ και την τιμή 0 αν $x\notin A$. Η θεωρία ασαφών συνόλων αυτό που κάνει είναι να αλλάζει τη μορφή της $\mu_A(x)$ ώστε να παίρνει τιμές σε όλο το διάστημα [0,1]. Το σύνολο τιμών της $\mu_A(x)$ περιορίζεται μεταξύ 0 και 1 γιατί εκφράζει το ποσοστό συμμετοχής του x στο σύνολο. Η $\mu_A(x)$ μπορεί να έχει οποιαδήποτε μορφή. Παρακάτω δίνουμε μερικά παραδείγματα συναρτήσεων συμμετοχής.



Το πρώτο σχήμα μπορεί κάλλιστα να είναι συνάρτηση συμμετοχής του συνόλου των νέων ανθρώπων και έχει μορφή:

$$\mu_A(x)=Y(x)=\begin{cases}1, & \alpha\nu\ \ x<25\\ \dfrac{40-x}{15}, & \alpha\nu\ \ 25<x<40\\ 0, & \alpha\nu\ \ 40<x\end{cases}$$

Ένας πιο μαθηματικός ορισμός του ασαφούς συνόλου είναι ο εξής:

$$A=\left\{(x,\mu_A(x))\middle| x\in X,\mu_A:X\to[0,1]\right\}$$

όπου το Χ λέγεται υπερσύνολο αναφοράς του Α και $\mu_A(x)$ αποτελεί τη συνάρτηση συμμετοχής των στοιχείων του Α. Με βάση τον παραπάνω ορισμό για τα ασαφή σύνολα ορίζεται ως κανονικό ένα ασαφές σύνολο Α όταν υπάρχει

ένα τουλάχιστον x του υπερσυνόλου αναφοράς X για το οποίο ισχύει $\mu_A(x)=1$. Μέτρο του A ορίζεται η ποσότητα $(A)=\Sigma\mu_A(x)$.

8.2. Πράξεις επί των ασαφών συνόλων

Ο Zadeh όρισε τις παρακάτω τρεις θεμελιώδεις πράξεις μεταξύ ασαφών συνόλων που όπως τονίσαμε είναι προέκταση των κλασικών συνόλων.

Έστω υπερσύνολο αναφοράς X και A και B ασαφή υποσύνολά του. Τότε ορίζεται ως:

i) Τομή των A και B το ασαφές σύνολο

$$C = A \cap B = \left\{ \left(x, \mu_D(x)\right) \middle| x \in X, \mu_c(x) = \min\left(\mu_A(x), \mu_B(x)\right) \right\}$$

ii) Ένωση των A και B το ασαφές σύνολο

$$C = A \cup B = \left\{ \left(x, \mu_D(x)\right) \middle| x \in X, \mu_D(x) = \max\left(\mu_A(x), \mu_B(x)\right) \right\}$$

iii) Συμπλήρωμα του A το ασαφές σύνολο

$$A' = \left\{ \left(x, \mu'_A(x)\right) \middle| x \in X, \mu'_A(x) = 1 - \mu_A(x) \right\}$$

Οι υπόλοιπες πράξεις που ορίζονται επί των ασαφών συνόλων είναι:

iv) Αλγεβρικό άθροισμα

$$A + B = \left\{ \left(x, \mu_{A+B}(x)\right) \middle| x \in X, \mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) + \mu_B(x) \right\}$$

v) Αλγεβρικό γινόμενο

$$A \cdot B = \left\{ \left(x, \mu_{AB}(x)\right) \middle| x \in X, \mu_{AB}(x) = \mu_A(x)\mu_B(x) \right\}$$

vi) Φραγμένο άθροισμα

$$A \oplus B = \left\{ \left(x, \mu_{A \oplus B}(x)\right) \middle| x \in X, \mu_{A \oplus B}(x) = \min\left(1, \mu_A(x) + \mu_B(x)\right) \right\}$$

vii) Φραγμένο γινόμενο

$$A \circ B = \left\{ \left(x, \mu_{A \circ B}(x)\right) \middle| x \in X, \mu_{A \circ B}(x) = \max\left(0, \mu_A(x) - \mu_B(x)\right) \right\}$$

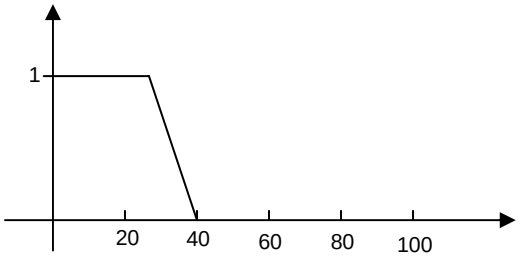
viii) Αριστερό τετράγωνο

$$^2A = \left\{ \left(x, \mu_{2A}(x) \right) \middle| x \in X, \mu_{2A}(x^2) = \mu_A(x) \right\}$$

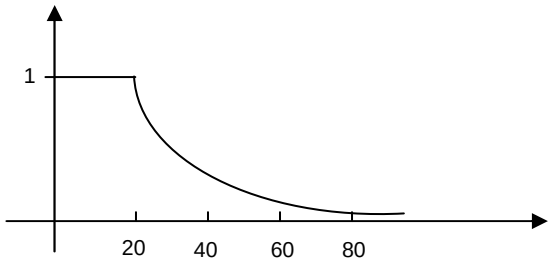
vvi) Κυρτός συνδυασμός

$$A \subset B = \left\{ \left(x, \mu_{A \subset B}(x) \right) \middle| x \in X, \mu_{A \subset B}(x) = w_1 \mu_A(x) + w_2 \mu_B(x), w_1 + w_2 = 1 \right\}$$

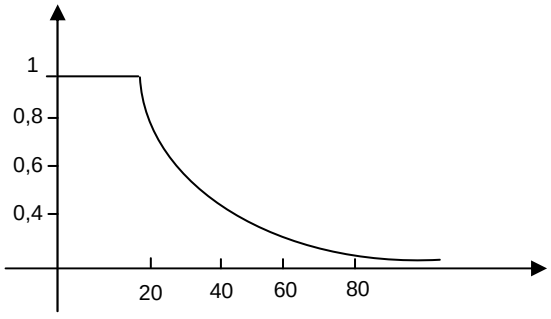
Παρακάτω δίνουμε πώς συνδυάζονται οι συναρτήσεις συμμετοχής με επίδραση αυτών των πράξεων.



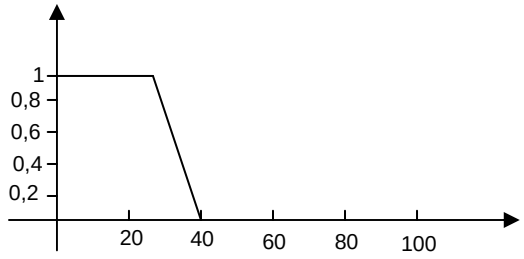
Συνάρτηση συμμετοχής A



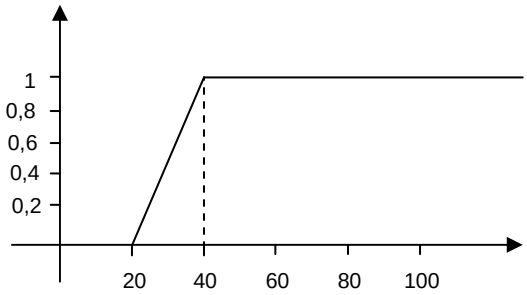
Συνάρτηση συμμετοχής B



Συνάρτηση συμμετοχής A∪B



Συνάρτηση συμμετοχής A∩B



Συνάρτηση συμμετοχής A′

Οι παραπάνω πράξεις μπορούν να γενικευθούν με τη βοήθεια T-τελεστών. Οι T-τελεστές διακρίνονται σε T-norms, T-conorms, N-negation συναρτήσεις. Αυτό είναι λογικό αφού και στην κλασική θεωρία συνόλων ορίζονται συναρτήσεις που αντιστοιχούν σημεία ενός συνόλου σε ένα άλλο σύνολο.

Με τον όρο T-norm εννοούμε μια συνάρτηση με δύο ορίσματα

$$T:[0,1]x[0,1]\rightarrow[0,1]$$

έτσι ώστε:

i) Για $x\leq y$, $w\leq z$, $xTw\leq yTz$ ή $T(x,w)\leq T(y,z)$

ii) Να είναι αντιμεταθετική

$XTy=yTx$ ή $T(x,y)=T(y,x)$

iii) Να είναι προσεταιριστική

$(xTy)Tz=xT(yTz)$ ή $T(x,T(y,z))=T(T(x,y),z)$

iv) Να ικανοποιεί το σύνολο των οριακών συνθηκών

$T(x,0)=0$, $T(x,1)=x$

Σε όλες τις παραπάνω σχέσεις ισχύουν $x,y,z,w\in[0,1]$

Οι πιο γνωστές T-norm συναρτήσεις είναι:

min(x,y): fuzzy intersection (ασαφής τομή)

xy : αλγεβρικό γινόμενο (algebraic product)

$x*y=\max(0,x+y-1)$: bounded product

$$\begin{cases} x & \alpha\nu \quad y=1 \\ y & \alpha\nu \quad x=1 \\ 0 & \alpha\nu \quad x,y<1 \end{cases}$$: drastic product

Αν ισχύουν κάποιοι άλλοι περιορισμοί διακρίνουμε μια συγκεκριμένη ομάδα T-norms όπως την Archimedean (Αρχιμήδεια) για την οποία ισχύει ότι:

i) είναι ασυνεχής με αναφορά το κάθε όρισμα

ii) $T(x,x)<x$ για κάθε $x\in[0,1]$

Η συνάρτηση $T^*:[0,1]x[0,1]\rightarrow[0,1]$ καλείται T-conorm αν και μόνον αν για κάθε $x,y,z\in[0,1]$ ισχύουν οι σχέσεις:

i) $T^*(x,y)=T^*(x,y)$ (αντιμεταθετική)

ii) $T^*(x,y)\leq T^*(x,z)$ $\forall y\leq z$

iii) $T^*(x,T^*(y,z))=T^*(T^*(x,y),z)$ (προσεταιριστική)

iv) $T^*(x,0)=0$

Οι πιο γνωστές T-conorm συναρτήσεις είναι:

max(x,y): ασαφής έναυση (fuzzy umon)

x+y-xy: αλγεβρικό άθροισμα (algebraic sum)

$x\pm y=\min(1,x+y)$: bounded sum (οριακό άθροισμα)

$$\begin{cases} x & \alpha\nu \quad y=0 \\ y & \alpha\nu \quad x=0 \\ 1 & \alpha\nu \quad x,y>0 \end{cases}$$: δραστικό άθροισμα (drastic sum)

Η συνάρτηση $N:[0,1]x[0,1]\rightarrow[0,1]$ καλείται negation function αν και μόνον αν $\forall$ $x,y,z\in[0,1]$ ισχύουν:

$N(0)=1$ $N(1)=0$

$N(x) \leq N(y) \quad \forall x \geq y$

Το θεώρημα της επέκτασης

Όπως στη θεωρία των κλασικών συνόλων αναπτύσσεται η θεωρία των συναρτήσεων με τις γνωστές έννοιες των πεδίων ορισμού, πεδίων τιμών, τύπος της συνάρτησης κλπ. έτσι και στη θεωρία των ασαφών συνόλων μπορούν να οριστούν και να αναπτυχθούν αντίστοιχες έννοιες.

Έστω η κλασική συνάρτηση:

$f: X_1xX_2x\ldots xX_n \rightarrow Y, \quad y=f(x_1,x_2,\ldots,x_n)$

και $A_1,A_2,\ldots,A_n$ ασαφή σύνολα οριζόμενα στα υπερσύνολα αναφοράς $X_,X_2,\ldots,X_n$ αντίστοιχα. Τότε συνδυάζοντας την έννοια της εικόνας ασαφούς συνόλου με την έννοια του καρτεσιανού γινομένου ασαφών συνόλων ορίζουμε ένα ασαφές σύνολο $f(A_1,A_2,\ldots,A_n)$ στο Y ως:

$$\mu_{f(A_1,\ldots,A_n)}(y) = \sup_{y=f(x_1,\ldots,x_n)} \mu_{A_1xA_2\ldots xA_n}(x_1,x_2,\ldots,x_n) =$$

$$= \sup_{y=f(x_1,\ldots,x_n)} \min\{\mu_{A_1}(x_1),\mu_{A_1},\ldots,\mu_{A_n}(x_n)$$

όπου το άνω φράγμα (suprerum) λαμβάνεται για όλα τα $(x_1,x_2,\ldots,x_n) \in X_1,X_2,\ldots,X_n$ που δίνουν την ίδια τιμή $y=f(x_1,x_2,\ldots,x_n)$.

Η παραπάνω σχέση εκφράζει την αρχή της ασαφούς επέκτασης η οποία επιτρέπει την μεταβίβαση της ασάφειας των $A_1,A_2,\ldots,A_n$ στο σύνολο ασαφούς εικόνας: $B=f(A_1,A_2,\ldots,A_n) \in F(Y)$.

Τα πεδία ορισμού και τιμών των ασαφών συναρτήσεων μπορούν να βρεθούν ως εξής:

Έστω X,Y δύο κλασικά σύνολα και $f:X \rightarrow Y$ κλασική συνάρτηση. Ορίζεται ως ασαφές δυναμοσύνολο του X το σύνολο όλων των δυνατών ασαφών συνόλων επί του X και συμβολίζεται με F(X), ενώ ως ασαφές δυναμοσύνολο του Y ορίζεται το σύνολο όλων των δυνατών ασαφών υποσυνόλων επί του Y και συμβολίζεται με F(Y). Με τη βοήθεια των παραπάνω ορισμών και την κλασική συνάρτηση $f:X \rightarrow Y$ ορίζονται τα ασαφή πεδία ορισμού και ασαφή πεδία τιμών ως εξής:

i) Για το πεδίο τιμών: Έστω τυχαίο y τότε αν δεν ορίζεται το $x=f^{-1}(y)$ ή ισχύει $f^{-1}(x)=\varnothing$ τότε $\mu_y(y)=0$.

Αν υπάρχει το $x=f^{-1}(y)$ τότε μεταξύ των y που δίνουν το ίδιο x επιλέγεται εκείνο του οποίου το αντίστοιχο x κάνει την ποσότητα $\mu_x(x)$ μέγιστη και τίθεται η ποσότητα αυτή ως $\mu_y(y)$.

ii) Πεδίο ορισμού: Έστω $Y \in F(y)$. Τότε $f^{-1}(Y)=X \in F(x)$. Για κάθε $y=f(x)$ αρκεί να βρεθεί το $\mu_y(f(x))$ και τότε $\mu_{f^{-1}(y)}(x) = \mu_y(f(x))$.

Τέλος πρέπει να πούμε ότι το θεώρημα της επέκτασης είναι το βασικότερο εργαλείο στη θεωρία των ασαφών συνόλων γιατί βοηθά στην ασαφοποίηση, γιατί επιτρέπει σε μια κλασική συνάρτηση f, με πεδίο ορισμού έστω το κλασικό σύνολο X, να αλλάξει το πεδίο ορισμού της και να δεχθεί ως πεδίο ορισμού ασαφή υποσύνολα του X.

Γλωσσικές μεταβλητές και γλωσσικός διαμορφωτής

Όπως έχουμε ήδη αναφέρει η ασαφής λογική χειρίζεται ασαφώς ορισμένες ποσότητες. Στο επόμενο κεφάλαιο (Ασαφή συστήματα) θα διαπιστώσουμε ότι στα συστήματα χρησιμοποιούμε γνώση η οποία προέρχεται από έναν ειδικό γνώστη του χειρισμού ενός συστήματος. Αυτή η γνώση είναι ασαφώς ορισμένη π.χ. «Αν η ταχύτητα είναι μεγάλη τότε πάτα πολύ το φρένο».

IF $U \in U_2$ THEN $y \in Y_2$. Στην παραπάνω πρόταση η ταχύτητα (ουσιαστικό) προσδιορίζεται από ένα επίθετο (μεγάλη). Η ταχύτητα είναι η γλωσσική μεταβλητή και το σύνολο των επιθέτων: μικρή, μεσαία, μεγάλη αποτελεί το σύνολο τιμών της (που είναι ασαφή σύνολα) γλωσσικής μεταβλητής. Άλλο ένα παράδειγμα γλωσσικών μεταβλητών (linguistic variables) είναι η ηλικία που έχει ως πεδίο τιμών της το: {μωρό, παιδί, έφηβος, ώριμος, μεσήλικας, γέρος} ή καλύτερα το: {«πολύ νέος», «αρκετά νέος», «λίγο ηλικιωμένος», «αρκετά ηλικιωμένος», «πολύ ηλικιωμένος»}.

Έτσι φτάνουμε στον εξής ορισμό της γλωσσικής μεταβλητής. Ως γλωσσική μεταβλητή νοείται μια μεταβλητή που οι τιμές της δεν είναι αριθμοί αλλά λέξεις ή φράσεις σε μια φυσική ή τεχνητή γλώσσα.

Ένας πιο αυστηρός μαθηματικός ορισμός είναι ο εξής:

Γλωσσική μεταβλητή είναι η πεντάδα

<x,T(x),U,G,M> όπου:

x: το όνομα της γλωσσικής μεταβλητής

T(x): το σύνολο των τιμών της (ασαφές σύνολο)

U: το υπερσύνολο αναφοράς πάνω στο οποίο δομείται το σύνολο τιμών T(x).

G: ένας συντακτικός κανόνας που παράγει τα ονόματα των τιμών της x, δηλαδή, τα ονόματα των ασαφών συνόλων.

M: ένας σημασιολογικός κανόνας που αποδίδει νόημα στα ονόματα.

Όπως και στην κλασική θεωρία συνόλων υπάρχουν συναρτήσεις οι οποίες τροποποιούν την αριθμητική ποσότητα αντιστοιχίζοντάς την σε ένα αριθμό (πεδίο τιμών) έτσι και στην Ασαφή λογική υπάρχουν τελεστές που δρουν πάνω στις τιμές των γλωσσικών μεταβλητών μεταβάλλοντάς τες. Οι τελεστές αυτοί είναι ο γλωσσικός διαμορφωτής ο οποίος εφαρμοζόμενος σε μια τιμή μιας γλωσσικής μεταβλητής που είναι ένα ασαφές σύνολο έχει ως αποτέλεσμα την αλλαγή του νοηματικού της περιεχομένου. Αν A είναι ένα ασαφές σύνολο και εφαρμοστεί ένας γλωσσικός διαμορφωτής m πάνω σ' αυτό το αποτέλεσμα είναι να προκύψει ένα άλλο τροποποιημένο ασαφές σύνολο B=m(A). Οι κυριότεροι γλωσσικοί διαμορφωτές (linguistic modifier) είναι οι ακόλουθοι:

- πολύ (A) : $\mu_{\text{πολύ(A)}}(x) = (\mu_A(x))^2$ Συγκεντρωτής
- σχεδόν (A) : $\mu_{\text{σχεδόν(A)}}(x) = (\mu_A(x))^{\frac{1}{2}}$ Εξαπλωτής
- συν (A) : $\mu_{\text{συν(A)}}(x) = (\mu_A(x))^{\frac{5}{4}}$ [Συν]
- μείον (A) : $\mu_{\text{μείον(A)}}(x) = (\mu_A(x))^{\frac{3}{4}}$ [Μείον]
- όχι (A) : $\mu_{\text{όχι(A)}}(x) = 1 - \mu_A(x)$ ['Αρνηση]
- μέτρο (A) : $\mu_{\text{μέτρο(A)}}(x) = \mu_A(x) / \max \mu_A(x)$ [κανονικοποίηση]

Γενικευμένοι κανόνες του Θέτειν και του Αναιρείν (Modus Poneus and Modus Tolleus)

Στη κλασική ανθρώπινη λογική τα συμπεράσματα στηρίζονται σε κλασικές λογικές ταυτολογίες απ' τις οποίες οι τέσσερις πιο γνωστές είναι:

modus poneus: $\{A \wedge (A \to B)\} \to A$

modus tolleus: $\{(A \to B) \wedge -B\} \to A$

Συλλογισμός: $\{(A \to B) \wedge (B \to C)\} \to (A \to C)$

Αντίθετο αναστροφή: $(A \to B) \to (-B \to -A)$

Από τις παραπάνω λογικές ταυτολογίες ο κανόνας modus poneus βρίσκει τη μεγαλύτερη εφαρμογή. Στη γενική του μορφή γράφεται:

Συνεπαγωγή: ΕΑΝ Α ΤΟΤΕ Β

ΓΕΓΟΝΟΣ: Α

Συμπέρασμα: Β

Ο παραπάνω κανόνας (modus poneus) έχει ένα μικρό πρόβλημα να εφαρμοστεί αυτούσιος όπως είναι σε ασαφές περιβάλλον. Έτσι πρέπει να υποστεί μια μικρή τροποποίηση για να μπορεί να βγάζει συμπεράσματα μέσα από ασαφή σύνολα. Έτσι προκύπτει ο κανόνας που λέγεται general modus poneus (G.M.P.) που έχει την εξής μορφή:

Συνεπαγωγή: ΕΑΝ x=a ΤΟΤΕ y=B

Γεγονός: x=A′

Συμπέρασμα: y=B′

όπου A, A′, B, B′ είναι ασαφή σύνολα και x,y είναι γλωσσικές μεταβλητές. Συνεπώς στον GMP δεν είναι απαραίτητο το γεγονός να ταυτίζεται απόλυτα με την υπόθεση της συνεπαγωγής όπως ισχύει στον M.P. (modus poneus). Τα σύμβολα A′, B′ σημαίνουν ότι το x παίρνει τιμή (γλωσσική) λίγο τροποποιημένη από την x=A με αποτέλεσμα και η y να παίρνει τιμή λίγο διαφορετική από την y=B δηλαδή y= B′. Ο GMP μιμείται πολύ περισσότερο τον ανθρώπινο τρόπο σκέψης γιατί βγάζει συμπεράσματα σε συνθήκες ομοιότητας όπως ο άνθρωπος χωρίς να περιμένει να υπάρξει απόλυτη ταύτιση.

Ο κανόνας ασαφούς σύνθεσης (max-min)

Όπως φάνηκε από όλα τα προηγούμενα στην περίπτωση των ασαφών συνόλων χρειάζεται κάποιος καινούργιος κανόνας ο οποίος θα μπορεί να βγάζει συμπεράσματα κάτω από συνθήκες ομοιότητας. Έναν τέτοιο κανόνα πρότεινε ο

Zadeh ο οποίος παρέχει έναν πολύ χρήσιμο τρόπο εξαγωγής συμπερασμάτων μέσα σε αβεβαιότητα. Ο κανόνας αυτός που καλείται max-min composition ή compositional rule of inference ορίζεται ως εξής:

Έστω A ασαφές σύνολο του υπερσυνόλου X, B ασαφές σύνολο του υπερσυνόλου αναφοράς Y και P ασαφής σχέση επί του καρτεσιανού γινόμενου XxY, ισχύουν δηλαδή:

$$A = \{(x, \mu_A(x)) | x \in X, \mu_A(x) : A \to [0,1]\}$$

$$B = \{(y, \mu_B(y)) | y \in Y, \mu_B(y) : B \to [0,1]\}$$

$$R = \{((x,y), \mu_R(x,y)) | (x,y) \in XxY, \mu_R((x,y)) = \min(\mu_A(x), \mu_B(y))\}$$

Με δεδομένα τα A και R, το B υπολογίζεται σύμφωνα με τη σχέση:

$$\mu_B(y) = \max_x \{\min\{\mu_A(x), \mu_R(x,y)\}\}$$

και συμβολίζεται με B=AoR όπου «o» είναι ο τελεστής max-min composition.

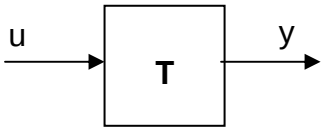
Ασαφή συστήματα

Εισαγωγή

Γενικά τα συστήματα μπορούν να θεωρηθούν σαν τμήματα του χώρου που επικοινωνούν με τον περιβάλλοντα χώρο με σήματα δηλαδή δέχονται σήματα σαν αίτια και παράγουν σήματα σαν αποτελέσματα. Η επικοινωνία του συστήματος με τον περιβάλλοντα χώρο γίνεται με τη βοήθεια των ακροδεκτών του συστήματος. Κάθε ζεύγος ακροδεκτών καλείται μία θύρα του συστήματος. Πρέπει να σημειώσουμε ότι τα σήματα που δέχεται ένα σύστημα καλούνται διεγέρσεις ή είσοδοι (inputs) ενώ τα σήματα που παράγει καλούνται αποκρίσεις ή έξοδοι (outputs).

Ένας πιο αυστηρός μαθηματικός ορισμός του συστήματος είναι ο εξής: Σύστημα καλείται ο μετασχηματισμός (ή ο τελεστής) που απεικονίζει ένα διάνυσμα διεγέρσεων u(t) σε ένα διάνυσμα αποκρίσεων y(t). Στο παρακάτω σχήμα παριστάνουμε ένα σύστημα το οποίο συμβολίζεται:

$$y(t)=T[u(t)]$$



Η μέχρι τώρα (πριν την ανάπτυξη της ασαφούς λογικής απ' τον L. Zadeh) προσέγγιση και ανάλυση των συστημάτων έχει γίνει με την παραδοχή ότι οι είσοδοι και οι έξοδοι ενός συστήματος είναι ποσότητες που μετρούνται αριθμητικά και είναι σαφώς ορισμένες. Όταν όμως στο όλο σύστημα προσμετράται και η υποκειμενική άποψη ενός ειδικού (ανθρώπου) ο οποίος εκφράζεται με επίθετα (πολύ, λίγο, αρκετά) τότε το νέο σύστημα πρέπει να επεξεργαστεί ασαφή σύνολα. Πιο συγκεκριμένα ο ειδικός (expert) δίνει στο ασαφές σύστημα ένα σύνολο ασαφώς ορισμένων κανόνων οι οποίοι χειρίζονται γλωσσικές μεταβλητές (linguistic variables): τα υποκειμενικά ασαφώς ορισμένα επίθετα (πολύ, λίγο κλπ.) σαν ασαφή σύνολα (fuzzy sets). Η βάση ασαφών κανόνων είναι ένα σύνολο κανόνων που έχουν την εξής μορφή:

$R^{(f)}$: If x_1 is F_1^f and ... and x_n is F_n^f

THEN y is $G^{(f)}$

όπου F_i^f και G_i^f είναι ασαφή σύνολα και x,y γλωσσικές μεταβλητές ενώ l=1,2,…,M όπου M το πλήθος των κανόνων βάσης.

Ένα πολύ γενικό ασαφές σύστημα είναι: «ο άνθρωπος που οδηγεί ένα αυτοκίνητο». Η ασαφής συλλογιστική του ανθρώπου χρησιμοποιείται για την οδήγηση του σαφώς ορισμένου αυτοκινήτου.

Ορισμός και ταξινόμηση ασαφών συστημάτων

Όπως είναι γνωστό απ' τη θεωρία συστημάτων τα συστήματα διακρίνονται σε Ντετερμινιστικά αν η παρούσα κατάσταση (που περιγράφεται απ' τις εξισώσεις κατάστασης οι οποίες με τη σειρά τους χρησιμοποιούν μεταβλητές κατάστασης) και είσοδος καθορίζουν κατά τρόπο επακριβή την επόμενη κατάσταση και την έξοδο και σε Στοχαστικά αν η επόμενη κατάσταση είναι η έξοδος (δεδομένης της παρούσας κατάστασης και της εισόδου) δεν

μπορούν να περιγραφούν επακριβώς αλλά απεναντίας μπορούν να περιγραφούν πιθανοτικές και στατιστικές τους ιδιότητες.

Ασαφές καλείται ένα σύστημα αν για την επόμενη κατάσταση και την έξοδο (δεδομένης της παρούσας κατάστασης και της εισόδου) το μόνο που μπορεί να λεχθεί είναι πως αποτελούν στοιχεία ενός συνόλου, το οποίο έχει επιπλέον και ασαφή όρια.

Τα ασαφή συστήματα ταξινομούνται σε τρεις κατηγορίες:

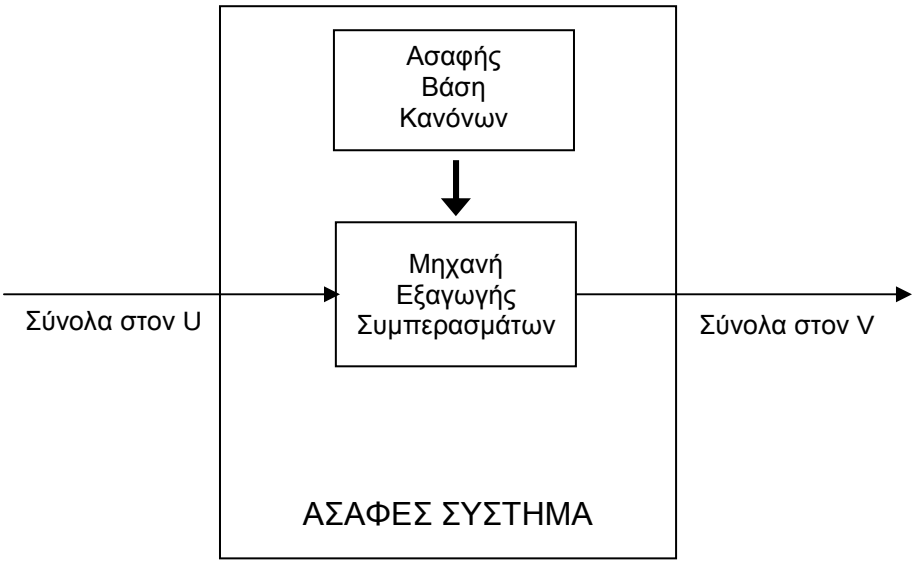
i) τα καθαρά ασαφή συστήματα

ii) τα ασαφή συστήματα Takagi και Sugeno

iii) τα ασαφή συστήματα με ασαφοποιητή και αποασαφοποιητή.

Καθαρά ασαφή συστήματα

Στο παρακάτω σχήμα φαίνεται η δομή ενός καθαρού ασαφούς συστήματος το οποίο αποτελείται από μια ασαφή βάση κανόνων, τη μηχανή εξαγωγής ασαφών συμπερασμάτων και δέχεται ως είσοδο ασαφή σύνολα.



Η μηχανή εξαγωγής ασαφών συμπερασμάτων χρησιμοποιεί τους κανόνες της ασαφούς βάσης κανόνων που έχουν τη μορφή IF-THEN για να αντιστοιχήσει τα ασαφή σύνολα της εισόδου του ασαφούς συστήματος σε ασαφή σύνολα της εξόδου του. Για να γίνει αυτό εφαρμόζει την πιο συνηθισμένη

αρχή της θεωρίας Ασαφών Συνόλων τον κανόνα max-min. Ειδικότερα, έστω A′ ένα οποιαδήποτε ασαφές σύνολο του U το οποίο αποτελεί την είσοδο ασαφούς συστήματος. Η έξοδος του ασαφούς συστήματος ορίζεται με τη βοήθεια κάθε κανόνα της βάσης σαν ένα ασαφές σύνολο $A'oR^l$ στον V η συνάρτηση συμμετοχής του οποίου είναι:

$$\mu_{A'oR^l}(y) = \max_x \in u[\min \mu_A(x), \mu_{F1x...Fn} \rightarrow Gl(x,y)]$$

Η τελική έξοδος του καθαρού ασαφούς συστήματος είναι ένα ασαφές σύνολο $A'o(R^{(1)},\ldots,R^{(M)})$ στο V το οποίο είναι συνδυασμός των M ασαφών συνόλων που προέκυψαν με τον παραπάνω τρόπο.

Ο συνδυασμός αυτός είναι:

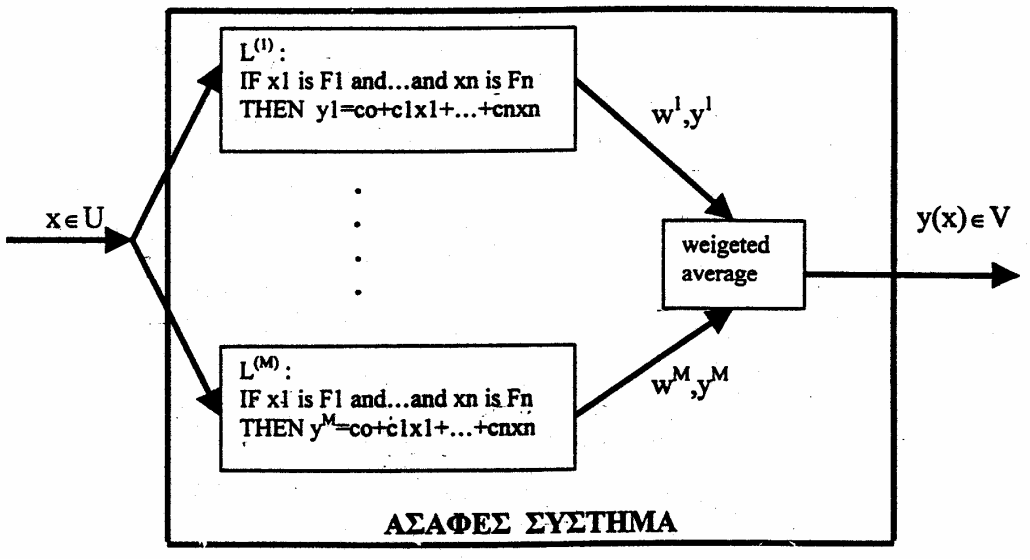
$$\mu_{Ao(R(1),\ldots,R(M))}(y) = \mu_{A'oR(1)}(y) \pm \ldots \pm \mu_{A'oR(M)}(y)$$

όπου ο τελεστής ± μπορεί να είναι ο τελεστής max, το αλγεβρικό άθροισμα ή κάποιος άλλος τελεστής. Αν χρησιμοποιήσουμε και ανάδραση τότε παίρνουμε τα ασαφή δυναμικά συστήματα στα οποία η έξοδος εξαρτάται από την είσοδο.

Παρατηρούμε ότι στα καθαρά ασαφή συστήματα η είσοδος και η έξοδός τους είναι ασαφή σύνολα ενώ στις πρακτικές εφαρμογές οι τιμές που πρέπει να χειριστεί ένα σύστημα είναι πραγματικές τιμές.

Τα ασαφή συστήματα Takagi – Sugeno

Οι Takagi και Sugeno πρότειναν την παρακάτω δομή ασαφούς συστήματος.



Η κυριότερη διαφορά από την προηγούμενη αρχιτεκτονική ασαφούς συστήματος είναι στη μορφή των κανόνων. Ένας κανόνας εδώ έχει την εξής μορφή:

$L^{(1)}$: IF x1 is F1 and… and xn is Fn THEN yl=co+c1x1+…+cnxn

όπου Fi είναι ασαφή σύνολα, ci είναι σταθεροί πραγματικοί αριθμοί, yl είναι η έξοδος του συστήματος που αντιστοιχεί στον κανόνα, ενώ υπάρχουν 1=1,2,…,M στο πλήθος κανόνες. Παρατηρώντας τη μορφή των κανόνων διακρίνεται ότι οι είσοδοι στο ασαφές σύνολο είναι πραγματικές μεταβλητές, τα IF μέρη των κανόνων είναι ασαφή σύνολα, ενώ η έξοδος των κανόνων είναι σταθεροί πραγματικοί αριθμοί και αποτελούν μάλιστα γραμμικό συνδυασμό των μεταβλητών της εισόδου. Η έξοδος του ασαφούς συστήματος είναι πραγματική τιμή και αποτελεί τον γραμμικό συνδυασμό των εξόδων των κανόνων πολλαπλασιασμένων με κατάλληλα βάρη σύμφωνα με την εξίσωση:

$$y(\underline{x}) = \sum_{il=1}^{M} w^1 y^1 / \sum_{i=1}^{M} w^1$$

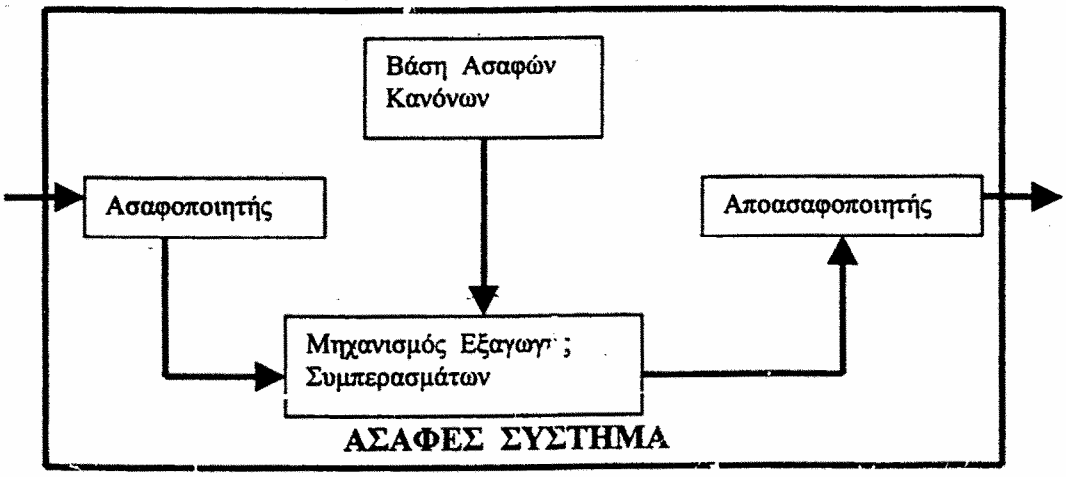
όπου τα βάρη w^1 υπολογίζονται για κάθε κανόνα σύμφωνα με τη σχέση:

$$w^1 = \prod_{i=1}^{n} \mu_{Fi}^1 (x_I)$$

και δίνουν την ολική αλήθεια του κανόνα για την είσοδο.

Το πλεονέκτημα της παραπάνω αρχιτεκτονικής ασαφούς συστήματος είναι ότι δέχεται ως εισόδους πραγματικές μεταβλητές και δίνει έξοδο επίσης πραγματική μεταβλητή πράγμα που δίνει τη δυνατότητα να εφαρμοστεί σε πρακτικά προβλήματα. Επίσης όπως φάνηκε παραπάνω μπορεί να υπάρξει ένα μαθηματικό μοντέλο που να μπορεί να περιγράψει το ασαφές σύστημα πράγμα επιθυμητό στην εκτίμηση των παραμέτρων του συστήματος και της τάξης του. Το αδύνατο σημείο της παραπάνω περιγραφής είναι το τμήμα των κανόνων δεν είναι ασαφές αποκλείοντας έτσι την ενσωμάτωση της γλωσσικής πληροφορίας στο τμήμα της εξόδου του συστήματος.

Τέλος, το ευρύτερα χρησιμοποιούμενο και περισσότερο δημοφιλές ασαφές σύστημα είναι αυτό με ασαφοποιητή και αποασαφοποιητή. Η δομή ενός τέτοιου συστήματος φαίνεται στο παρακάτω σχήμα:



Η δομή αυτού του συστήματος είναι όμοια με αυτή του καθαρού ασαφούς συστήματος μόνο που έχει έναν ασαφοποιητή στην είσοδο ο οποίος αντιστοιχεί τις πραγματικές τιμές της εισόδου σε ασαφή σύνολα και έναν αποασαφοποιητή στην έξοδο ο οποίος αντιστοιχεί ασαφή σύνολα σε πραγματικές εξόδους στο σύστημα. Έτσι αποκτάται η δυνατότητα χρήσης του ασαφούς συστήματος σε πραγματικές εφαρμογές. Η παραπάνω δομή έχει προταθεί από τον Mamadani και έχει χρησιμοποιηθεί σε πληθώρα εφαρμογών ως ελεγκτής γι' αυτό είναι γνωστή και ως ασαφής ελεγκτής. Τα πλεονεκτήματα του συστήματος αυτού είναι πολλά γιατί δέχεται ως εισόδους πραγματικές τιμές και παράγει εξόδους επίσης πραγματικές, υπάρχει ποικιλία επιλογών όσον αφορά τον ασαφοποιητή, το μηχανισμό εξαγωγής συμπερασμάτων, και τον αποασαφοποιητή. Επίσης όπως θα φανεί παρακάτω μπορεί να βρεθεί μαθηματική περιγραφή του συστήματος πράγμα που διευκολύνει τη δημιουργία ενός προσαρμοστικού ασαφούς συστήματος (ασαφούς νευρωνικού δικτύου) και στην ανάπτυξη αλγόριθμων εκπαίδευσης αυτού. Η κατασκευή των επιμέρους τμημάτων του ασαφούς συστήματος με ασαφοποιητή και αποασαφοποιητή στηρίζεται στη θεωρία των ασαφών συνόλων που αναπτύχθηκε στα προηγούμενα.

Ασαφοποιητής: Ο ασαφοποιητής πραγματοποιεί την αντιστοίχιση από μια πραγματική μεταβλητή $\underline{x} = (x_1, \ldots, x_n)^T \in U$ σε ένα ασαφές σύνολο A′ του U.

Υπάρχουν τουλάχιστον δύο δυνατές επιλογές ως προς τη δομή του ασαφοποιητή.

α) *Ο μονότιμος ασαφοποιητής.* Ο ασαφοποιητής αυτός ορίζεται από την εξής σχέση:

$$\mu_{A'}(\underline{x}') = \begin{cases} 1 & \text{αν } \underline{x}' = \underline{x} \\ 0 & \text{αν } \underline{x}' \neq x \end{cases}$$

β) *Ο μη μονότιμος ασαφοποιητής.* Παίρνει μέγιστη τιμή $\mu_{A'}(\underline{x}) = 1$ σε κάποια θέση $\underline{x}$ και μειώνεται καθώς απομακρύνεται από αυτή την τιμή σύμφωνα με τη σχέση:

$$\mu_{A'}(\underline{x}') = \exp\left[-(\underline{x}' - \underline{x})^T (\underline{x}' - x)/\sigma^2\right]$$

όπου σ^2 είναι παράμετρος που χαρακτηρίζει τη μορφή της καμπύλης της συνάρτησης συμμετοχής.

Από τους παραπάνω ασαφοποιητές χρησιμοποιείται ο μονότιμος περισσότερο ενώ ο μη μονότιμος βρίσκει εφαρμογή σε θορυβώδες περιβάλλον.

Αποασαφοποιητής: Ο αποασαφοποιητής αντιστοιχεί ασαφή σύνολα στο V σε πραγματικές τιμές επίσης στο V. Οι πιο γνωστοί αποασαφοποιητές είναι:

α) *Ο αποασαφοποιητής μεγίστου:* Ο αποασαφοποιητής αυτός ορίζεται από την παρακάτω εξίσωση:

$$y = \arg\left(\max_y \in v(\mu_{B'}(y))\right)$$

όπου $\mu_{B'}$ δίνεται από τη σχέση $\mu_{B'}(y)=\mu_B^1(y)\pm\ldots\pm\mu_B^M(y)$

β) *Ο αποασαφοποιητής μέσου όρου:* Η εξίσωση ορισμού του ασαφοποιητή αυτού είναι:

$$y = \sum_{i=1}^{M} y^1\left(\mu_{B1}(y^1)\right) / \sum_{il=1}^{M}\left(\mu_{B1}(y^1)\right)$$

όπου y^1 είναι το κέντρο του ασαφούς συνόλου G^1 και $\mu_B^1(y)$ προκύπτει από την εφαρμογή του κανόνα max-min μεταξύ του γινόμενου των κανόνων και της εισόδου.

γ) *Ο τροποποιημένος αποασαφοποιητής μέσου όρου:* Ορίζεται με βάση τη σχέση:

$$y = \sum_{l=1}^{M} y^1\left(\mu_{B1}\left(y^1\right)/\delta^1\right) / \sum_{l=1}^{M} y^1\left(\mu_{B1}\left(y^1\right)/\delta^1\right)$$

όπου δ^1 είναι παράμετρος που χαρακτηρίζει τη μορφή των συναρτήσεων συμμετοχής.

Ασαφής Βάση Κανόνων. Η ασαφής βάση κανόνων αποτελείται από ένα σύνολο κανόνων της μορφής:

$$R^{(1)}\text{: IF } x_1 \text{ is } F_1 \text{ and } \ldots \text{ and } x_n \text{ is } F_n \text{ THEN y is } G^1$$

όπου F_i^1 και G^1 είναι ασαφή σύνολα στο $U_i \subset R$ και $V \subset R$ αντίστοιχα, $\underline{x} = \left(x_1, \ldots, x_n\right)^T \in U_1 x \ldots x U_n$ και $y \in V$ είναι γλωσσικές μεταβλητές M είναι το σύνολο κανόνων της παραπάνω μορφής που περιέχει η βάση. Τα $\underline{x}$ και y είναι η είσοδος και η έξοδος του συστήματος που θεωρείται σύστημα πολλών εισόδων μιας εξόδου μια και έχει αποδειχθεί ότι κάθε σύστημα πολλών εισόδων – πολλών εξόδων μπορεί να αποσυζευχθεί σε ένα σύνολο συστημάτων μιας εξόδου. Η βάση κανόνων αποτελεί την καρδιά του ασαφούς συστήματος μια και εκεί υπάρχει αποθηκευμένη η γλωσσική πληροφορία για τη λειτουργία του. Οι κανόνες αυτοί μπορεί να αποκτηθούν ρωτώντας έναν έμπειρο άνθρωπο ή εκπαιδεύοντας το σύστημα σε επιθυμητά ζεύγη δεδομένων εισόδου και εξόδου.

Μηχανή Εξαγωγής Συμπερασμάτων. Στο τμήμα αυτό του ασαφούς συστήματος συνδυάζονται οι παραπάνω κανόνες που βρίσκονται στη βάση και παράγεται ένα ασαφές σύνολο εξόδου. Ο κάθε κανόνας της βάσης ερμηνεύεται σαν μια ασαφής σχέση $F_1^1 x \ldots x F_n^1 \rightarrow G^1$ στο UxV. Αν ένα ασαφές σύνολο A′ στο U είναι η είσοδος στη μηχανή εξαγωγής συμπερασμάτων τότε κάθε κανόνας καθορίζει ένα ασαφές σύνολο B^1 στο V σύμφωνα με τον κανόνα max-min ως εξής:

$$\mu_{B'}(y) = \max_{\underline{x}} \in U\left[\min\left(\mu_{F1}{}^1 x \ldots x F_n{}^1\right) \rightarrow G^1\left(\underline{x}, y\right), \mu_{A'}(\underline{x})\right]$$

Η ασαφής σχέση που υπάρχει στην παραπάνω εξίσωση μπορεί να έχει πολλές μορφές από τις οποίες η πιο γνωστές και αυτές που χρησιμοποιηθούν εδώ είναι:

$$\mu_{A \rightarrow B}(x, y) = \min\left\{\mu_A(x), \mu_B(y)\right\}$$
$$\mu_A(x0 = \min\left\{\mu_{F1}\left(x_1\right), \ldots, \mu_{Fn}(x)^n\right\}$$

$$\mu_{A\to B}(x,y)=\mu_A(\underline{x})\mu_B(y)$$
$$\mu_A(\underline{x})=\mu_{F1}{}^1(x_1)\ldots\mu_{Fn}{}^1(x_n)$$

όπου $A=F_l{}^1x\ldots xF_n{}^1$ και $B=G^1$.

Έχει αποδειχθεί ότι ένα ασαφές σύστημα της παραπάνω μορφής με μονότιμο ασαφοποιητή, αποασαφοποιητή μέσου όρου, και μηχανισμό εξαγωγής συμπεράσματος που βασίζεται στις εξισώσεις μπορεί να περιγραφεί με την εξής μαθητή εξίσωση:

$$f(\underline{x})=\sum_{l=1}^{M} y^1\left(\prod_{l=1}^{n}\mu_{Fi}{}^1(x_I)\right)/\sum_{l=1}^{M}\left(\prod_{l=1}^{n}\mu_{Fi}{}^1(x_I)\right)$$

όπου y^1 είναι το σημείο στο οποίο η $\mu_G{}^1$ παίρνει τη μέγιστη τιμή της, η οποία θεωρείται ότι είναι $\mu_G{}^1(y^1)=1$. Η εξίσωση αυτή χρησιμοποιείται στην κατασκευή νευρωνικού δικτύου, τα λεγόμενα ασαφή νευρωνικά δίκτυα, το οποίο δίνει τη δυνατότητα στο ασαφές σύστημα να προσαρμόζει τη βάση κανόνων έτσι ώστε να αποκτά μόνο του τη γλωσσική πληροφορία.

Τα ασαφή συστήματα μπορούν να περιγράφουν με έναν τρόπο αντίστοιχο των εξισώσεων κατάστασης με αυτό των κλασικών συστημάτων. Οι εξισώσεις αυτές είναι γνωστές ως ασαφείς εξισώσεις κατάστασης. Μ' αυτό τον τρόπο η θεωρία των Ασαφών Συνόλων δανείζεται έννοιες από τη Μοντέρνα Θεωρία Αυτόματου Ελέγχου κερδίζοντας τα πλεονεκτήματα που παρέχει η παραπάνω περιγραφή.

Έστω λοιπόν U, V και W διακριτά υπερσύνολα αναφοράς για τις ασαφείς μεταβλητές κατάστασης ελέγχου και εξόδου αντίστοιχα τα οποία θεωρούνται πεπερασμένα. Ισχύει δηλαδή X={x1,x2,…,xn} ασαφές σύνολο κατάστασης, U={u1,u2,…,um} ασαφές σύνολο ελέγχου και Y={y1,y2,…,yv} ασαφές σύνολο εξόδου.

Ένα ασαφές δυναμικό σύστημα στο χώρο κατάστασης περιγράφεται με τη βοήθεια των ασαφών σχεσιακών εξισώσεων ως εξής:

$$X_{k+p} = U_k \circ X_k \circ X_{k+1} \circ X_{k+2} \circ \ldots \circ X_{k+p-1} \circ R$$
$$Y_{k+p} = X_{k+p} \circ S$$

Όπου U_k ασαφής μεταβλητή ελέγχου τη χρονική στιγμή k, $X_{k+p}, X_{k+p-1}, \ldots, X_k$ ασαφής μεταβλητή κατάστασης τις χρονικές στιγμές k+p,k+p-1,…,k, Y_k ασαφής

μεταβλητή εξόδου τη χρονική στιγμή k+p, R,S ασαφείς σχέσεις που έχουν να κάνουν με τη δυναμική του συστήματος και ορίζονται ως εξής:

R: U x Xx … xX ⟶ [0,1]

S: X x Y ⟶ [0,1]

Και ο τελεστής max-min

Το ζευγάρι των εξισώσεων κατάστασης έχει την ίδια μορφή με τις εξισώσεις κατάστασης των κλασικών συστημάτων. Η πρώτη εξίσωση αποτελεί την εξίσωση κατάστασης που συνδέει την ασαφή κατάσταση του συστήματος τη χρονική στιγμή k+p με τις ασαφείς καταστάσεις των χρονικών στιγμών k,…,k+p-1 και τον ασαφή έλεγχο τη χρονική στιγμή k. Η δεύτερη εξίσωση αποτελεί την εξίσωση εξόδου του συστήματος και συνδέει την ασαφή έξοδο τη χρονική στιγμή με την ασαφή κατάσταση την ίδια χρονική στιγμή. Τα ασαφή συστήματα διακρίνονται σε ασαφή συστήματα πρώτης τάξης καθώς και σε ανώτερης τάξης τα οποία μπορούν να περιγράψουν με τη βοήθεια των συστημάτων πρώτης τάξης.

Ένα ασαφές σύστημα πρώτης τάξης περιγράφεται με την εξής ασαφή σχεσιακή εξίσωση της μορφής:

$$X_{k+1} = U_k \circ X_k \circ R$$

όπου R η σχέση του ασαφούς συστήματος X_k η ασαφής κατάσταση τη χρονική στιγμή k,X_{k+1} η ασαφής κατάσταση τη χρονική στιγμή k+1 και U_k η ασαφής είσοδος του συστήματος. Η παραπάνω εξίσωση με τη βοήθεια των συναρτήσεων συμμετοχής γράφεται ως εξής:

$$\mu_{Xk+1}(y)=\max_x\{\min(\mu_{Uk}(u), \max_u (\min (\mu_{Xk}(x), \mu_R (u,x,y)))\}$$

Τα ασαφή συστήματα ανώτερης τάξης μπορούν να περιγραφούν ως εξής. Έστω η τάξη του συστήματος είναι p και οι ασαφείς σχεσιακές εξισώσεις οι ακόλουθες:

$$X_{k+p} = U_k \circ X_k \circ X_{k+1} \circ X_{k+2} \circ \dots \circ X_{k+p-1} \circ R$$

$$Y_{k+p} = X_{k+p} \circ S$$

Αντικαθιστώντας όπου $X=X_{k+p}'$, $X_{k+1} \circ X_{k+2} \circ \dots \circ X_{k+p-1} \circ R=R'$, $Y_{k+p} = Y_{k+p}'$ και $S=S'$ προκύπτει το ακόλουθο σύστημα ασαφών σχεσιακών εξισώσεων πρώτης τάξης.

$$X_{k+1}' = U_k \circ X_k \circ R'$$

$$Y_{k+1} = X_{k+1}' \text{ o } S'$$

Επομένως κάθε σύστημα οποιασδήποτε τάξης μπορεί να περιγράφει με ασαφείς εξισώσεις κατάστασης πρώτης τάξης.

Νευροασαφή συστήματα

Εισαγωγή

Μέχρι τώρα παρουσιάσαμε τα νευρωνικά δίκτυα και τα ασαφή συστήματα ξεχωριστά και είδαμε πώς μπορούν να χρησιμοποιηθούν για την επίλυση κάποιων προβλημάτων. Είδαμε επίσης ότι στα ασαφή συστήματα δεν προσπαθούμε να λύσουμε ένα πρόβλημα μοντελοποιώντας το σύστημα και να εφαρμόσουμε φυσικούς νόμους που ισχύουν γι' αυτό. Δεν ασχολούμαστε δηλαδή με τη δομή και ανάλυση του συστήματος. Απλώς χρησιμοποιούμε τη γνώση και εμπειρία ενός ειδικού (εμπειρογνώμονα) που ξέρει τη λειτουργία του συστήματος και τη συμπεριφορά του.

Ας θεωρήσουμε, για παράδειγμα, πώς οι άνθρωποι μαθαίνουν να οδηγούν ένα αυτοκίνητο. Δεν λύνουν διαφορετικές εξισώσεις που περιγράφουν τη φυσική συμπεριφορά του αυτοκινήτου. Αντίθετα, παρατηρούν κάποιον που ξέρει να οδηγεί ένα αυτοκίνητο και παίρνουν, αρχικά, κάποιες θεμελιώδεις γνώσεις χειρισμού του. Στη συνέχεια κάνοντας αποτυχημένες ή μη προσπάθειες αποκτούν εμπειρίες και τελικά ικανότητα οδήγησης.

Απ' αυτό το παράδειγμα μπορούμε να δούμε τρεις απόψεις για τη λύση ενός προβλήματος χωρίς να εξετάζουμε τις λεπτομέρειές του:

α) αποκτώντας γνώση από παρατήρηση

β) αποκτώντας γνώση από διδασκαλία

γ) αποκτώντας γνώση από μάθηση

Αυτά τα σημεία είναι ένα σημαντικό μέρος της ανάλυσης της γνώσης.

Στην ανάλυση της γνώσης το πρώτο πράγμα που χρειαζόμαστε είναι μια διαδικασία απόκτησης γνώσης. Τέτοιες διαδικασίες είναι η παρατήρηση και η συνδιάλεξη (interviews) σύμφωνα με τους McGraw και Harbison – Briggs. Γι' αυτό απαιτείται ένας ειδικός ο οποίος να μπορεί να εκφράσει τις γνώσεις του με τη μορφή γλωσσικών κανόνων (linguistic rules) κάτι το οποίο δεν είναι πάντοτε

δυνατό αφού ο ειδικός δεν μπορεί πάντα να μεταδώσει και να σχηματοποιήσει τις γνώσεις του. Σ' αυτή την περίπτωση είναι προτιμότερο να παρατηρήσουμε τον ειδικό καθώς λύνει το πρόβλημα που μας ενδιαφέρει και να γράψουμε τις ενέργειές του μαζί με τις αντίστοιχες καταστάσεις του προβλήματος. Δεν θα ασχοληθούμε άλλο με τον τρόπο με τον οποίο μπορούμε να αποκτήσουμε γνώση.

Τα νευρωνικά δίκτυα και τα ασαφή συστήματα λύνουν προβλήματα προσεγγίζοντας συναρτήσεις. Αν θέλουμε να χρησιμοποιήσουμε ένα νευρωνικό δίκτυο για να λύσουμε ένα πρόβλημα, πρέπει να περιγράψουμε το πρόβλημα επαρκώς με δείγματα δεδομένων. Αν έχουμε ζευγάρια εισόδου-εξόδου χρησιμοποιούμε εποπτευόμενη μάθηση όπως με τον back-propagation στα πολυεπίπεδα perceptrons. Αν δεν ξέρουμε τίποτα για τις τιμές εξόδου χρησιμοποιούμε ενισχυμένη μάθηση.

Στην επίλυση προβλημάτων με νευρωνικά δίκτυα δεν χρειαζόμαστε μαθηματικά μοντέλα για το πρόβλημα που μας ενδιαφέρει παρά μόνο δεδομένα εκπαίδευσης. Απ' την άλλη μεριά δεν μπορούμε να ερμηνεύσουμε τη λύση που εξασφαλίζουμε απ' τη διαδικασία μάθησης. Δεν μπορούμε δηλαδή να ελέγξουμε αν η λύση είναι εύλογη. Αυτό σημαίνει επίσης ότι δεν μπορούμε να αρχικοποιήσουμε ένα νευρωνικό δίκτυο με a priori γνώση, αν έχουμε. Η διαδικασία μάθησης είναι χρονοβόρα και δεν έχουμε εγγύηση επιτυχίας.

Ένα ασαφές σύστημα μπορεί να χρησιμοποιηθεί για να λύσει πρόβλημα, αν έχουμε γνώση για τη λύση του, με τη μορφή γλωσσικών IF-THEN κανόνων. Δεν χρειαζόμαστε ένα σχήμα του μοντέλου του προβλήματος που μας ενδιαφέρει καθώς και δεδομένα εκπαίδευσης τα οποία δύσκολα μπορούμε να τα εξασφαλίσουμε. Απ' την άλλη μεριά ένα ασαφές σύστημα δεν μπορεί να βγάλει αποτέλεσμα χωρίς ασαφούς κανόνες.

Ανακεφαλαιώνοντας τη σύγκριση των νευρωνικών δικτύων και των ασαφών συστημάτων καταλήγουμε στα εξής πλεονεκτήματα και μειονεκτήματα:

<u>Νευρωνικά Δίκτυα</u>

Πλεονεκτήματα:

1) Δεν απαιτείται μαθηματική διαδικασία μοντέλου.

2) Δεν απαιτείται γνώση βασισμένη σε κανόνες.
3) Διαφορετικοί αλγόριθμοι μάθησης είναι διαθέσιμοι.

Μειονεκτήματα:

1) Μαύρο κουτί (Δεν ξέρουμε πώς και γιατί βγάζουν ένα αποτέλεσμα).
2) Η προσαρμοστικότητά τους σε τροποποιημένο περιβάλλον είναι δύσκολη και είναι αναγκαία ή επανεκπαίδευσή τους.
3) Δεν μπορούμε να χρησιμοποιήσουμε a priori γνώση.
4) Δεν εγγυώνται τη σύγκλιση μάθησης.

Ασαφή συστήματα

Πλεονεκτήματα:

1) Δεν απαιτείται η γνώση του μοντέλου.
2) Μπορούν να χρησιμοποιήσουν a priori γνώση.
3) Απλή ερμηνεία και υλοποίηση.

Μειονεκτήματα:

1) Είναι απαραίτητη η δημιουργία κανόνων.
2) Δεν μπορούν να μάθουν.
3) Δεν υπάρχουν σχηματοποιημένες μέθοδοι για συντονισμό (tuning).
4) Η προσαρμογή σε τροποποιημένο περιβάλλον είναι δύσκολη.
5) Ο συντονισμός μπορεί να μην είναι επιτυχής.

Προκειμένου να εξαλείψουμε τα μειονεκτήματα των νευρωνικών και ασαφών συστημάτων μπορούμε να τα συνδυάσουμε φτιάχνοντας τα νευροασαφή συστήματα.

Ορισμός νευρο-ασαφούς συστήματος

Στα νευροασαφή συστήματα βρίσκουμε τις παραμέτρους ενός ασαφούς συστήματος χρησιμοποιώντας τις μεθόδους μάθησης των ΝΝ (νευρωνικών δικτύων). Εδώ υπάρχει ένα μικρό πρόβλημα: η εφαρμογή αλγορίθμων μάθησης, στα ασαφή συστήματα χωλαίνει στο γεγονός ότι οι αλγόριθμοι χρησιμοποιούν μεθόδους gradient-descent, ενώ οι συναρτήσεις των ασαφών

συστημάτων δεν είναι διαφοροποιήσιμες. Αυτό το πρόβλημα λύνεται αντικαθιστώντας τις συναρτήσεις που χρησιμοποιούνται από τα ασαφή συστήματα με διαφοροποιήσιμες συναρτήσεις και αποφεύγοντας τη χρήση αλγορίθμων μάθησης βασισμένες στην gradient-descent μέθοδο. Υπάρχουν μοντέλα (ANFIS) για παράδειγμα που υλοποιούν ένα sugeno ασαφές σύστημα σε μια νευρωνική δομή και εφαρμόζουν μια μικτή διαδικασία back-propagation και ελάχιστων μέσων τετραγώνων για την εκπαίδευση του συστήματος.

Υπάρχουν πολλές διαφορετικές προσεγγίσεις των neuro-fuzzy systems που έχουν πολλά κοινά αλλά διαφέρουν στην υλοποίησή τους. Για να βρούμε τα κοινά χαρακτηριστικά όλων αυτών των προσεγγίσεων και να δώσουμε στον όρο «neuro-fuzzy system» νόημα, πρέπει να τον περιορίσουμε σε συστήματα που έχουν τις ακόλουθες ιδιότητες:

1) Ένα νευρο ασαφές (neuro-fuzzy) σύστημα είναι ένα ασαφές σύστημα που εκπαιδεύεται από έναν αλγόριθμο μάθησης προερχόμενο απ' τη θεωρία NN. Η διαδικασία μάθησης λειτουργεί σε τοπικές πληροφορίες και προκαλεί μόνο τοπικές τροποποιήσεις στο ασαφές σύστημα.

2) Ένα νευρο-ασαφές σύστημα (NFS) μπορεί να ερμηνευτεί σαν ένα σύστημα από ασαφούς κανόνες. Μπορούμε να εισάγουμε σ' αυτό και δεδομένα εκπαίδευσης και προθύστερη γνώση με τη μορφή ασαφών κανόνων.

3) Η διαδικασία μάθησης ενός NFS λαμβάνει υπόψη τις σημαντικές ιδιότητες των ασαφών συστημάτων.

4) Ένα NFS προσεγγίζει μια η-διάστατη άγνωστη συνάρτηση η οποία δίνεται εν μέρει απ' τα δεδομένα εκπαίδευσης. Οι ασαφείς κανόνες κωδικοποιούνται μέσα στο σύστημα αναπαριστώντας αόριστα δείγματα, και μπορούν να θεωρηθούν σαν αόριστα πρωτότυπα των δεδομένων εκπαίδευσης. Ένα NFS δεν μπορεί να θεωρηθεί σαν ένα είδος ασαφούς συστήματος και δεν έχει να κάνει με ασαφή λογική.

5) Ένα NFS μπορεί να θεωρηθεί σαν ένα ειδικό 3-επιπέδων προσοτροφοδοτούμενο NN. Οι μονάδες σ' αυτό το δίκτυο χρησιμοποιούν t-norms ή t-conorms αντί για τις συναρτήσεις ενεργοποίησης που κανονικά χρησιμοποιούνται στα NN: Το πρώτο επίπεδο αναπαριστά μεταβλητές εισόδου, το δεύτερο τους ασαφείς κανόνες και το τρίτο τις μεταβλητές εξόδου. Τα ασαφή

σύνολα κωδικοποιούνται σαν συνδέσεις βαρών. Μερικά NFS χρησιμοποιούν περισσότερα από 3 επίπεδα και κωδικοποιούν τα ασαφή σύνολα σαν συναρτήσεις ενεργοποίησης.

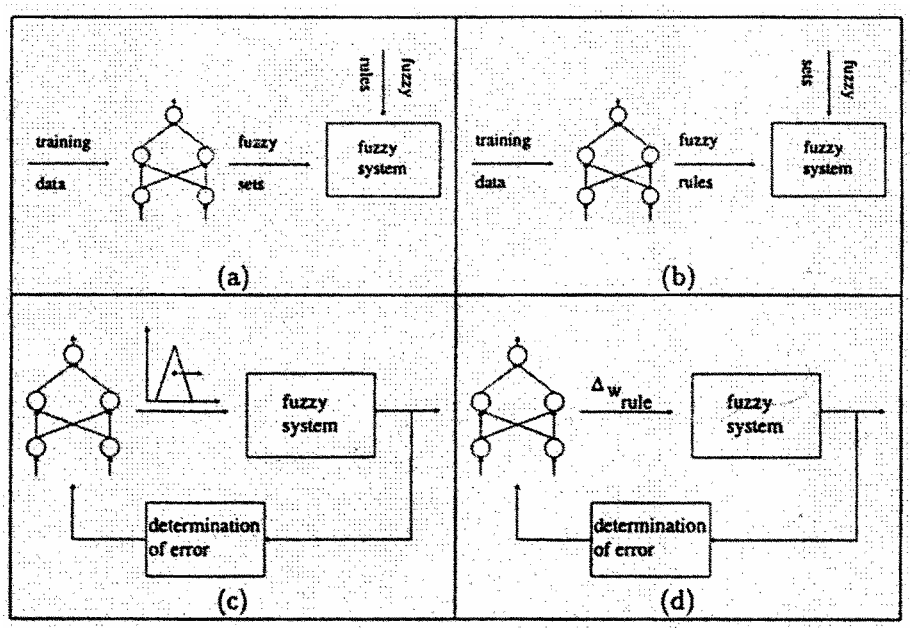
Τύποι νευρο-ασαφών (NFS) συστημάτων

Υπάρχουν δύο είδη συνδυασμού των NN και των ασαφών συστημάτων (FS):

i) Στην πρώτη τα NN και το FS δουλεύουν ανεξάρτητα το ένα από το άλλο. Σ' αυτήν την περίπτωση ο συνδυασμός απλώνεται στον καθορισμό συγκεκριμένων παραμέτρων του ασαφούς συστήματος (FS) από ένα NN (νευρωνικό δίκτυο) ή από έναν αλγόριθμο μάθησης του NN. Αυτό γίνεται on-line ή off-line. Αυτό το είδος συνδυασμού καλείται cooperative neuro-fuzzy system.

ii) Στη δεύτερη περίπτωση έχουμε μια ομογενή δομή συνήθως όμοια με τη δομή των NN. Αυτό μπορεί να γίνει ερμηνεύοντας το FS σαν μια ειδική περίπτωση ενός NN. Αυτό το είδος συνδυασμού καλείται υβριδικό νευρο-ασαφές σύστημα. Ο όρος «υβριδικό» (hybrid) χρησιμοποιείται όταν δύο διαφορετικές διαδικασίες προσέγγισης εργάζονται μαζί. Συνεπώς τα υβριδικά NFS μπορούν να θεωρηθούν είτε σαν NN είτε σαν FS, αλλά έχουμε μόνο ένα σύστημα που δεν μπορεί να διαχωριστεί σε δύο διαφορετικά συστήματα.

Στα παρακάτω σχήματα φαίνονται τα συνεργαζόμενα (cooperative) νευρο-ασαφή συστήματα:



1. (α) Σ’ αυτό το σχήμα το ΝΝ καθορίζει τις συναρτήσεις συμμετοχής από τα δεδομένα εκπαίδευσης (training data). Αυτό μπορεί να γίνει με το να καθορίσουμε κατάλληλες παραμέτρους ή με το να προσεγγίσουμε τις συναρτήσεις συμμετοχής με ένα ΝΝ. Μ’ αυτόν τον τρόπο ορίζουμε off-line τα ασαφή σύνολα (fuzzy sets) που σε συνδυασμό με τους ανεξάρτητα ορισμένους ασαφείς κανόνες (fuzzy rules) υλοποιούν ένα ασαφές σύστημα (fuzzy system).

1. (β) Σ’ αυτήν την περίπτωση το ΝΝ ορίζει τους ασαφείς κανόνες με τη βοήθεια των δεδομένων εκπαίδευσης. Αυτό επιτυγχάνεται με μια ομαδοποιημένη προσέγγιση που υλοποιείται με αυτοοργανούμενους χάρτες ή παρόμοιες νευρωνικές αρχιτεκτονικές. Το ΝΝ χρησιμοποιείται πριν την υλοποίηση του ασαφούς συστήματος και μαθαίνει τους κανόνες off-line ενώ οι συναρτήσεις συμμετοχής καθορίζονται ξεχωριστά.

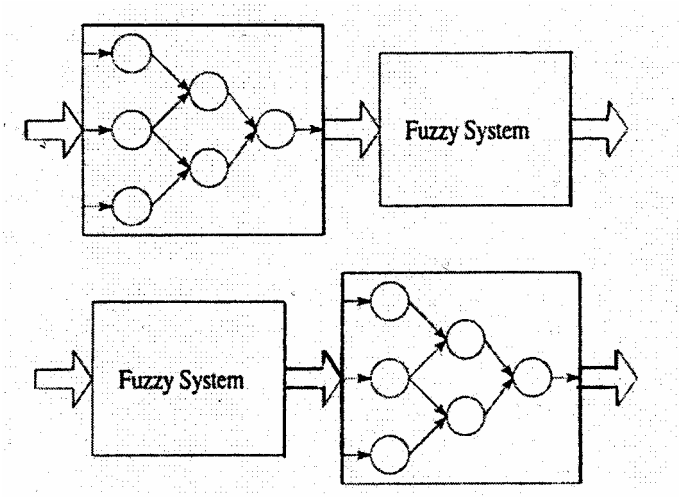
1. (γ) Σ’ αυτό το μοντέλο οι ασαφείς κανόνες και οι αρχικές συναρτήσεις συμμετοχής πρέπει να είναι γνωστές. Το σύστημα μαθαίνει τις παραμέτρους on-line. Δεν υπάρχει στην πραγματικότητα κανένα ΝΝ σ’ αυτό τον τύπο νευρο-ασαφούς συστήματος παρά μονάχα ένας αλγόριθμος μάθησης.

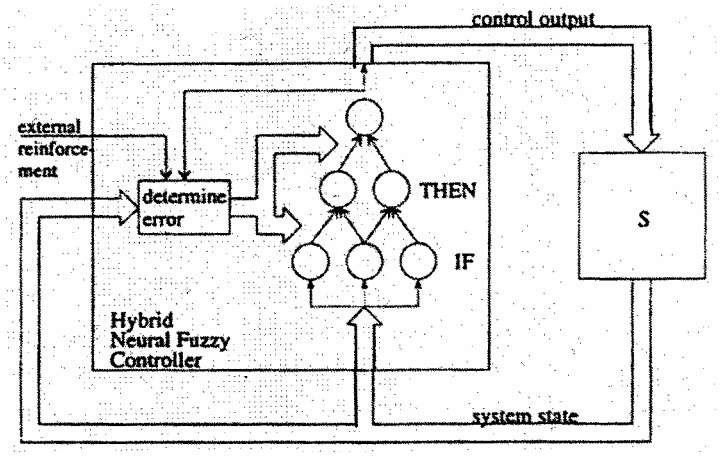
1. (δ) Ένα ΝΝ καθορίζει τα βάρη των κανόνων για τους ασαφείς κανόνες είτε on-line είτε off-line. Τα βάρη των κανόνων μπορούν να αντικατασταθούν ισοδύναμα από τροποποιημένες συναρτήσεις συμμετοχής.

Όπως βλέπουμε στις παραπάνω περιπτώσεις NFS τα ΝΝ χρησιμοποιούνται σαν προεπεξεργαστές για ένα ασαφές σύστημα. Δηλαδή δεν βελτιώνουμε, με αυτό τον τρόπο, ένα ασαφές σύστημα αλλά βελτιώνουμε την απόδοση του συνολικού (NFS) συστήματος. Επίσης η διαδικασία μάθησης χρησιμοποιείται μόνο από τα ΝΝ ενώ τα ασαφή συστήματα παραμένουν αμετάβλητα. Σε αντίθεση με τα συμβατικά ΝΝ, τα cooperative NFS προσφέρουν το πλεονέκτημα μιας γρήγορης υλοποίησης με τη χρήση a priori γνώσης.

Υβριδικά νευροασαφή συστήματα

Στο σχήμα 2 παριστάνεται ένα υβριδικό νευροασαφές μοντέλο στη μορφή ενός νευροασαφούς ελεγκτή. Το πλεονέκτημα των υβριδικών νευροασαφών συστημάτων (HNFS) είναι η ενοποιημένη αρχιτεκτονική τους.





Μπορούν να εκπαιδευτούν τόσο on-line όσο και off-line.

Στα HNFS η βάση κανόνων του ασαφούς συστήματος ερμηνεύονται σαν όροι του νευρωνικού δικτύου. Τα ασαφή σύνολα μπορούν να θεωρηθούν σαν βάρη και οι μεταβλητές εισόδου και εξόδου και οι κανόνες μπορούν να ερμηνευθούν σαν νευρώνες. Κατά αυτόν τον τρόπο ένα ασαφές σύστημα μπορεί να θεωρηθεί σαν μια ειδική περίπτωση ενός νευρωνικού δικτύου. Ο αλγόριθμος μάθησης δουλεύει με τροποποίηση της δομής και/ή των παραμέτρων. Οι αλλαγές που οφείλονται στη διαδικασία μάθησης μπορούν να (εξηγηθούν) ερμηνευτούν από την άποψη και των NN και FS. Αυτό είναι σημαντικό επειδή η συνηθισμένη συμπεριφορά μαύρου κουτιού των NN αποφεύγεται.

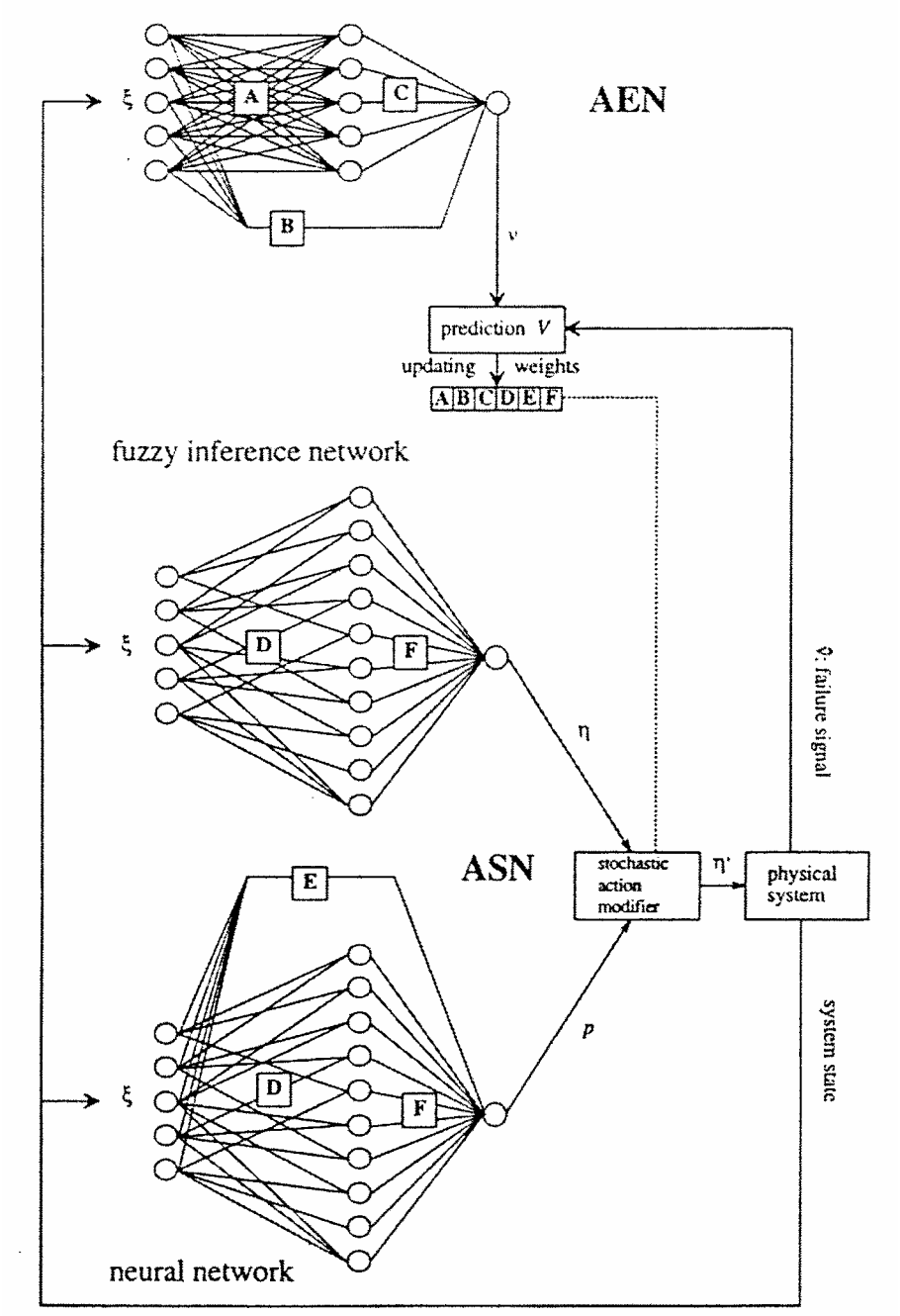
Ένα HNFS μαθαίνει με εποπτευόμενο τρόπο. Παρακάτω περιγράφουμε διάφορα HNFS (hybrid neuro-fuzzy system):

i) Το ARIC μοντέλο
ii) Το Critic δίκτυο AEN
iii) Το Action δίκτυο ASN
iv) Το ANFIS μοντέλο
v) Το NNDFR μοντέλο
vi) Το FuNe μοντέλο

i) Το ARIC μοντέλο

Το ARIC μοντέλο (Approximate Reasoning based Intelligent Control) είναι ένα υβριδικό νευροασαφές μοντέλο που υλοποιεί ένα ασαφή ελεγκτή χρησιμοποιώντας πολλά ειδικευόμενα προσοτροφοδοτούμενα NN. Η αρχιτεκτονική του ARIC είναι παρόμοια με ένα προσαρμοστικό κριτικό, ένας ειδικός νευρωνικός ελεγκτής που μαθαίνει με ενίσχυση και γενίκευση το νευρωνικό μοντέλο του Barto στο πεδίο του ασαφούς ελέγχου. Το ARIC αποτελείται από δύο πλαίσια: το ASN (Action Selection Network) και το AEN (Action State Evaluation Network). Το AEN είναι ένας προσαρμοστικός κριτικός που αποτιμά τις πράξεις του ASN.

Το ASN αποτελείται από δύο προσοτροφοδοτούμενα τριών επιπέδων NN. Το δίκτυο ελέγχου είναι μια απευθείας αναπαράσταση ενός ασαφούς ελεγκτή. Το στρώμα (επίπεδο) εισόδου αναπαριστά μεταβλητές κατάστασης μιας διαδικασίας και οι κρυμμένες μονάδες αναπαριστούν τους ασαφείς κανόνες. Οι είσοδοί τους είναι οι ακόλουθοι των κανόνων. Το ARIC θεωρεί ότι η βάση κανόνων είναι γνωστή. Η έξοδος του δικτύου ελέγχου αναπαριστά την αποασαφοποιημένη τιμή ελέγχου του ασαφούς ελεγκτή. Το δεύτερο δίκτυο υπολογίζει μια τιμή που χρησιμοποιείται για να αλλάξει την έξοδο του δικτύου ελέγχου. Δεν θα ασχοληθούμε αναλυτικά με το ARIC μοντέλο το οποίο φαίνεται στο παρακάτω σχήμα



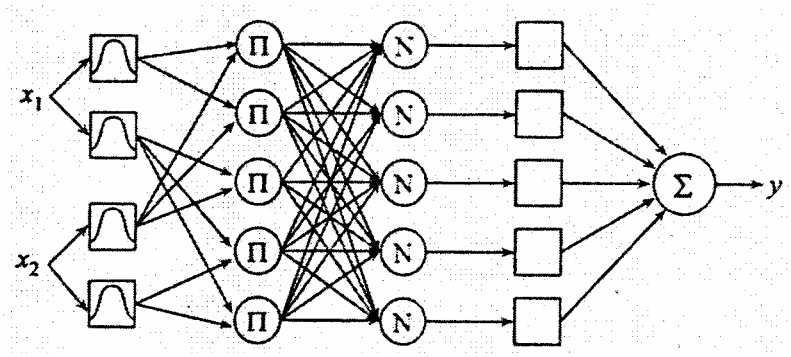
ii) Το ANFIS μοντέλο

Το ANFIS (Adaptive Network based Fuzzy Inference System) το οποίο αναπτύχθηκε απ' τον Jang είναι ένα από τα πρώτα υβριδικά νευρο-ασαφή συστήματα. Αναπαριστά ένα Sugeno-τύπο ασαφές σύστημα σε μια ειδική πέντε-στρωμάτων προσοτροφοδοτούμενη δομή δικτύου. Το ANFIS υλοποιεί κανόνες της μορφής:

$$R_r : \text{If } x_1 \text{ is } A_{j1}^{(1)} \Lambda \ldots \Lambda x_n \text{ is } A_{jn}^{(n)} + \text{hen}$$

$$y = \alpha_0^{(r)} + \alpha_1^{(r)} x_1 + \ldots + \alpha_n^{(r)} x_n$$

Ένα ANFIS μοντέλο φαίνεται στο παρακάτω σχήμα.



Στρώμα 1: Κάθε μονάδα στο U_1 (πρώτο στρώμα) αποθηκεύει τρεις παραμέτρους για να ορίσουν μια καμπανωτή (bell-shaped) συνάρτηση συμμετοχής που αναπαριστά έναν γλωσσικό όρο:

$$\mu_j^i(x_i) = \frac{1}{1 + \left(\left(\frac{x_i - \iota}{\alpha} \right)^2 \right)^b}$$

όπου x_i είναι μια μεταβλητή εισόδου. Κάθε μονάδα είναι συνδεμένη σε ακριβώς μία μονάδα εισόδου και υπολογίζει το βαθμό συμμετοχής της επικρατούσας τιμής εισόδου.

Στρώμα 2: Κάθε κανόνας αναπαριστάται από μια μονάδα στο U_2. Κάθε μονάδα συνδέεται σ' αυτές τις μονάδες του προηγούμενου στρώματος οι οποίες είναι οι προγενέστεροι (πρόγονοι) των κανόνων.

Στρώμα 3: Σ' αυτό το στρώμα για κάθε κανόνα R_r υπάρχει ένα στρώμα που υπολογίζει το δικό του βαθμό πληρότητας.

$$\bar{\tau}_r = \frac{\tau_r}{\sum_{R_i \in U_2} \tau_i}$$

Κάθε μονάδα συνδέεται σε όλες τις μονάδες κανόνων του U_2.

Στρώμα 4: Οι μονάδες του U_4 είναι συνδεμένες με όλες τις μονάδες εισόδου και σε ακριβώς μία μονάδα του U_3. Κάθε μονάδα υπολογίζει την έξοδο του κανόνα R_r με:

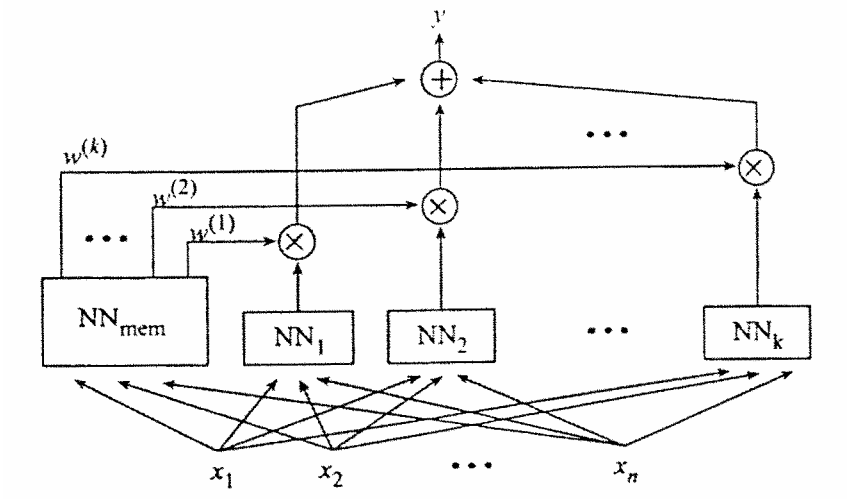
$$O_r = \bar{\tau}_r\left(\alpha_0^{(r)} + \alpha_1^{(r)} x_1 + \ldots + \alpha_n^{(r)} x_n\right)$$

Στρώμα 5: Μια μονάδα εξόδου υπολογίζει την τελική έξοδο y προσθέτοντας όλες τις εξόδους του U_4.

iii) Άλλοι τύποι υβριδικών NFS

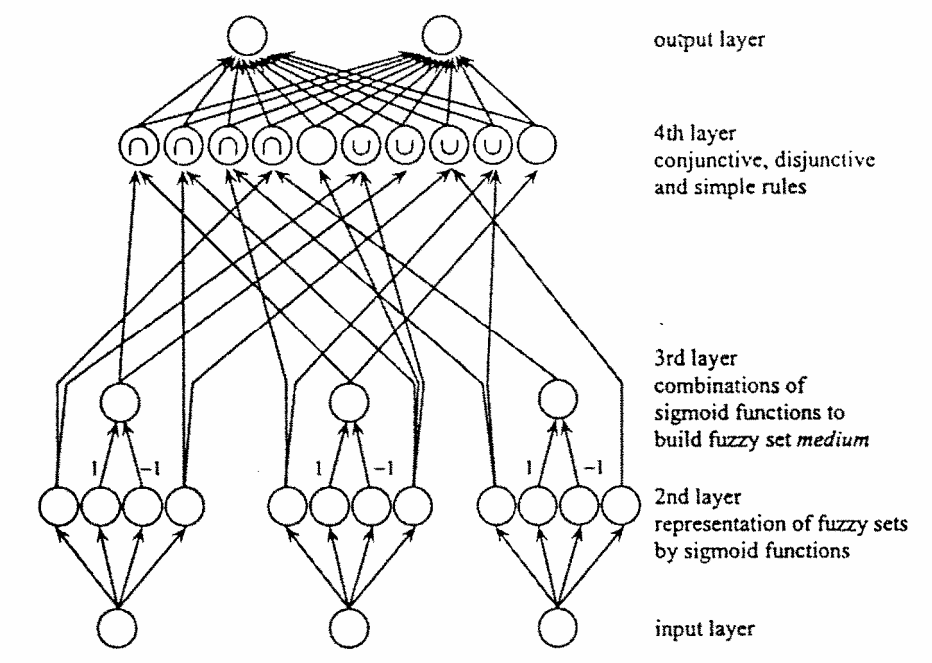
Παρακάτω παραθέτουμε κάποια HNFS χωρίς να τα αναλύσουμε/

α) NNDFR (Neural Network Driven Fuzzy Reasoning)



β) Το FuNe μοντέλο

1.



Εισαγωγή

Σε αυτό το τμήμα της διπλωματικής θα αναπτύξουμε και θα εφαρμόσουμε μια τεχνική για την αναγνώριση (indentification) και γραμμικοποίηση ενός άγνωστου διακριτού χρόνου μη γραμμικού δυναμικού συστήματος όπως αυτή παρουσιάζεται απ' τους Qiang Gan και Chris J. Harris. Η τεχνική βασίζεται σε αναδρομικά νευροασαφή δίκτυα και δομές χρησιμοποιώντας B-spline συναρτήσεις σαν συναρτήσεις συμμετοχής. Η τεχνική εφαρμόζεται στο σύστημα του πλοίου το οποίο είναι άγνωστο, διακριτού χρόνου μη γραμμικό και δυναμικό.

Παρουσιάζουμε και περιγράφουμε τον MASMOD (modified algorithm for adaptive spline modeling of observation data) αλγόριθμο με τη βοήθεια του οποίου αναγνωρίζουμε και γραμμικοποιούμε το σύστημα του πλοίου. Είναι βασικό να σημειώσουμε ότι η γραμμικοποίηση είναι απαραίτητη λόγω του πλήθους των μεθόδων που έχουν αναπτυχθεί και εφαρμοσθεί για τον έλεγχο και την εκτίμηση κατάστασης γραμμικών συστημάτων. Τέλος πρέπει να τονίσουμε ότι η καινούργια μέθοδος έχει χρήσιμες πρακτικές ιδιότητες όπως οικονομία στις παραμέτρους του μοντέλου και σύγκλιση μάθησης (model parametric parsimony and learning convergence)

1. Γραμμοποίηση και εκτίμηση κατάστασης αγνώστων διακριτού χρόνου μη γραμμικών δυναμικών συστημάτων χρησιμοποιώντας αναδρομικά νευροαδαφή δίκτυα.

Τα μη γραμμικά συστήματα σύμφωνα με τη θεώρηση της διπλωματικής είναι βασικά διακριτού χρόνου και μπορούν να περιγραφούν απ' ακόλουθες γενικές μη γραμμικές εξισώσεις κατάστασης:

$$x(t+1) = f[x(t), u(t) + \upsilon(t)]$$

$$y(t) = h[x(t)] + w(t)$$

όπου

t = διακριτός χρόνος

$x(t) = [x_1(t), \ldots, x_n(t)]^T$ διάνυσμα κατάστασης

$u(t) = [u_1(t), \ldots, u_m(t)]^T$ διάνυσμα εισόδου

$y(t) = [y_1(t), \ldots, y_M(t)]^T$ διάνυσμα παρατήρησης

$f[\cdot]$ άγνωστη ομαλή, μη γραμμική,

	διανυσματική συνάρτηση
h[·]	ομαλή διανυσματική συνάρτηση που αναπαριστά τη διαδικασία μέτρησης
υ(t)	προσθετικός Gaussian θόρυβος
w(t)	παρατηρήσεις Gaussian θόρυβος

Όπως αναφέραμε και στην εισαγωγή η γραμμικοποίηση αποτελεί ένα σημαντικό εργαλείο στην επίλυση μη γραμμικών προβλημάτων γιατί μας δίνει τη δυνατότητα χρήσης καλά αναπτυγμένων γραμμικών τεχνικών για εκτίμηση κατάστασης και προσαρμοστικού ελέγχου στα μη γραμμικά συστήματα. Οι μέθοδοι για γραμμικοποίηση περιλαμβάνουν αναπτύγματα σειρών Taylor τοπική γραμμικοποίηση (local linearization) και γραμμικοποίηση με ανατροφοδότηση (feedback linearization). Αυτές οι μέθοδοι υποθέτουν συνθήκη ομαλότητας της μη γραμμικής συνάρτησης γνωστές ως Lipschitz συνθήκες. Οι δύο παραπάνω τεχνικές γραμμικοποίησης έχουν αναπτυχθεί και πετύχει τα τελευταία χρόνια.

Η τεχνική που θα αναπτύξουμε χρησιμοποιεί προσαρμοστικά κατασκευασμένα νευροασαφή δίκτυα τα οποία μπορούν να θεωρηθούν σαν προσέγγιση της γραμμικοποίησης με ανατροφοδότηση.

Η γραμμικοποίηση με ανατροφοδότηση είναι μια τεχνική με την οποία αναγκάζουμε ένα μη γραμμικό σύστημα να συμπεριφερθεί γραμμικά με ένα μετασχηματισμό συντεταγμένων κατάστασης και έναν ανατροφοδοτούμενο νόμο ελέγχου.

Η γραμμικοποίηση με ανατροφοδότηση έχει εφαρμοσθεί σε συνεχούς χρόνου μη γραμμικά συστήματα τα περισσότερα απ' τα οποία οριοθετούνται στην παρακάτω ομάδα μη γραμμικών συστημάτων:

$$\dot{x} = f(x,u) = p(x) + G(x)u \qquad (3)$$

$$y=h(x)$$

όπου η διανυσματική συνάρτηση p(·) και η συνάρτηση πίνακας G(·) είναι μη γραμμικές αλλά ομαλές. Με τη βοήθεια Lie παραγώγων μπορούμε να κατασκευάσουμε έναν μετασχηματισμό συντεταγμένων κατάστασης και έναν ανατροφοδοτούμενο νόμο ελέγχου οπότε το σύστημα που περιγράφεται απ' την (3) παίρνει την ακόλουθη γραμμικοποιημένη με ανατροφοδότηση μορφή:

$$\dot{z} = A_c z + B_c r(z,u) \quad (4)$$

$$y = C_c z \quad (5)$$

με $$z = T(x) \quad (6)$$

$$r(z,u) = q(z) + S(z)u \quad (7)$$

Δεν θα αναλύσουμε άλλο τη γραμμικοποίηση με ανατροφοδότηση για συνεχούς χρόνου συστήματα επειδή το σύστημά μας δεν κατατάσσεται σ' αυτήν την κατηγορία.

Η θεωρητική ανάλυση της γραμμικοποιησημότητας διακριτού χρόνου μη γραμμικών συστημάτων είναι ένα πολύ δύσκολο θεωρητικό πρόβλημα. Σε αντίθεση με τα συνεχούς χρόνου συστήματα όπου μπορούμε να κατασκευάσουμε έναν μετασχηματισμό συντεταγμένων κατάστασης και έναν ανατροφοδοτούμενο νόμο ελέγχου στα διακριτά συστήματα δεν υπάρχουν θεωρητικά αποτελέσματα. Η τεχνική που παρουσιάζουμε χρησιμοποιεί ένα προσαρμοστικά κατασκευασμένο αναδρομικό νευροασαφές δίκτυο και το δικό του σχήμα μάθησης και προσπαθεί να αποδείξει τη σχέση αυτής της μεθόδου με τη συμβατική γραμικοποίηση με ανατροφοδότηση.

1.1 Γραμμικοποίηση χρησιμοποιώντας νευρωνικά δίκτυα

1.1.1 Τοπική γραμμικοποίηση

Μια απ' τις πολλές εφαρμογές των νευρωνικών δικτύων είναι η γραμμικοποίηση μη γραμμικών δυναμικών συστημάτων στα οποία η μη γραμμική συνάρτηση είναι άγνωστη ή αβέβαιη. Εκτός απ' τη γραμμικοποίηση με ανατροφοδότηση στην οποία αναφερθήκαμε παραπάνω τα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν και στην τοπική γραμμικοποίηση (local linearization) την οποία θα αναπτύξουμε πρώτα.

Η τοπική γραμμικοποίηση βασίζεται στο ακόλουθο μοντέλο Takagi – Sugeno ασαφές μοντέλο:

$$f(\xi) = \hat{f}(\xi) + \Delta f(\xi) = \sum_{\kappa=1}^{K} \mu_\kappa(\xi) l_\kappa(\xi) + \Delta f(\xi) \quad (8)$$

όπου ξ αναπαριστά ένα διάστημα εισόδου το οποίο μπορεί επίσης να περιέχει ανατροφοδοτούμενες καταστάσεις ή προηγούμενες εξόδους, f(ξ) είναι η μη γραμμική συνάρτηση που θα μοντελοποιήσουμε, Δf(ξ) αναπαριστά το σφάλμα

προσέγγισης $\mu_\kappa(\xi)$ είναι συναρτήσεις συμμετοχής των ασαφών συνόλων που διαιρούν το διάστημα εισόδου και $l_\kappa(\xi)$ είναι τα τοπικά γραμμικά μοντέλα:

$$l_\kappa(\xi) = \alpha_{k0} + \alpha_\kappa^T \xi \qquad \text{με} \qquad \alpha_k = [\alpha_{\kappa 1}, \alpha_{\kappa 2}, \ldots, \alpha_{\kappa n}]^T$$

Επειδή το διάστημα εισόδου είναι ασαφώς διαιρεμένο, η (8) αναπαριστά ένα ασαφές τοπικά γραμμικοποιημένο μοντέλο.

Η εξίσωση (8) μπορεί να γραφτεί με τον ακόλουθο τρόπο:

$$\hat{f}(\xi) = \sum_{\kappa=1}^{K} \mu_\kappa(\xi)\left(\alpha_{\kappa 0} + \alpha_\kappa^T \xi\right) = \sum_{\kappa=1}^{K}\left[\mu_\kappa(\xi)\sum_{i=0}^{n}\alpha_{\kappa i}\xi_\kappa\right] = \sum_{i=0}^{\eta}\left[\sum_{\kappa=1}^{K}\mu_\kappa(\xi)\alpha_{\kappa i}\right]\xi_i =$$

$$= \sum_{i=0}^{K} \alpha_i(t)\xi_i \qquad (9)$$

όπου $\xi_0=1$ και $\alpha_i(t)$ αναπαριστούν χρονο-μεταβλητούς συντελεστές σε ένα χρονο-μεταβλητό γραμμικό μοντέλο. Βλέπουμε δηλαδή ότι ένα μη χρονο-μεταβλητό, μη γραμμικό σύστημα μετασχηματίζεται σε ένα χρονο-μεταβλητό γραμμικό σύστημα με τους χρονο-μεταβλητούς συντελεστές εκτιμούμενους από νευροασαφή δίκτυα $\sum_{\kappa=1}^{K}\mu_\kappa(\xi)\alpha_{\kappa i}$. Αν η f(ξ) είναι προσεγγιστικά γραμμική σε μια περιοχή λειτουργίας οι συντελεστές $\alpha_i(t)$ θα αλλάζουν αργά μέσα σ' αυτήν την περιοχή λειτουργίας.

Χρησιμοποιώντας νευρωνικούς αλγόριθμους μάθησης μπορούμε να εξασφαλίσουμε τις δομές των ασαφών συνόλων (fuzzy set) και τις ελεύθερες παραμέτρους του μοντέλου, ενώ πρέπει να αναγνωρίσουμε τόσο τις συναρτήσεις συμμετοχής όσο και τα τοπικά γραμμικά μοντέλα. Έχει αναπτυχθεί ένα ANFIS (adaptive – network – based fuzzy inference system), σύστημα στο οποίο Gaussian συναρτήσεις χρησιμοποιούνται σαν συναρτήσεις συμμετοχής και οι ελεύθερες παράμετροι και στις συναρτήσεις συμμετοχής και στα τοπικά γραμμικά μοντέλα αναγνωρίζονται χρησιμοποιώντας ένα απότομης κλίσης – καθόδου τύπο αλγορίθμου, γνωστός ως gradient – descent type algorithm. Άλλος τρόπος για την επίλυση του προβλήματος είναι η χρησιμοποίηση B-spline συναρτήσεων σαν συναρτήσεις συμμετοχής και ένα κανονικοποιημένο ελαχίστων μέσων τετραγώνων αλγόριθμο (least – mean – squares) για ενημέρωση των παραμέτρων στα τοπικά γραμμικά μοντέλα. Οι Wu και Harris έχουν εφαρμόσει ένα νευροασαφές τοπικά γραμμικοποιημένο μοντέλο για μη γραμμική εκτίμηση κατάστασης. Ένα σημαντικό ζήτημα το οποίο δεν αντιμετωπίζεται στις παραπάνω περιπτώσεις είναι το πώς θα διαιρέσουμε το

διάστημα εισόδου χρησιμοποιώντας ασαφή σύνολα αποτελεσματικά και αυτόματα. Με σκοπό να λύσουν αυτό το πρόβλημα ο Gan και ο Harris ανασυνθέτουν το ασαφές τοπικά γραμμικοποιημένο μοντέλο σε υπομοντέλα σε μια ANOVA (analysis of variance) μορφή και προτείνουν ένα τροποποιημένο προσαρμοστικό spline μοντέλο αλγορίθμου (MASMOD) για αυτόματη ασαφής διαίρεση του διαστήματος εισόδου, στον οποίο οι B-splines, σαν συναρτήσεις συμμετοχής, κατασκευάζονται αυτόματα. Δηλαδή, ο αριθμός των υπομοντέλων, ο αριθμός των B-splines που απαιτούνται σε κάθε υπομοντέλο και οι θέσεις τους και τα σχήματά τους είναι αυτόματα καθορισμένα από τον MASMOD αλγόριθμο. Πρέπει να σημειωθεί ότι η τοπική γραμμικοποίηση είναι μια τεχνική που εφαρμόζεται με την προϋπόθεση ότι το σύστημα είναι τοπικά γραμμικοποιήσιμο. Σε διαφορετική περίπτωση η σύγκλιση του αλγορίθμου για εκτίμηση κατάστασης ή έλεγχο βασισμένο σε τοπικά γραμμικοποιημένα μοντέλα δεν μπορεί να είναι εγγυημένη.

1.1.2 Γραμμικοποίηση με ανατροφοδότηση

Η γραμμικοποίηση με ανατροφοδότηση χρησιμοποιώντας νευρωνικά δίκτυα έχει προσελκύσει μεγάλο ενδιαφέρον τα τελευταία χρόνια. Οι μέχρι τώρα αναπτυχθείσες τεχνικές όσον αφορά την γραμμικοποίηση με ανατροφοδότηση απαιτούν βαρύ υπολογιστικό φορτίο και μπορεί να μην είναι αρκετά ορθές ώστε να εγγυηθούν τη σύγκλιση της ενημέρωσης (update) των παραμέτρων και του προσαρμοστικού ελέγχου. Τα αποτελέσματα της γραμμικοποίησης με ανατροφοδότηση βασισμένα στην προσέγγιση με νευρωνικά δίκτυα των μη γραμμικών συναρτήσεων σε πρωτότυπες συντεταγμένες μπορούν να θεωρηθούν σαν τα ήδη υπάρχοντα θεωρήματα και απαιτείται περαιτέρω εργασία για να κάνεις αυτά τα αποτελέσματα πιο ρεαλιστικά. Εναλλακτικά επιτυγχάνεται μεγάλο κέρδος προσεγγίζοντας απευθείας τον μη γραμμικό ανατροφοδοτούμενο όρο στην κανονική μορφή. Προκειμένου να προσεγγίσουμε τις μη γραμμικές συναρτήσεις στην κανονική τους μορφή μπορούμε να χρησιμοποιήσουμε είτε πολύ επίπεδα νευρωνικά δίκτυα είτε γραμμικά στις παραμέτρους δίκτυα. Αλγόριθμοι ρύθμισης βαρών των νευρωνικών δικτύων έχουν αναπτυχθεί και η σύγκλισή τους καθώς και η ευστάθειά τους έχει αναλυθεί. Πάντως, όλη η εργασία σ’ αυτή τη διπλωματική υποθέτει ότι το διάνυσμα κατάστασης στην κανονική μορφή είναι παρατηρήσιμο ή το πρότυπο,

μη γραμμικό σύστημα μπορεί να περιγραφεί από ένα μη γραμμικό ARMA μοντέλο το οποίο μπορεί να μετατραπεί σε μια κανονική διαστημα-κατάσταση (state – space) μορφή με καθυστερημένες εξόδους σαν καταστάσεις (states) π.χ. τα μη γραμμικά συστήματα πρέπει να έχουν την ακόλουθη μορφή:

$$
\begin{aligned}
&x_1(t+1) = x_2(t)\\
&\vdots\\
&x_{n-1}(t+1) = x_n(t)\\
&x_n(t+1) = f\left[x(t), \theta_f\right] + \sum_j g_j\left[x(t), \theta_{gj}\right] u_j + d(t)\\
&y(t) = x_1(t) \qquad (10)
\end{aligned}
$$

όπου x(t), είναι ένα παρατηρήσιμο διάνυσμα κατάστασης ή ένα καθυστερημένο διάνυσμα εξόδου, θ_f και θ_{gj} είναι ελεύθερες παράμετροι, και το d(t) αναπαριστά μια περιορισμένη διαταραχή. Η πραγματικότητα είναι ότι το διάνυσμα κατάστασης στην γραμμικοποιημένη με ανατροφοδότηση κανονική μορφή είναι συχνά σε μια μετασχηματισμένη μορφή και είναι συνήθως μη παρατηρήσιμο παρόλο που το διάνυσμα κατάστασης στην πρότυπη μορφή είναι παρατηρήσιμο. Γι’ αυτό ο μετασχηματισμός συντεταγμένων κατάστασης δεν είναι γνωστός a priori ή δεν μπορεί να κατασκευαστεί σωστά αν το πρότυπο σύστημα είναι άγνωστο ή αβέβαιο.

Υπάρχουν καλά αναπτυγμένες θεωρίες στο μετασχηματισμό συνεχούς χρόνου μη γραμμικών συγγενικών συστημάτων σε μια συγγενική συνηθισμένη (κανονική) μορφή με γραμμικοποίηση με ανατροφοδότηση κατάστασης (state – feedback linearization). Η περισσότερη εργασία και έρευνα στην γραμμικοποίηση με ανατροφοδότηση χρησιμοποιώντας νευρωνικά δίκτυα έχει εστιαστεί σε συστήματα σ’ αυτή τη συνηθισμένη μορφή ή στη διακριτού χρόνου μορφή. Πάντως, υπάρχει κάποια εργασία η οποία προσπαθεί να επεκτείνει τα αποτελέσματα σε πιο πολύπλοκα συστήματα (συνεχούς χρόνου). Το αυστηρό ανατροφοδοτούμενο σύστημα ή η παραμετρικά αγνή ανατροφοδοτούμενη μορφή είναι ένα από αυτά.

Το ακόλουθο συνεχούς χρόνου μίας εισόδου σύστημα θεωρείται:

$$
\begin{aligned}
\dot{z}_1 &= z_2 + r_1(z_1, z_2, \theta_1) \\
\dot{z}_2 &= z_3 + r_2(z_1, z_2, z_3, \theta_2) \\
&\vdots \\
\dot{z}_{n-1} &= z_n + r_{n-1}(z_1, \ldots, z_n, \theta_{n-1}) \\
\dot{z}_n &= r_n(z_1, \ldots, z_n, \theta_n) + r_{n+1}(z_1, \ldots, z_n, \theta_{n+1})u \\
y &= z_1
\end{aligned}
\tag{11}
$$

όπου $r_i(\cdot)$ αναπαριστούν τους ανατροφοδοτούμενους όρους και θ_i είναι ελεύθερες παράμετροι.

Όπως είναι φανερό εκτός απ' την γραμμικοποίηση με ανατροφοδότηση κατάστασης υπάρχει και η γραμμικοποίηση με ανατροφοδότηση εξόδου, η οποία χρησιμοποιεί προσαρμοστικούς αλγορίθμους ή γραμμικά στις παραμέτρους νευρωνικά δίκτυα. Σ' αυτήν την περίπτωση θεωρούμε συχνά τα ακόλουθα συνεχούς χρόνου μονής – εισόδου – μονής – εξόδου (single – input – single – output, SISO) συστήματα:

$$
\begin{aligned}
\dot{x} &= f(x) + q_0(x,u) + \sum_{i=1}^{p} \theta_i q_i(x,u) \\
y &= h(x)
\end{aligned}
\tag{12}
$$

Όλες οι συναρτήσεις στην (12) είναι ομαλές και γνωστές a priori και υπάρχει μόνο παραμετρική αβεβαιότητα για τα $\{\theta_i\}$. Αν κάποιες συνθήκες που επιβάλλονται στις $f(\cdot)$, $q_i(\cdot)$ και $h(\cdot)$ ικανοποιούνται με τη χρησιμοποίηση ενός μετασχηματισμού συντεταγμένων κατάστασης και ενός ανα/μένου νόμου ελέγχου μπορεί να μετασχηματιστεί στην ακόλουθη κανονική μορφή:

$$
\begin{aligned}
\dot{z} &= A_c z + \psi_0(y,u) + \sum_{i=1}^{p} \theta_i \psi_i(y,u) \\
y &= C_c z
\end{aligned}
\tag{13}
$$

όπου

$z=T(x)$	διάνυσμα κατάστασης στις μετα/μένες συντεταγμένες.
A_c και C_c	είναι σε κανονική μορφή
$\psi_i(\cdot)$	ομαλή συνάρτηση διανυσμάτων

Συγκρινόμενη με τη γραμμικοποίηση με ανατροφοδότηση κατάστασης ο ανατροφοδοτούμενος εξόδου όρος $\psi_0(y,u) + \sum_{i=1}^{p} \theta_i \psi_i(y,u)$ έχει κάποιες ειδικές ιδιότητες. Είναι μια συνάρτηση παρατηρούμενων εισόδων και εξόδων (u,y),

στην οποία οι συναρτήσεις βάσης είναι γνωστές a priori και υπάρχει αβεβαιότητα μόνο στις παραμέτρους $\{\theta_i\}$.

Έτσι η εφαρμογή των νευρωνικών δικτύων στη γραμμικοποίηση με ανατροφοδότηση περιορίζεται στην αναγνώριση των αβέβαιων παραμέτρων ή βαρών, θ (10) – (13) με σταθερές συναρτήσεις βάσης ή προεπιλεγμένες δομές δικτύου. Συγχρόνως όλα τα αποτελέσματα της γραμμικοποίησης με ανατροφοδότηση χρησιμοποιώντας νευρωνικά δίκτυα βασίζονται σε μεθόδους οι οποίες υποθέτουν τα ακόλουθα:

1) Το διάνυσμα κατάστασης στην κανονική μορφή είναι παρατηρήσιμο ή ο μετασχηματισμός συντεταγμένων κατάστασης μπορεί να κατασκευαστεί σωστά.

2) Οι συναρτήσεις βάσης στο γραμμικό στις παραμέτρους δίκτυο είναι γνωστές a priori ή η δομή του πολυεπίπεδου δικτύου είναι γνωστή ή προεπιλεγμένη, που σημαίνει ότι εκεί υπάρχει μόνο παραμετρική αβεβαιότητα.

Στην επόμενη παράγραφο αναπτύσσουμε μια αναδρομική νευροασαφής δομή και το δικό της σχήμα μάθησης βασισμένο στον MASMOD αλγόριθμο. Μπορεί να θεωρηθεί σαν μια μέθοδο για προσέγγιση της γραμμικοποίησης με ανατροφοδότηση ή νευροασαφή γραμμικοποίηση με ανατροφοδότηση με τις ακόλουθες θεωρήσεις:

1) Η πρωτότυπη εξίσωση εξόδου ή η διαδικασία μέτρησης είναι γραμμική (είναι λογικό να υποθέσουμε ότι οι αισθητήρες έχουν ρυθμιστεί να είναι γραμμικοί).

2) Ο μετασχηματισμός συντεταγμένων κατάστασης για να εξασφαλίσουμε την ανατροφοδοτούμενη κανονική μορφή είναι άγνωστος αλλά γραμμικός.

3) Μη γραμμικές συναρτήσεις είναι άγνωστες π.χ. και οι παράμετροι (weights) και οι συναρτήσεις βάσης πρέπει να αναγνωρισθούν.

1.2 Αναδρομικά νευροασαφή δίκτυα, δομή και το δικό τους σχήμα μάθησης

Όπως ήδη αναφέραμε η γραμμικοποίηση με ανατροφοδότηση υλοποιείται με έναν μετασχηματισμό συντεταγμένων κατάστασης και ένα ανατροφοδοτούμενο νόμο ελέγχου. Παρόλο που ο γραμμικός μετασχηματισμός συντεταγμένων κατάστασης είναι συνήθως άγνωστος a priori είναι δυνατόν να

αναγνωρίσουμε τον μετασχηματισμό με νευρωνικούς αλγορίθμους μάθησης. Με την υπόθεση ότι ο μετασχηματισμός συντεταγμένων κατάστασης είναι γραμμικός π.χ. $\dot{z} = Px$, όπου P αναπαριστά έναν μη μοναδιαίο πίνακα, η ανατροφοδοτούμενα γραμμικοποιημένη κανονική μορφή που περιγράφεται απ' τις εξισώσεις (4) και (5) μπορεί να ανασχηματιστεί (reformed) και αναπαρασταθεί σε μια διακριτού χρόνου μορφή όπως παρακάτω (με το διάνυσμα κατάστασης σε πρωτότυπες συντεταγμένες):

$$x(t+1)=Ax(t)+Br[Px(t),u(t),\alpha] \quad (14)$$

$$y(t)=Cx(t) \quad (15)$$

με $$r[Px(t),u(t),\alpha]=[r_1(x(t),u(t),\alpha_1),\ldots,r_L(x(t),u(t),\alpha_L)]^T \quad (16)$$

όπου A και B περιέχουν πληροφορίες σχετικά με τον μετασχηματισμό συντεταγμένων κατάστασης και είναι άγνωστες a priori, $r_l(\cdot)$ αναπαριστούν τους ανατροφοδοτούμενους όρους που περιέχουν και τις άγνωστες συναρτήσεις βάσης και τα άγνωστα διανύσματα παραμέτρων α_ρ. Η εξίσωση (14) μπορεί να θεωρηθεί σαν δομή γενικού αναδρομικού νευρωνικού δικτύου (recurrent neural network), η οποία είναι ικανή να προσεγγίσει μεγάλο μη γραμμικών δυναμικών συστημάτων και μπορεί να εκπαιδευτεί χρησιμοποιώντας ποικίλους αλγόριθμους μάθησης, όπως ο αναδρομικός back-propagation και ο back-propagation δια μέσω του χρόνου. Στην (14) βλέπουμε ότι το μη-γραμμικό σύστημα προσεγγίζεται από ένα προσοτροφοδοτούμενο γραμμικό μοντέλο και (+) ένα ανατροφοδοτούμενο μη-γραμμικό μοντέλο σφάλματος. Μπορούμε να εφαρμόσουμε κάποιες γραμμικές τεχνικές για εκτίμηση κατάστασης και έλεγχο στη γραμμικοποιημένη με ανατροφοδότηση μορφή (14) και (15) αν η μη γραμμικότητα στο σύστημα δεν είναι τόσο δυνατή (μεγάλη).

Πραγματικά, όλες η ανατροφοδοτούμενα γραμμικοποιημένες κανονικές μορφές μπορούν να θεωρηθούν ως ειδικές περιπτώσεις των RNN's. Οι μέχρι τώρα έρευνες για την προσέγγιση μη γραμμικών συναρτήσεων με νευρωνικά δίκτυα υποθέτουν ότι το πρωτότυπο σύστημα μπορεί να αναπαρασταθεί απευθείας στην κανονική μορφή χωρίς μετασχηματισμό συντεταγμένων κατάστασης. Έτσι μπορούμε να δούμε τις (14) και (15) σαν μια επέκταση των προηγουμένων αναπτυχθέντων εργασιών όσον αφορά τη γραμμικοποίηση με ανατροφοδότηση χρησιμοποιώντας νευρωνικά δίκτυα. Η επέκταση σύμφωνα με το φυλλάδιο περιλαμβάνει:

1) Οι πίνακες A, B και C δεν είναι απαραίτητα σε κανονική μορφή και αναγνωρίζονται μέσω της μάθησης.

2) Και η δομή του νευροασαφούς δικτύου και οι ελεύθερες παράμετροι είναι επίσης προσαρμοστικά αναγνωρισμένοι απ' τους αλγόριθμους μάθησης.

Θεωρώντας ότι η εξίσωση μέτρησης είναι γνωστή a priori εστιάζουμε στην αναγνώριση της (14) που μπορεί να ανασχηματιστεί ως εξής:

$$x_i(t+1) = \sum_{j=1}^{n} \alpha_{ij} x_j(t) + \sum_{l=1}^{L} b_{il} \cdot r_l[x(t), u(t), \alpha_l]$$

$$i=1,2,\ldots,n \qquad (17)$$

με τους ανατροφοδοτούμενους όρους να αναπαριστώνται με ένα ανάπτυγμα τοπικής συνάρτησης βάσης.

$$r_l[x(t), u(t), \alpha_l] = \alpha_{l0} + \sum_{\kappa=1}^{K} \alpha_{l\kappa} \cdot N_{l\kappa}[x(t), u(t)] \qquad (18)$$

όπου $N_{l\kappa}[x(t), u(t)]$ είναι B-spline συναρτήσεις βάσης και α_{lk} είναι ελεύθερες παράμετροι. Η (18) ανασυντίθεται στην ακόλουθη ANOVA μορφή:

$$r_l[x(t), u(t), \alpha_l] = \alpha_{l0} + \sum_{s=1}^{S} \sum_{\kappa=1}^{K_s} \alpha_{l\kappa}^{(s)} \cdot N_{l\kappa}^{(s)}[x_s(t), u_s(t)] \qquad (19)$$

όπου s είναι ένας όρος των υπομοντέλων, x_s και u_s αναπαριστούν υποδιαστήματα των x και u αντίστοιχα.

Η διαίρεση του διαστήματος εισόδου σε υπομοντέλα και η περαιτέρω διαίρεση του κάθε υπομοντέλου από B-spline δεσμούς (knots) μπορεί αυτόματα να εξαχθεί απ' τον MASMOD αλγόριθμο ο οποίος δείχνει μια ειδική προτίμηση σε μονομεταβλητά ή διμεταβλητά υπομοντέλα στην (19). Συνδυάζοντας τις (17) – (19) εξασφαλίζουμε:

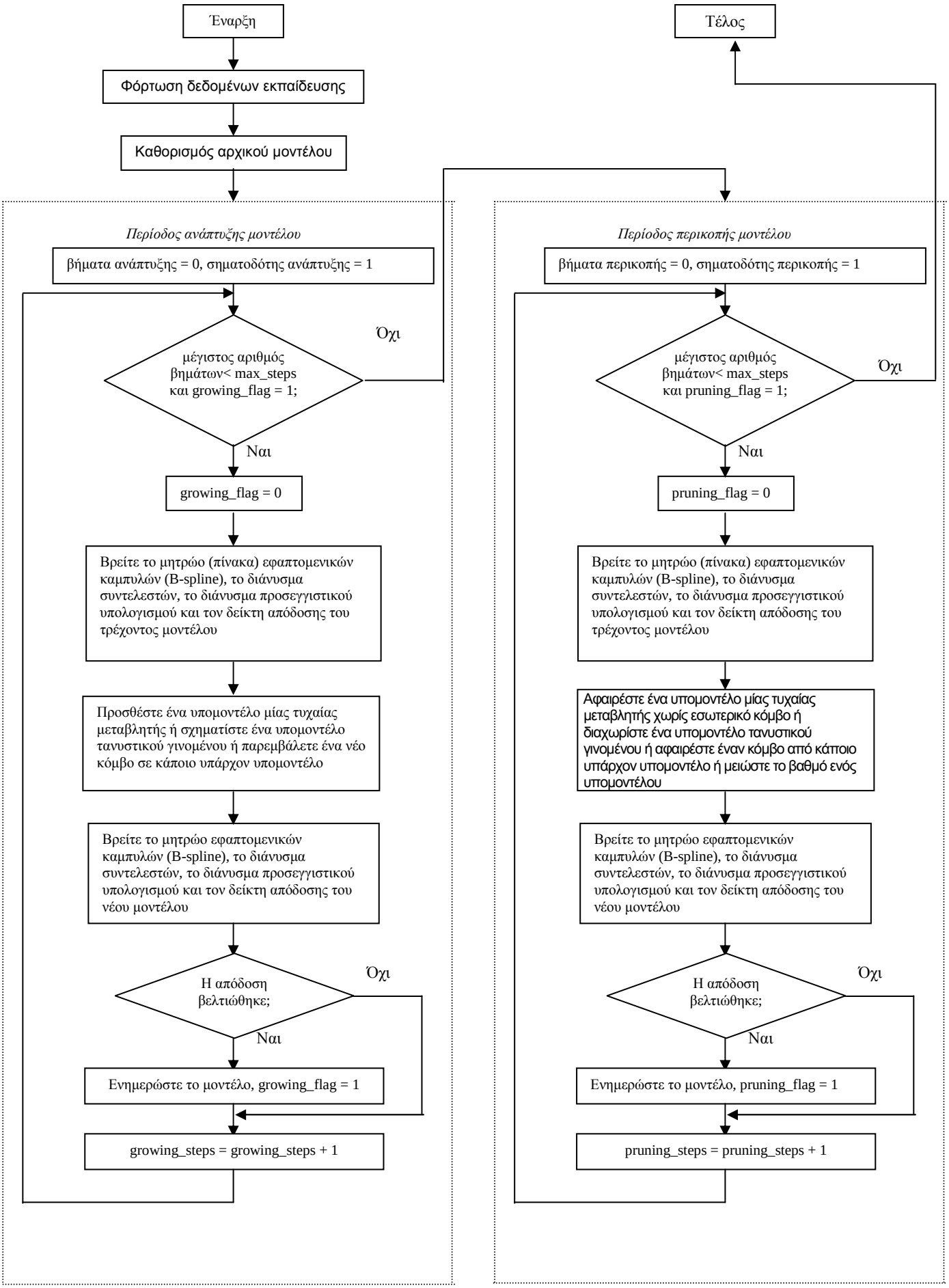
$$x_i(t+1) = \sum_{q=1}^{Q} \theta_{iq} \varphi_q[x(t), u(t)] = \theta_i^T \varphi[x(t), u(t)] = \theta_i^{(0)T} \varphi^{(0)}[x(t)] + \theta_i^{(1)T} \varphi^{(1)}[x(t), u(t)] \qquad (20)$$

όπου

$$\varphi[x(t), u(t)] = \varphi(t) = \left[\varphi^{(0)}(t); \varphi^{(1)}(t)\right] =$$
$$= [1, x_1(t), \ldots, x_n(t); N_1[x(t), u(t)], \ldots, N_{Q-n}[x(t), u(t)]]^T$$

μπορεί να θεωρηθεί σαν regressor διάνυσμα με $N_q[x(t), u(t)]$ να αντιστοιχούν σε μια ειδική B-spline συνάρτηση $N_{l\kappa}^{(s)}[x_s(t), u_s(t)]$ στο $s_=^o$ υπομοντέλο και $\theta_i = \left[\theta_i^{(0)}, \theta_i^{(1)}\right] = \left[\theta_{i0}, \theta_{i1}, \ldots, \theta_{in}; \theta_{i(n+1)}, \ldots, \theta_{iQ}\right]^T$ είναι ένα ελεύθερων παραμέτρων διάνυσμα. Πάντως η αναγνώριση της (14) αποδεικνύεται ότι είναι ένα

γενικευμένο πρόβλημα γραμμικής οπισθοδρόμισης. Σημειώστε ότι οι πρώτοι n+1 regressors είναι γραμμικοί και επιδιορθωμενοι και οι εναπομείναντες regressors είναι B-spline συναρτήσεις οι οποίες μπορούν να είναι αυτόματα κατασκευασμένοι χρησιμοποιώντας τον MASMOD αλγόριθμο ο οποίος παρουσιάζεται παρακάτω σαν block διάγραμμα.



Εικ.1 Διάγραμμα ροής 1 του αλγορίθμου MASMOD

Εικ. 2 Διάγραμμα ροής 2 του αλγορίθμου MASMOD

Ο MASMOD αλγόριθμος υπάρχει στο τέλος της διπλωματικής σαν κώδικας MATLAB όπως αυτός έχει υλοποιηθεί απ’ τον Qiang Gan. Εδώ πρέπει να σημειώσουμε ότι ο MASMOD αλγόριθμος είναι μια τροποποίηση του ASMOD αλγορίθμου: Modified, Algorithm, Adaptive, Spline, Modeling, Observation DATA. Ο τροποποιημένος ASMOD (MASMOD) αλγόριθμος μπορεί να περιγραφεί απ’ τα ακόλουθα βήματα:

1) Αρχικοποιούμε το μοντέλο, π.χ., τοποθετούμε ένα άδειο μοντέλο, τοποθετούμε την αρχική σειρά των B-spline και τις αρχικές τιμές των ελεύθερων παραμέτρων.

2) Ενημερώνουμε το διάνυσμα παραμέτρων $\alpha_0^{(s)}$ και τη μήτρα $A^{(s)}$ χρησιμοποιώντας τον αλγόριθμο των ελαχίστων τετραγώνων. Τα $\alpha_0^{(s)}$ και $A^{(s)}$ ορίζονται παρακάτω.

3) Υπολογίζουμε το rms σφάλμα (root-mean-square error) και τον όρο SRM (structural risk minimization – περιγράφεται παρακάτω). Αν τα κριτήρια ικανοποιούνται, πηγαίνετε στο βήμα 2 για περαιτέρω εκπαίδευση ή έξοδος.

4) Εξευγένισε (refine) το μοντέλο με την κατασκευή B-splines. Η εξευγένιση περιλαμβάνει πρόσθεση νέων μονομεταβλητών υπομοντέλων, είσοδο νέων δεσμών (knots) στα υπάρχοντα υπομοντέλα, κλάδεμα δεσμών ή υπομοντέλων.

5) Πήγαινε στο βήμα 2.

Μια περιληπτική ανάλυση της μεθόδου προσέγγισης μιας μη γραμμικής συνάρτησης με τη χρήση του MASMOD αλγορίθμου είναι η εξής:

Έστω f(x) η άγνωστη συνάρτηση και $\hat{f}(x)$ η προσέγγιση αυτής. Τότε:

$$\hat{f}(x) = f_0 + \sum_{s=1}^{S} f_s(x_s) \quad \text{με} \qquad f_s(x'_s) = \sum_{\kappa=1}^{Ps} N_\kappa^{(s)}(x_s)(\alpha_{\kappa 0}^{(s)} + \alpha_\kappa^{(s)} x_s) =$$

$$= \left(\alpha_0^{(s)} + A^{(s)} x_s\right)^T N^{(s)}(x_s)$$

όπου s είναι ο όρος των B-spline υπομοντέλων, $x_s=[x_{s1},x_{s2},\ldots,x_{sns}]^T$ αναπαριστά τα χαμηλής διάστασης υπομοντέλα του διαστήματος εισόδου x, $\alpha_0^{(s)} = \left[\alpha_{10}^{(s)}, \alpha_{20}^{(s)}, \ldots, \alpha_{ps0}\right]^T$ και $A^{(s)}\left[\alpha_{\kappa i}^{(s)}\right]_{Psxns}$ είναι τοπικά μοντέλα παραμέτρων και $N^{(s)}(x_s) = \left[N_1^{(s)}(x_s), N_2^{(s)}(x_s), \ldots, N_{Ps}^{(s)}(x_s)\right]^T$ είναι B-splines σαν συναρτήσεις συμμετοχής των ασαφών συνόλων διαιρώντας το υποδιάστημα. Διευκρινίζουμε ότι: (n) είναι η διάσταση της προσεγγισμένης συνάρτησης, (s) ο αριθμός των υπομοντέλων που απαιτούνται P_s βαθμός ή τάξη των B-spline συναρτήσεων και ως εκ τούτου n_s είναι η $n^{οστή}$ διάσταση του $s^{ου}$ υπομοντέλου.

Αυτό που κάνει ο MASMOD αλγόριθμος είναι να αυξάνει σε πρώτη φάση τον αριθμό των υπομοντέλων (s) ή τον αριθμό των υπομοντέλων (s) ή τον αριθμό των δεσμών (knots) των B-spline συναρτήσεων ώστε η προσέγγιση της

f(x) να ικανοποιεί διάφορα κριτήρια (rms, SRM, dof) και στη συνέχεια να αφαιρεί υπομοντέλα ή δεσμούς αν αυτά είναι περιττά.

Το SRM (structural risk minimization) είναι μια συνάρτηση του rms (root-mean-square) των σφαλμάτων εκτίμησης μοντέλου. Το dof (degree of freedom) αναπαριστά τους βαθμούς ελευθερίας των τοπικά γραμμικοποιημένων μοντέλων και M είναι ο αριθμός των δειγμάτων εκπαίδευσης. Ο όρος SRM ορίζεται ως εξής:

$$\mathrm{PN} = \mathrm{K}_1\sqrt{(\mathrm{dof}.\log(2\mathrm{M}) - \log(\mathrm{dof}) - \mathrm{K}_2)/\mathrm{M}}$$

$$\mathrm{SRM} = \begin{cases} \dfrac{\mathrm{rms}}{\sqrt{1-\mathrm{PN}}}, & \text{αν } \mathrm{PN} < 1 \\ \infty \quad\quad , & \text{αν αλλιώς} \end{cases}$$

όπου $\mathrm{rms} = \sqrt{\sum_{m=1}^{M}\left[y(m) - \hat{f}(x(m))\right]^2 / M}$

Ο αριθμός των υπομοντέλων, ο αριθμός των B-splines που απαιτούνται σε κάθε υπομοντέλο και οι θέσεις και τα σχήματα (B-spline σειρά) είναι όπως αναφέραμε αυτόματα καθορισμένα απ' τον αλγόριθμο. Η κατασκευαστική διαδικασία ξεκινάει με το βασικό διάνυσμα παλινδρόμησης (regressor vector) $\varphi^{(0)}[x(t)] = [1, x_1(t), \ldots, x_n(t)]^T$ οι τιμές των παραμετρικών διανυσμάτων $\theta_i^{(0)} = [\theta_{i0}, \theta_{i1}, \ldots, \theta_{in}]^T$ εξασφαλίζονται από έναν αλγόριθμο ελαχίστων τετραγώνων. Στη συνέχεια ο MASMOD αλγόριθμος πραγματοποιεί τη διύλιση (refinement) του μοντέλου βασισμένη στο κριτήριο της δομικής ελαχιστοποίησης ρίσκου (SRM): προσθέτοντας νέες μονομεταβλητές B-splines ή κλαδεύοντας δεσμούς (knots) και B-splines αν αυτοί είναι περιττοί.

Μέχρι εδώ αναλύσαμε διεξοδικά την τεχνική την οποία θα χρησιμοποιήσουμε για την αναγνώριση και γραμμικοποίηση του πλοίου όπως αυτή αναπτύχθηκε απ' τους Qiang Gan και Chris J. Harris. Παρακάτω αναλύουμε την εφαρμογή της μεθόδου για το μοντέλο του πλοίου.

2. Εξίσωση πλοίου

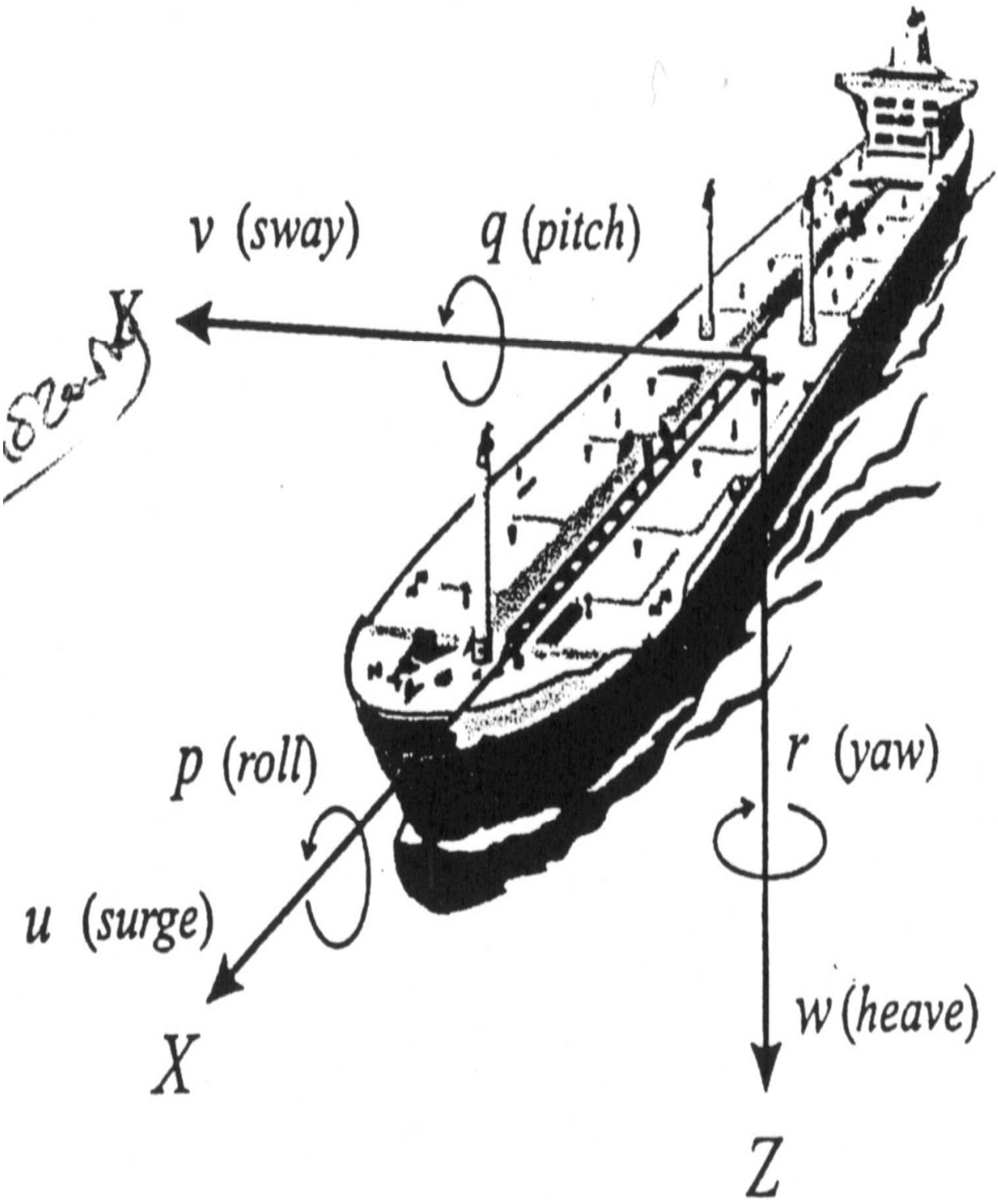
Σ' αυτή την παράγραφο θα αναπτύξουμε την εξίσωση του πλοίου. Πρέπει να τονίσουμε ότι αυτή την εξίσωση τη χρησιμοποιούμε για να πάρουμε τα ζευγάρια δεδομένων (data pairs) τα οποία είναι απαραίτητα για την εξαγωγή αποτελέσματος απ' τον MASMOD αλγόριθμο. Αυτό είναι λογικό αν σκεφτούμε ότι παρόλο που το σύστημα του πλοίου είναι άγνωστο μπορούμε να πάρουμε από ένα πραγματικό πλοίο δεδομένα εισόδου – εξόδου π.χ. να ασκήσουμε στο πλοίο μια συγκεκριμένη δύναμη και να πάρουμε την ταχύτητά του. Δηλαδή η εξίσωση του πλοίου παραμένει άγνωστη, μη γραμμική και δυναμική και μπορούμε να εφαρμόσουμε την αναπτυχθέντα τεχνική χρησιμοποιώντας τα ζευγάρια εισόδου – εξόδου.

Η δυναμική και κινηματική εξίσωση του πλοίου δίνεται παρακάτω˝

$$\dot{M}v + C(v)v + D(v)v + g(n) = \begin{bmatrix} \tau \\ o \end{bmatrix}$$

$$\dot{n} = J(n)v$$

όπου $n \in \mathbb{R}^{n_1}$, $v \in \mathbb{R}^{n_2}$ με $n_1 \geq n_2$. Το διάνυσμα των εισόδων ελέγχου είναι $\tau \in \mathbb{R}^m$ ($m<n_2$). Ο πίνακας M, ο οποίος είναι ο πίνακας της αδράνειας συμπεριλαμβανομένης και της προστιθέμενης υδροδυναμικής αδράνειας, είναι σταθερός, συμμετρικός, μη μοναδικός και θετικά ορισμένος. Η συμμετρική ιδιότητα βασίζεται στις υποθέσεις χαμηλής ταχύτητας, ιδανικής ρευστότητας και απουσίας κυμάτων. Ο πίνας C(v) ονομάζεται Coriolis κεντρομόλος πίνακας, ο οποίος περιέχει προστιθέμενες αδρανειακές επιδράσεις και είναι λοξά συμμετρικός (skew symmetric). Η υδροδυναμική του υγρού μπορεί να είναι γραμμική στην ταχύτητα v, δίνοντας μια σταθερή μήτρα D, ή μπορεί να περιέχει όρους που είναι υψηλής τάξης συναρτήσεις της ταχύτητας. Η κινηματική μετασχηματιστική μήτρα J έχει πλήρη βαθμό (rank) (rank (J)=n_2) εκτός απ' το σκαμπανέβασμα της γωνίας ±90 deg. Το διάνυσμα g(n) είναι το διάνυσμα έλξης και των δυνάμεων πλεύσης. Το διάνυσμα $v=[u,v,w,p,q,r]^T$ δηλώνει τις γραμμικές ταχύτητες του κύματος, ταλάντωσης και ανύψωσης (surge, sway, και heave αντίστοιχα) και τις γωνιακές ταχύτητες κύλισης (roll), σκαμπανεβάσματος(pitch)και περιστροφής όπως φαίνονται στην παρακάτω εικόνα.



Εικόνα πλοίου

Το διάνυσμα (n) δηλώνει τη θέση και τον προσανατολισμό ανασυνθετημένες στο σύστημα συντεταγμένων της γης. Το (τ) δηλώνει τις δυνάμεις ελέγχου. Χωρίς να χάσουμε τη γενικότητα υποθέτουμε ότι η θέση και ο προσανατολισμός του πλοίου είναι στο πρωτότυπο σύστημα συντεταγμένων της γης.

Θεωρώντας:

1) Οι τιμές των ταχυτήτων ανύψωσης (heave) σκαμπανεβάσματος (pitch) και κύλισης (roll) είναι αμελητέες δηλαδή z=φ=θ=0. Όπου z είναι η μεταβλητή θέσης της ανύψωσης, φ είναι η γωνία κύλισης και θ η γωνία σκαμπανεβάσματος. Η παραπάνω υπόθεση είναι πολύ καλά γνωστή και χρησιμοποιείται σε όλα τα βιομηχανικά συστήματα ελέγχου πλοίου. Επιπρόσθετα, τα συμβατικά πλοία δεν είναι εξοπλισμένα με ενεργοποιητές (actuators) που να επιδρούν στη δυναμική των heave, roll και pitch. Οπότε το διάνυσμα $n=[x,y,\psi]^T$ δηλώνει τη θέση και τον προσανατολισμό (ψ) του πλοίου στο σύστημα συντεταγμένων της γης. Το διάνυσμα $v=[u,\upsilon,r]^T$ δηλώνει τις γραμμικές ταχύτητες κύματος, ταλάντωσης και τη γωνιακή ταχύτητα περιστροφής (yaw) αναλυμένες στο σύστημα συντεταγμένων του πλοίου. Το διάνυσμα $\tau=(\tau_1,0,\tau_3]^T$ δηλώνει τις εισόδους ελέγχου (control inputs), τ_1 είναι η δύναμη του κύματος και τ_3 είναι η ροπή περιστροφής (yaw).

2) Οι πίνακες M και D είναι διαγώνιες.

$m_{23}=m_{32}\neq 0$ και $d_{23}\neq d_{32}\neq 0$

Το 3 DOF (τρεις βαθμοί ελευθερίας) μοντέλο πλοίου είναι:

$$M\dot{v}+C(v)v+D(v)v=\begin{bmatrix}\tau_1\\0\\\tau_3\end{bmatrix}$$

$$\dot{n}=J(\psi)v$$

όπου

$$M=\mathrm{diag}\{m_{11},m_{22},m_{33}\}$$

$$C(v)=\begin{bmatrix}0 & 0 & -m_{22}\upsilon\\ 0 & 0 & m_{11}u\\ m_{22}\upsilon & -m_{11}u & 0\end{bmatrix}$$

$$D(v)=\mathrm{diag}\left\{d_{11}+\sum_{i\geq 2}du^i|u|^{i-1},d_{22}+\sum_{i\geq 2}d\upsilon^i|\upsilon|^{i-1},d_{33}+\sum_{i\geq 2}dr^i|r|^{i-1}\right\}$$

$$J(\psi)=\begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}$$

3. Υπολογισμός δεδομένων (data pairs)

Οι εξισώσεις κατάστασης του πλοίου όπως προκύπτουν απ' την παραπάνω ανάλυση του συστήματος του πλοίου είναι:

$$\dot{u}=\frac{m_{22}}{m_{11}}\upsilon r-\frac{d_{11}}{m_{11}}u-\sum_{i\geq 2}\frac{du^i}{m_{11}}u|u|^{i-1}+\frac{1}{m_{11}}\tau_1$$

$$\dot{\upsilon}=\frac{m_{11}}{m_{22}}ur-\frac{d_{22}}{m_{22}}\upsilon-\sum_{i\geq 2}\frac{d\upsilon^i}{m_{22}}\upsilon|\upsilon|^{i-1}$$

$$\dot{r}=\frac{m_{11}-m_{22}}{m_{33}}u\upsilon-\frac{d_{33}}{m_{33}}r-\sum_{i\geq 2}\frac{dr^i}{m_{33}}r|r|^{i-1}+\frac{1}{m_{33}}\tau_3$$

Σύμφωνα με την τεχνική που αναπτύχθηκε προηγουμένως ότι η εξίσωση μέτρησης, δηλαδή η $\dot{n}=\zeta(\psi)v$, είναι γνωστή a priori οπότε εστιάζουμε στην αναγνώριση των τριών παραπάνω εξισώσεων.

Πριν προχωρήσουμε διακριτοποιούμε τις παραπάνω εξισώσεις σύμφωνα με την εξίσωση:

$$\dot{u}=\frac{u(\kappa)-u(\kappa-1)}{T}=\frac{m_{22}}{m_{11}}\upsilon(\kappa-1)r(\kappa-1)\ldots$$

$$\dot{\upsilon}=\frac{\upsilon(\kappa)-\upsilon(\kappa-1)}{T}=\ldots$$

Τα αποτελέσματα της διακριτοποίησης φαίνονται στο παρακάτω πρόγραμμα που έχει γραφτεί σαν κώδικας MATLAB. Σ' αυτό το πρόγραμμα έχουμε βάλει τυχαίες τιμές στις σταθερές m_{22}, m_{11}, d_{11}, d_{22}, d_{33}, m_{33} που αντιστοιχούν σε ένα δεδομένο τύπο πλοίου. Αυτές μπορούν να διαφοροποιηθούν για άλλους τύπους πλοίων που ενδέχεται να ενδιαφέρουν τους αναγνώστες.

ΠΡΟΓΡΑΜΜΑ MATLAB

```
%This program computed training data for RNFN MASMOD algorithm.

m11=10;
m22=8;
m33=4;
d11=5;
d22=10;
```

```
d33=15;
du2=3;
du3=5;
dr2=1;
dr3=2;
dv2=6;
dv3=4;

T=.08;
tfinal=30;

u=[];
v=[];
r=[];
t1=[];
t3=[];

u(1)=0;
v(1)=0;
r(1)=0;
t1(1)=2;
t3(1)=4;

for k=2:(tfinal/T)

  t1(k)=2*sin(T*k);
  t3(k)=4*sin(T*k);

 % t1(k)=2;
  %t3(k)=4;

 u(k)=u(k-1)+T*((m22/m11)*v(k-1)*r(k-1)-(d11/m11)*u(k-1)-
(du2/m11)*u(k-1)*abs(u(k-1))-(du3/m11)*(u(k-1)^3)+(1/m11)*t1(k-1));
   v(k)=v(k-1)+T*(-(m11/m22)*u(k-1)*r(k-1)-(d22/m22)*v(k-1)-
(dv2/m22)*v(k-1)*abs(v(k-1))-(dv3/m22)*(v(k-1)^3));
   r(k)=r(k-1)+T*(((m11-m22)/m33)*u(k-1)*v(k-1)-(d33/m33)*r(k-1)-
(dr2/m33)*r(k-1)*abs(r(k-1))-(dr3/m33)*(r(k-1)^3)+(1/m33)*t3(k-1));

end

figure(1)
 plot(u)
 figure(2)
 plot(v)
 figure(3)
 plot(r)
% figure(4)
% plot(t1)

%xTest=4*rand(400,1);
%yTest=u(k-1)+T*((m22/m11)*v(k-1)*r(k-1)-(d11/m11)*u(k-1)-
(du2/m11)*u(k-1)*abs(u(k-1))-(du3/m11)*(u(k-1)^3)+(1/m11)*xTest);

xTrain=[1:(tfinal/(4*T));t1(1:93);t3(1:93)]';
%yTrain=[u(1:93);v(1:93);r(1:93)]';
yTrain=[u(1:93)]';
xTest=[(tfinal/(4*T)):(tfinal/(2*T));t1(94:187);t3(94:187)]';
%yTest=[u(94:187);v(94:187);r(94:187)]';
yTest=[u(94:187)]';


degree=1;
```

```
resolution=0;
initSubmodels=[];
xmin=[];xmax=[];

initializeAsmod(xTrain,degree,resolution,xmin,xmax,initSubmodels);
[traces,est]=train(xTrain,yTrain,xTest,yTest,'SRM',15);
[rmse,nrmse,ev,mae,maxe,est]=testModel(xTest,yTest);

print -deps masmod_1d.eps;
plot(xTest,yTest,'.',xTest,est,'+');
drawnow;
print -deps masmod_1d_est.eps;
fname='masmod_1d_struct';
saveModel(fname);
```

4. Εφαρμογή του MASMOD αλγορίθμου για την εύρεση των $u(\tau_1(\kappa))$, $v(\kappa)$, $r(\tau_3(\kappa))$.

Πριν προχωρήσουμε στην ανάπτυξη του προγράμματος ώστε να βρούμε τις άγνωστες συναρτήσεις $u(\tau_1(\kappa))$, $v(\kappa)$, $r(\tau_3(\kappa))$ πρέπει να επισημάνουμε ότι από το προηγούμενο πρόγραμμα παίρνουμε τιμές των διακριτών συναρτήσεων u, v, r για διάφορες τιμές εισόδων $\tau_1(\kappa)$, κ, $\tau_3(\kappa)$ δηλαδή παίρνουμε ζευγάρια εισόδων – εξόδων. Παρακάτω φαίνεται το πρόγραμμα για την εύρεση αυτών των συναρτήσεων. Για να βγάλουμε ορθά αποτελέσματα χρησιμοποιήσαμε ποικιλόμορφα δεδομένα $\left(t_1(\kappa) = 2\sin(T\kappa) t_3(\kappa) = 4\sin(T * \kappa)\right)$. Επίσης γράψαμε τρία ξεχωριστά προγράμματα για τις 3 συναρτήσεις u, v, r. Τα προγράμματα αυτά φαίνονται παρακάτω:

ΠΡΟΓΡΑΜΜΑΤΑ

```
%This program computed training data for RNFN MASMOD algorithm.

m11=10;
m22=8;
m33=4;
d11=5;
d22=10;
d33=15;
du2=3;
du3=5;
dr2=1;
dr3=2;
dv2=6;
dv3=4;

T=.08;
tfinal=30;

u=[];
v=[];
r=[];
```

```
t1=[];
t3=[];

u(1)=0;
v(1)=0;
r(1)=0;
t1(1)=2;
t3(1)=4;

for k=2:(tfinal/T)

  t1(k)=2*sin(T*k);
  t3(k)=4*sin(T*k);

 % t1(k)=2;
  %t3(k)=4;

 u(k)=u(k-1)+T*((m22/m11)*v(k-1)*r(k-1)-(d11/m11)*u(k-1)-
(du2/m11)*u(k-1)*abs(u(k-1))-(du3/m11)*(u(k-1)^3)+(1/m11)*t1(k-1));
   v(k)=v(k-1)+T*(-(m11/m22)*u(k-1)*r(k-1)-(d22/m22)*v(k-1)-
(dv2/m22)*v(k-1)*abs(v(k-1))-(dv3/m22)*(v(k-1)^3));
   r(k)=r(k-1)+T*(((m11-m22)/m33)*u(k-1)*v(k-1)-(d33/m33)*r(k-1)-
(dr2/m33)*r(k-1)*abs(r(k-1))-(dr3/m33)*(r(k-1)^3)+(1/m33)*t3(k-1));

end

%figure(1)
% plot(u)
% figure(2)
% plot(v)
% figure(3)
% plot(r)
% figure(4)
% plot(t1)

%xTest=4*rand(400,1);
%yTest=u(k-1)+T*((m22/m11)*v(k-1)*r(k-1)-(d11/m11)*u(k-1)-
(du2/m11)*u(k-1)*abs(u(k-1))-(du3/m11)*(u(k-1)^3)+(1/m11)*xTest);

xTrain=[1:(tfinal/(4*T));t1(1:93);t3(1:93)]';
%yTrain=[u(1:93);v(1:93);r(1:93)]';
yTrain=[u(1:93)]';
xTest=[(tfinal/(4*T)):(tfinal/(2*T));t1(94:187);t3(94:187)]';
%yTest=[u(94:187);v(94:187);r(94:187)]';
yTest=[u(94:187)]';


degree=3;

resolution=0;
initSubmodels=[];
xmin=[];xmax=[];

initializeAsmod(xTrain,degree,resolution,xmin,xmax,initSubmodels);
[traces,est]=train(xTrain,yTrain,xTest,yTest,'SRM',15);
[rmse,nrmse,ev,mae,maxe,est]=testModel(xTest,yTest);

print -deps masmod_1d.eps;
plot(xTest,yTest,'.',xTest,est,'+');
drawnow;
print -deps masmod_1d_est.eps;
fname='masmod_1d_struct';
saveModel(fname);
```

```
%This program computed training data for RNFN MASMOD algorithm.

m11=10;
m22=8;
m33=4;
d11=5;
d22=10;
d33=15;
du2=3;
du3=5;
dr2=1;
dr3=2;
dv2=6;
dv3=4;

T=.08;
tfinal=30;

u=[];
v=[];
r=[];
t1=[];
t3=[];

u(1)=0;
v(1)=0;
r(1)=0;
t1(1)=2;
t3(1)=4;

for k=2:(tfinal/T)

  t1(k)=2*sin(T*k);
  t3(k)=4*sin(T*k);

 % t1(k)=2;
  %t3(k)=4;

 u(k)=u(k-1)+T*((m22/m11)*v(k-1)*r(k-1)-(d11/m11)*u(k-1)-
(du2/m11)*u(k-1)*abs(u(k-1))-(du3/m11)*(u(k-1)^3)+(1/m11)*t1(k-1));
   v(k)=v(k-1)+T*(-(m11/m22)*u(k-1)*r(k-1)-(d22/m22)*v(k-1)-
(dv2/m22)*v(k-1)*abs(v(k-1))-(dv3/m22)*(v(k-1)^3));
   r(k)=r(k-1)+T*(((m11-m22)/m33)*u(k-1)*v(k-1)-(d33/m33)*r(k-1)-
(dr2/m33)*r(k-1)*abs(r(k-1))-(dr3/m33)*(r(k-1)^3)+(1/m33)*t3(k-1));

end

%figure(1)
% plot(u)
% figure(2)
% plot(v)
% figure(3)
% plot(r)
% figure(4)
% plot(t1)

%xTest=4*rand(400,1);
%yTest=u(k-1)+T*((m22/m11)*v(k-1)*r(k-1)-(d11/m11)*u(k-1)-
(du2/m11)*u(k-1)*abs(u(k-1))-(du3/m11)*(u(k-1)^3)+(1/m11)*xTest);

xTrain=[1:(tfinal/(4*T));t1(1:93);t3(1:93)]';
%yTrain=[u(1:93);v(1:93);r(1:93)]';
yTrain=[v(1:93)]';
```

```
xTest=[(tfinal/(4*T)):(tfinal/(2*T));t1(94:187);t3(94:187)]';
%yTest=[u(94:187);v(94:187);r(94:187)]';
yTest=[v(94:187)]';


degree=3;

resolution=0;
initSubmodels=[];
xmin=[];xmax=[];

initializeAsmod(xTrain,degree,resolution,xmin,xmax,initSubmodels);
[traces,est]=train(xTrain,yTrain,xTest,yTest,'SRM',15);
[rmse,nrmse,ev,mae,maxe,est]=testModel(xTest,yTest);

print -deps masmod_1d.eps;
plot(xTest,yTest,'.',xTest,est,'+');
drawnow;
print -deps masmod_1d_est.eps;
fname='masmod_1d_struct';
                           saveModel(fname);


%This program computed training data for RNFN MASMOD algorithm.

m11=10;
m22=8;
m33=4;
d11=5;
d22=10;
d33=15;
du2=3;
du3=5;
dr2=1;
dr3=2;
dv2=6;
dv3=4;

T=.08;
tfinal=30;

u=[];
v=[];
r=[];
t1=[];
t3=[];

u(1)=0;
v(1)=0;
r(1)=0;
t1(1)=2;
t3(1)=4;

for k=2:(tfinal/T)

  t1(k)=2*sin(T*k);
  t3(k)=4*sin(T*k);

 % t1(k)=2;
  %t3(k)=4;

 u(k)=u(k-1)+T*((m22/m11)*v(k-1)*r(k-1)-(d11/m11)*u(k-1)-
(du2/m11)*u(k-1)*abs(u(k-1))-(du3/m11)*(u(k-1)^3)+(1/m11)*t1(k-1));
```

```
   v(k)=v(k-1)+T*(-(m11/m22)*u(k-1)*r(k-1)-(d22/m22)*v(k-1)-
(dv2/m22)*v(k-1)*abs(v(k-1))-(dv3/m22)*(v(k-1)^3));
   r(k)=r(k-1)+T*(((m11-m22)/m33)*u(k-1)*v(k-1)-(d33/m33)*r(k-1)-
(dr2/m33)*r(k-1)*abs(r(k-1))-(dr3/m33)*(r(k-1)^3)+(1/m33)*t3(k-1));

end

%figure(1)
% plot(u)
% figure(2)
% plot(v)
% figure(3)
% plot(r)
% figure(4)
% plot(t1)

%xTest=4*rand(400,1);
%yTest=u(k-1)+T*((m22/m11)*v(k-1)*r(k-1)-(d11/m11)*u(k-1)-
(du2/m11)*u(k-1)*abs(u(k-1))-(du3/m11)*(u(k-1)^3)+(1/m11)*xTest);

xTrain=[1:(tfinal/(4*T));t1(1:93);t3(1:93)]';
%yTrain=[u(1:93);v(1:93);r(1:93)]';
yTrain=[r(1:93)]';
xTest=[(tfinal/(4*T)):(tfinal/(2*T));t1(94:187);t3(94:187)]';
%yTest=[u(94:187);v(94:187);r(94:187)]';
yTest=[r(94:187)]';


degree=3;

resolution=0;
initSubmodels=[];
xmin=[];xmax=[];

initializeAsmod(xTrain,degree,resolution,xmin,xmax,initSubmodels);
[traces,est]=train(xTrain,yTrain,xTest,yTest,'SRM',15);
[rmse,nrmse,ev,mae,maxe,est]=testModel(xTest,yTest);

print -deps masmod_1d.eps;
plot(xTest,yTest,'.',xTest,est,'+');
drawnow;
print -deps masmod_1d_est.eps;
fname='masmod_1d_struct';
saveModel(fname);
```

Τα μισά δεδομένα τα χρησιμοποιήσαμε για να ελέγξουμε (τεστάρουμε) το μοντέλο και τα άλλα μισά για εκπαίδευση του MASMOD αλγορίθμου. Στη συνέχεια ορίσαμε το βαθμό των B-spline συναρτήσεων (degree = 3) που θα χρησιμοποιήσει ο αλγόριθμος και τον αριθμό των εσωτερικών δεσμών (knots) (resolution = 0; Αρχικός αριθμός εσωτερικών knots). Με την επόμενη εντολή αρχικοποιούμε το μοντέλο σαν ένα άδειο μοντέλο (empty model): ini + sub models = [ ]; Με τις εντολές xmin=[ ]; xmax=[ ]; ορίζουμε το αρχικό διάστημα εισόδου το οποίο καθορίζεται αυτόματα απ' τον αλγόριθμο.

Τα αποτελέσματα των τριών προγραμμάτων φαίνονται παρακάτω:

try3.m

	Variable	Internal knots	Spline degree
Sub model:	1	1	3

[2 0 3]

try4.m

	Variable	Internal knots	Spline degree
Sub model:	1	2	3

[2 0 3]

try5.m

	Variable	Internal knots	Spline degree
Sub model:	1	2	3

[2 0 3]

5. Σχολιασμός αποτελεσμάτων

Όπως έχουμε ήδη αναφέρει το τ_1, κ, τ_3 είναι οι είσοδοι για τα u, v, r. Δηλαδή αν f(·) η συνάρτηση που συνδέει το u με το τ_1 τότε:

Αν $x=\tau_1$ $y=f(x)=u$

$$f(x) = f_0 + \sum_{s=1}^{S} f_s(x_s) \underline{\underline{s=1}} f_1(x_1) = f_1(\tau_1)$$

Παρατηρούμε ότι υπάρχει ένας σταθερός όρος (f_0) και οι παράμετροι α_{u0} στα τοπικά γραμμικά μοντέλα. Σύμφωνα με το φυλλάδιο, τίθενται και τα δύο ίσα με (0) μηδέν. Πριν προχωρήσουμε πρέπει να πούμε ότι:

n: είναι η διάσταση της προσεγγισμένης συνάρτησης (n=1)

s: είναι ο αριθμός των υπομοντέλων που απαιτούνται (s=1)

P_s: βαθμός των B-spline συναρτήσεων (3)

n_s: $n^{οστή}$ διάσταση του $s^{ου}$ υπομοντέλου (n_s=1)

Άρα

$$f_1(\tau_1) = \sum_{\kappa=1}^{Ps} N_{\kappa}^{(1)}(\tau_1)\left(\alpha_{u0}^{(1)} + \alpha_{\kappa}^{(1)T}\tau_1\right) = \left(\alpha_0^{(1)} + A^{(1)}\tau_1\right)^T N^1(\tau_1)$$

$$\alpha_0^{(1)} = \left[\alpha_{10} \;\; \alpha_{20} \;\; \alpha_{30}\right] = [0 \;\; 0 \;\; 0]$$

$$A^{(1)} = \left[\alpha_{\kappa i}^{(1)}\right]_{Psxns} = \left[\alpha_{ui}^{(1)}\right]_{3x1} = [2 \;\; 0 \;\; 3]$$

Συνεπώς:

$$u = f(x) = [[2 \quad 0 \quad 3]r_1][N_1(r_1), N_2(r_1), N_3(r_1)]^T$$

Ομοίως

$$v = [[2 \quad 0 \quad 3]k][N_1(k), N_2(k), N_3(k)]^T$$

$$r = [[2 \quad 0 \quad 3]r_3][N_1(r_3), N_2(r_3), N_3(r_3)]^T$$

Οι B-spline στην u έχουν (1) ένα εσωτερικό δεσμό (Knot) στην v και r (2) δύο εσωτερικούς Knots.

П А Р А Р Т Н М А

```
% computeBasis.m
% for fuzzy local linearisation model
```

```
function basis = computeBasis ( x, knotVectors, NoKnots, degrees,
indim)
%basis = computeBasis ( x, knotVectors, NoKnots, degrees, inDim);
%Computes a column vector of
%B-spline basis function values for the input row vector x.
%Modified by John Gan

basis1 = computeSplineBasis (x(1), knotVectors(:,1), NoKnots(1),
degrees(1) );

for i=2:indim
      basis1 = tmult (basis1, computeSplineBasis (x(i),
knotVectors(:,i), NoKnots(i), degrees(i)));
end

%basis1=basis1(2:size(basis1,2));
%To avoid linear dependency in basis functions.

basis1 = basis1';

%basis=basis1;
%for i=1:indim
%     basis=[basis;x(i)*basis1];
%end

basis=x(1)*basis1;
for i=2:indim
      basis=[basis;x(i)*basis1];
end
```

```
% computeB.m
```

```
function B = computeB ( x, knotVectors, NoKnots, degrees)
%B = computeB ( x, knotVectors, NoKnots, degrees)
%Computes a matrix (no_samples * no-coefficients) of
%B-spline basis function values for the input data set x.

[indim, nSamples] = size(x);
for n=nSamples:-1:1
      B(:,n) = computeBasis (x(:,n), knotVectors, NoKnots, degrees,
indim);
end
```

```
% computeSplineBasis.m

function bv = computeSplineBasis ( x, knots, nKnots, degree)
```

```
% bv = computeSplineBasis ( x, knots, nKnots degree);
% Computes a row vector of
% B-spline basis function values for the input x.

knots = knots(1:nKnots);
bv = zeros(1, nKnots - degree + 1);
if (x <= knots(degree))
            bv(1) = 1;
            return;
end

last = (find(knots > x));

if isempty(last)
            bv(length(bv)) = 1;
            'outside input range'
            return;
end

last = last(1);         % the last nonzero basis function

bv(last) = 1;

for r=0:degree-1
      i=last - r;
      for j=i:last
            w = (x - knots(j-1)) / (knots(j+r) - knots(j-1));
            bv(i-1) = bv(i-1) + (1 - w) * bv(i);
            bv(i) = w * bv(i);
      i=i+1;
      end
end
```

```
% clearModel.m
```

```
function clearModel()
% clearModel()
% Clears the model variables

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_
global initKnVectors_ initNoKnots_
global traces_

noSm_ = 0;
knVectors_ = [];
noKnots_=[];
degrees_ = [];
inDim_=[];
inVars_ = [];
c_ = [];
initKnVectors_ = [];
initNoKnots_ = [];
traces_ = [];
```

```
% findBestPruned.m
```

```
function [nsm,knV, noKn, inDim, inVars, c, bestCrit, degrees,
refineFlag,est]= ...
      findBestPruned(x, y, criterion);
% function [nsm,knV, noKn, inDim, inVars, c, bestCrit, degrees,
refineFlag]= ...
%     findBestPruned(x, y, criterion);
%
% Test all pruning candidate models for the current model,
% and return the best candidate measured by the given criterion
% refineFlag is set if a candidate better than the current model is
found
% criterion selects the optimization criterion to use (see train)

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_
global initKnVectors_ initNoKnots_

% assume that the current model is the best
nsm = noSm_;
knV = knVectors_;
noKn = noKnots_;
inDim = inDim_;
inVars = inVars_;
c = c_;
refineFlag = 0;
degrees = degrees_;

[bestC, est, B, Binv] = SVDFit(x, y, nsm, knV, noKn, inDim, inVars,
degrees_);
bestCrit = compCriteria(y, est, B, Binv);

% test removal of univariate linear submodels with no internal knots
for n = 1:noSm_
      if ( inDim_(n) == 1 )
            ind=findKVIndices(n, inDim_);
            if (degrees_(inVars_(n,1)) == 1 & noKnots_(ind) == 2)
                  cnsm = noSm_ - 1;
                  cknV = knVectors_;
                  cknV(:,ind) = [];
                  cnoKn = noKnots_;
                  cnoKn(ind) = [];
                  cinDim = inDim_;
                  cinDim(n) = [];
                  cinVars = inVars_;
                  cinVars(n,:) = [];
                  [cc, est, B, Binv] = SVDFit(x, y, cnsm, cknV, cnoKn,
cinDim, cinVars, degrees_);
                  cCrit = compCriteria(y, est, B, Binv);

      if(~isnan(cCrit(criterion))&(cCrit(criterion)<bestCrit(criterion
)))
                        bestCrit = cCrit;
                        nsm = cnsm;
                        knV = cknV;
                        noKn = cnoKn;
                        inDim = cinDim;
                        inVars = cinVars;
                        c = cc;
                        refineFlag = 1;
                  end
            end
      end
end
```

```
%disp('Test splitting of tensor submodels')
for i = 1:noSm_
      for nto = 1:inDim_(i) - 1
            for j = 1:inDim_(i) - nto  % number of variables taken out
in split
                  cnsm = noSm_ + 1;
                  cknV = knVectors_;
                  cnoKn = noKnots_;
                  cinDim = inDim_;
                  cinDim(i+1:cnsm) = cinDim(i:cnsm-1);
                  cinDim(i:i+1) = [nto,inDim_(i) - nto];
                  cinVars = inVars_;
                  cinVars(i+1:cnsm,:) = cinVars(i:cnsm-1,:); %
duplicate row
                  cinVars(i,1:nto) = cinVars(i,j:j+nto-1); % take nto
variables
                  ind = nto+1:size(cinVars,2);
                  cinVars(i,ind) = ind*0;
                  cinVars(i+1,1:inDim_(i) - nto) = cinVars(i+1,[1:j-
1,j+nto:inDim_(i)]);
                  ind = inDim_(i) - nto+1:size(cinVars,2);
                  cinVars(i+1,ind) = ind*0;
                  [cc, est, B, Binv] = SVDFit(x, y, cnsm, cknV, cnoKn,
cinDim, cinVars, degrees_);
                  cCrit = compCriteria(y, est, B, Binv);

      if(~isnan(cCrit(criterion))&(cCrit(criterion)<bestCrit(criterion
)))
                        bestCrit = cCrit;
                        nsm = cnsm;
                        knV = cknV;
                        noKn = cnoKn;
                        inDim = cinDim;
                        inVars = cinVars;
                        c = cc;
                        refineFlag = 1;
                  end
            end
      end
end

%disp('Test knot removal candidates')
for i = 1:noSm_
      indim = inDim_(i);
      invars = inVars_(i,1:indim);
      ind1 = findKVIndices(i, inDim_);
      knots = knVectors_(:,ind1);
      noknots = noKnots_(:,ind1);

      for j = 1:indim
            refKnVectors =
pruneKnotVector(knots(:,j),noknots(j),degrees_(invars(j)));
            [k, nrv] = size(refKnVectors);
            for k=1:nrv
                  knots = knVectors_(:,ind1);
                  noknots = noKnots_(:,ind1);
                  knots(1:size(refKnVectors,1),j) = refKnVectors(:,k);
                  noknots(j) = noknots(j) - 1;
                  [cnsm, cknV, cnoKn, cinDim, cinVars] = ...

      insertNewSubmodel(noSm_,knVectors_,noKnots_,inDim_,inVars_, ...
                        degrees_, indim, invars, knots, noknots);
```

```
                  [cc, est, B, Binv] = SVDFit(x, y, cnsm, cknV, cnoKn,
cinDim, cinVars, degrees_);
                  cCrit = compCriteria(y, est, B, Binv);

      if(~isnan(cCrit(criterion))&(cCrit(criterion)<bestCrit(criterion
)))
                        bestCrit = cCrit;
                        nsm = cnsm;
                        knV = cknV;
                        noKn = cnoKn;
                        inDim = cinDim;
                        inVars = cinVars;
                        c = cc;
                        refineFlag = 1;
                  end
            end
      end
end

%disp('Test degree reduction candidates')
for i = 1:noSm_
      indim = inDim_(i);
      invars = inVars_(i,1:indim);
      ind1 = findKVIndices(i, inDim_);
      knots = knVectors_(:,ind1);
      noknots = noKnots_(:,ind1);
      degs = degrees_;
      for j = 1:indim
            deg = degs(invars(j));
            if (deg > 1 & noknots(j) == 2*deg)
                  knots = knVectors_(:,ind1);
                  noknots = noKnots_(:,ind1);
                  deg = deg-1;
                  knots(1:2*deg,j) =
[knots(1:deg,j);knots(deg+3:deg+deg+2,j)];
                  degs(invars(j)) = degs(invars(j))-1;
                  noknots(j) = noknots(j) - 2;
                  [cnsm, cknV, cnoKn, cinDim, cinVars] = ...

      insertNewSubmodel(noSm_,knVectors_,noKnots_,inDim_,inVars_, ...
                        degs, indim, invars, knots, noknots);
                  [cc, est, B, Binv] = SVDFit(x, y, cnsm, cknV, cnoKn,
cinDim, cinVars, degs);
                  cCrit = compCriteria(y, est, B, Binv);

      if(~isnan(cCrit(criterion))&(cCrit(criterion)<bestCrit(criterion
)))
                        bestCrit = cCrit;
                        nsm = cnsm;
                        knV = cknV;
                        noKn = cnoKn;
                        inDim = cinDim;
                        inVars = cinVars;
                        degrees = degs;
                        c = cc;
                        refineFlag = 1;
                  end
            end
      end
end
```

```
% compCriteria.m

function criteria = compCriteria(y, est, B, Binv)
% criteria = compCriteria(y, est, xTest, yTest, B, Binv)
%
% compCriteria returns a row vector with the number of independent
% parameters and different optimisation criteria for the current model
% criteria = [RMS, CV, SRM, FPE, AIC, MDL, dof];
% RMS = rms of estimation errors for the training data
% CV = cross validation error with leaveout of single samples
% SRM = the structural risk minimisation criterion (Vapnik, 1982,
1989)
% FPE = Akaike's final prediction error criterion (Akaike, 1969)
% AIC = Akaike's information criterion (Akaike, 1974)
% MDL = the minimum description length criterion (Rissanen, 1978)
% dof = degrees of freedom computed as the trace of the smoothing
matrix
% (Hastie and Tibshirani: "Generalized additive models", 1993 )
%
% For FPE, AIC and MDL it is required that there are at least twice as
many training
% samples as there are parameters in the model. Otherwise NaN is
returned.
%
% y and est are vectors of target and estimated values
% xTest and yTest are test input vectors and target values (in rows)
% B and Binv are the matrices returned by the SVDFit function

res = y - est;

RMS = sqrt(res' * res / length(res));

Sdiag = zeros(size(B,1),1);
for i = 1:size(B,1)
      Sdiag(i) = B(i,:) * Binv(:,i);
end
Ones = find(Sdiag == 1);
if (Ones)
      Sdiag( Ones ) = Sdiag( Ones ) - 0.000001;
end

dof = sum(Sdiag);
nS = length(y);

if (dof < nS/2)
      FPE = RMS * sqrt((1+dof/nS)/(1-dof/nS));
else
      FPE = NaN;
end

if (dof < nS/2)
      AIC = log(RMS) + 2 * dof/nS;
else
      AIC = NaN;
end

if (dof < nS/2)
      MDL = log(RMS) + dof * log(nS) / nS;
else
      MDL = NaN;
end

T = .9*sqrt((dof*log(2*nS)-log(prod(1:dof))-log(.1/12))/nS);
```

```
if (T < 1)
      SRM = RMS/sqrt(1-T);
else
      SRM = NaN;
end

res = res ./ (1 - Sdiag);
CV = sqrt(res' * res / length(res));

criteria = [RMS, CV, SRM, FPE, AIC, MDL, 0, dof];
```

```
% initializeAsmod.m

function ok=initializeAsmod(x,degrees,resolutions,xmin,
xmax,initSubmodels)
% ok=initializeAsmod(x,degrees,resolutions,xmin, xmax,initSubmodels);
%
% Creates a new ASMOD model. The model will have input dimensionality
% equal to the number of columns in x. x is a matrices of
% input vectors (in rows) used for training.
%
% The model will for the input variables have the spline degrees,
% resulutions (number of internal knots in the initial submodels)
% and ranges [xmin, xmax] as given by the corresponding elements of
% the parameters.
%
% If any of these parameter are scalars or vectors shorter than the
% number of input variables, the first element is used for all
% variables. If ranges is empty, the input ranges are automatically
% set to the range of the data in x, pluss 10% added on both sides.
%
% The parameter initSubmodels, specifies a set of submodels to be
% created at initialization. One submodels is specified per matrix row
% with the dimensionality of the input vector in the first column,
% and the input variable numbers (columns in x) in the following
columns.

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_
global initKnVectors_ initNoKnots_

clearModel;

[i, nInputVars]=size(x);

if (length(degrees) >= nInputVars)
      degrees_ = degrees(1:nInputVars);
else
      degrees_ = ones(nInputVars,1) * degrees(1);
end
if length(resolutions) >= nInputVars
      resol = resolutions(1:nInputVars);
else
      resol = ones(nInputVars,1) * resolutions(1);
end

autoscale = 0;
if (isempty(xmin) | isempty(xmax) | length(xmin) ~= length(xmax))
      autoscale = 1;
elseif (length(xmin) < nInputVars | min(xmax - xmin) <=0 )
      autoscale = 1;
end

if (autoscale == 1 )
      xMin = min(x);
      xMax = max(x);
      xMax = xMax + (xMax - xMin) * .1;
      xMin = xMin - (xMax - xMin) * .1;
else
      xMin = xmin;
      xMax = xmax;
end

initKnVectors_=[];
for i=nInputVars:-1:1
```

```
      initKnVectors_(1:resol(i)+ 2 * degrees_(i),i)= ...
            [ones(degrees_(i)-1,1)*xMin(i); ...
            linspace(xMin(i),xMax(i),resol(i)+2)'; ...
            ones(degrees_(i)-1,1) * xMax(i)];
end
initNoKnots_ = resol + degrees_ + degrees_;

% Create the submodles for the initial model
[n, i] = size(initSubmodels);
for i = 1:n
      inDim = initSubmodels(i,1);
      inVars = initSubmodels(i,2:1+inDim);
      knots = initKnVectors_( :,inVars );
      noKnots = initNoKnots_( inVars );
      [noSm_, knVectors_, noKnots_, inDim_, inVars_] = ...
      insertNewSubmodel(noSm_, knVectors_, noKnots_, inDim_, inVars_,
...
       degrees_, inDim, inVars, knots, noKnots);
end
```

```
% findBestRefined.m

function [nsm,knV, noKn, inDim, inVars, c, bestCrit, ref,est]= ...
      findBestRefined(x, y, criterion);
%     function [nsm,knV, noKn, inDim, inVars, c, bestCrit, ref]= ...
%     findBestRefined(x, y, criterion);
% Test all candidate models for the current model,
% and return the best candidate measured by the given criterion
% criterion selects the optimization criterion to use (see train)

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_
global initKnVectors_ initNoKnots_

% assume that the current model is the best
nsm = noSm_;
knV = knVectors_;
noKn = noKnots_;
inDim = inDim_;
inVars = inVars_;
c = c_;
ref = 0;

[bestC, est, B, Binv] = SVDFit(x, y, nsm, knV, noKn, inDim, inVars,
degrees_);
bestCrit = compCriteria(y, est, B, Binv);

% test insertion of new univariate functions
for i = 1:length (degrees_)
      used = find (inVars_ == i);
      if (isempty(used))
            [cnsm, cknV, cnoKn, cinDim, cinVars] = ...

      insertNewSubmodel(noSm_,knVectors_,noKnots_,inDim_,inVars_, ...
                        degrees_, 1, i, initKnVectors_(:,i),
initNoKnots_(i));
            [cc, est, B, Binv] = SVDFit(x, y, cnsm, cknV, cnoKn,
cinDim, cinVars, degrees_);
            cCrit = compCriteria(y, est, B, Binv);
            if (~isnan ( cCrit(criterion) ) )
                  if (cCrit(criterion) < bestCrit(criterion))
                        bestCrit = cCrit;
                        nsm = cnsm;
                        knV = cknV;
                        noKn = cnoKn;
                        inDim = cinDim;
                        inVars = cinVars;
                        c = cc;
                        ref = 1;
                  end
            end
      end
end

%disp('Test tensor candidates')
for i = 1:noSm_
      for j = i+1:noSm_
            indim = inDim_(i) + inDim_(j);
            invars = [inVars_(i,1:inDim_(i)),inVars_(j,1:inDim_(j))];
            ind1 = findKVIndices(i, inDim_);
            ind2 = findKVIndices(j, inDim_);
            knots = [knVectors_(:,ind1),knVectors_(:,ind2)];
            noknots = [noKnots_(:,ind1),noKnots_(:,ind2)];
            [cnsm, cknV, cnoKn, cinDim, cinVars] = ...
```

```
      insertNewSubmodel(noSm_,knVectors_,noKnots_,inDim_,inVars_, ...
                        degrees_, indim, invars, knots, noknots);
            [cc, est, B, Binv] = SVDFit(x, y, cnsm, cknV, cnoKn,
cinDim, cinVars, degrees_);
            cCrit = compCriteria(y, est, B, Binv);
            if (~isnan ( cCrit(criterion) ) )
                  if (cCrit(criterion) < bestCrit(criterion))
                        bestCrit = cCrit;
                        nsm = cnsm;
                        knV = cknV;
                        noKn = cnoKn;
                        inDim = cinDim;
                        inVars = cinVars;
                        c = cc;
                        ref = 1;
                  end
            end
      end
end

%disp('Test knot insertion candidates')
for i = 1:noSm_
      indim = inDim_(i);
      invars = inVars_(i,1:indim);
      ind1 = findKVIndices(i, inDim_);
      knots = knVectors_(:,ind1);
      noknots = noKnots_(:,ind1);

      for j = 1:indim
            refKnVectors =
refineKnotVector(knots(:,j),noknots(j),degrees_(invars(j)));
            [k, nrv] = size(refKnVectors);
            for k=1:nrv
                  knots = knVectors_(:,ind1);
                  noknots = noKnots_(:,ind1);
                  knots(1:size(refKnVectors,1),j) = refKnVectors(:,k);
                  noknots(j) = noknots(j) +1;
                  [cnsm, cknV, cnoKn, cinDim, cinVars] = ...

      insertNewSubmodel(noSm_,knVectors_,noKnots_,inDim_,inVars_, ...
                        degrees_, indim, invars, knots, noknots);
                  [cc, est, B, Binv] = SVDFit(x, y, cnsm, cknV, cnoKn,
cinDim, cinVars, degrees_);
                  cCrit = compCriteria(y, est, B, Binv);
                  if (~isnan ( cCrit(criterion) ) )
                        if (cCrit(criterion) < bestCrit(criterion))
                              bestCrit = cCrit;
                              nsm = cnsm;
                              knV = cknV;
                              noKn = cnoKn;
                              inDim = cinDim;
                              inVars = cinVars;
                              c = cc;
                              ref = 1;
                        end
                  end
            end
      end
end
```

```
% insertNewSubmodel.m

function [Rnsm, RknV, RnoKn, RinDim, RinVars] = ...
      insertNewSubmodel(nsm, knV, noKn, inDim, inVars, deg, ...
            newInDim, newInVars, newKnots, newNoKnots)
% function [Rnsm, RknV, RnoKn, RinDim, RinVars] = ...
%     insertNewSubmodel(nsm, knV, noKn, inDim, inVars, deg, ...
%           newInDim, newInVars, newKnots, newNoKnots)
% Create a new model with the new submodel inserted.
% Existing submodels with input variables included in the new
% submodel are removed. Parameters for the new model is returned

% Copy the old model

Rnsm = nsm;
RknV = knV;
RnoKn = noKn;
RinDim = inDim;
RinVars = inVars;

% Add the new submodel
n = nsm+1;
Rnsm = n;
RinDim(n) = newInDim;
RinVars(n,1:newInDim) = newInVars(1:newInDim);

ind=findKVIndices(n, RinDim);
RnoKn(ind) = newNoKnots;
RknV(1:size(newKnots,1), ind) = newKnots;
% Flag submodels with all input variables included by the new submodel
for i=1:n-1
      removeFlag(i) = i;
      for j= 1:RinDim(i)
            if (isempty( find(RinVars(i ,j) ==  newInVars) ) )
                  removeFlag(i) = 0;
            end
      end
end

% Remove flagged submodels
for i=1:n-1
      r = removeFlag(i);
      if (r > 0)
            ind=findKVIndices(r, RinDim);
            RinDim(r) = [];
            RinVars(r,:) = [];
            RnoKn(ind) = [];
            RknV(:, ind) = [];
            Rnsm = Rnsm -1;
            removeFlag = removeFlag - 1; % decrement since one
submodel is removed
      end
end
```

```
% findKVIndices.m
```

```
function ind=findKVIndices(n, inDims)
% ind=findKVIndices(n, inDims);
%
% Returns a vector of indices to knot vectors
% corresponding to submodel n
% inDims are the input dimensions for all submodels

first = sum(inDims(1:n-1))+1;
ind = first:first+inDims(n)-1;
```

```
% loadModel.m

function loadModel(name)
%function loadModel(name)
% load a model  from a .mat file with given name

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_
global initKnVectors_ initNoKnots_ traces_

eval(['load ', name,])
```

```
% globals.m

% All global variables used in ASMOD functions
% Should only be run if access to the variables is wanted

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_
global initKnVectors_ initNoKnots_ traces_
```

```
%mapData.m

function est = mapData(x)
% est = mapData(x);
% Computes model outputs for input data x

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_

[nS, nV] = size(x);
if (nV ~= length(degrees_))
      'ERROR: No. of columns in x is not the same as no. input Vars.'
      return;
end

% Initialize the B matrix with a constant basis function
B = ones ( size( x(:,1) ) );

% Compute and add to B the B matrix for the submodels
for n = 1:noSm_;
      knotIndcs = findKVIndices(n, inDim_);
      vars = inVars_(n,:);
      vars = vars(find(vars > 0));
      b = computeB ( x(:,vars)', knVectors_(:,knotIndcs), ...
            noKnots_(knotIndcs), degrees_(vars))';
      B = [B, b(:,2:size(b,2))]; % Don't add the first coefficient
                                 % (avoids linear dependency in
basis functions);
end

est = B * c_;
```

```
%printModelStruct.m

function printModelStruct()
%function printModelStruct()
% prints the model structure to standard out

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_
global initKnVectors_ initNoKnots_

fid=fopen('masmod_run_rec.dat','a+');

fprintf(1,'\nCurrent model has %d coefficients', length(c_));
fprintf(fid,'\nCurrent model has %d coefficients', length(c_));
fprintf(1,'\n               Variable  Internal knots  Spline degree');
fprintf(fid,'\n               Variable  Internal knots  Spline
degree');

for i=1:noSm_
      fprintf(1,'\nSubmodel %d:  ', i);
      fprintf(fid,'\nSubmodel %d: ', i);
      ind = findKVIndices(i, inDim_);
      nkn = noKnots_(ind);
      for j=1:inDim_(i)
            fprintf(1,'     %d            %d               %d
', ...
            inVars_(i,j), nkn(j)-
2*degrees_(inVars_(i,j)),degrees_(inVars_(i,j)));
            fprintf(fid,'     %d            %d               %d
', ...
            inVars_(i,j), nkn(j)-
2*degrees_(inVars_(i,j)),degrees_(inVars_(i,j)));
      end
end
fprintf(1,'\n');
fprintf(fid,'\n');
fclose(fid);
```

```
% prunedKnotVector.m

function prunedKnVectors = pruneKnotVector(kv, nkn, degree)
% prunedKnVectors = pruneKnotVector(kv, nkn, degree);
%
% Returns a set of pruned knot vectors for input knot vector
% nkn is number of knots in the old vector and degree is spline degree

n= nkn-1;
m = nkn-2*degree;
prunedKnVectors = zeros(n,m);

for i=1:m
      prunedKnVectors(:,i) = [kv(1:degree+i-1);
                                      kv(degree+i+1:n+1)] ;
end
```

```
% refinedKnotVector.m

function refKnVectors = refineKnotVector(kv, nkn, degree)
% refKnVectors = refineKnotVector(kv, nkn, degree);
%
% Returns a set of refined knot vectors for input vector
% nkn is number of knots in teh old vector and degree is spline degree
n= nkn+1;
m = nkn-2*degree+1;
refKnVectors = zeros(n,m);

for i=1:m
      refKnVectors(:,i) = [kv(1:degree+i-1);
                                    (kv(degree+i-1) + kv(degree+i))/2;
                                    kv(degree+i:n-1)] ;
end
```

```
% saveModel.m

function saveModel(name)
%function saveModel(name)
% saves model to a .mat file with given name

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_
global initKnVectors_ initNoKnots_ traces_

eval(['save ', name,' noSm_  knVectors_ noKnots_ degrees_ inDim_
inVars_ c_ initKnVectors_ initNoKnots_ traces_'])
```

```
% SVDFit.m
```

```
% for fuzzy local linearisation model

function [c, est,B,Binv] = ...
      SVDFit(x, y, nsm, knotVs, nKnots, inDims, inVars, degs)
%  function [c, est,B,Binv] = ...
%     SVDFit(x, y, nsm, knotVs, nKnots, inDims, inVars, degs);
%
% Fits the a model of several submodels by the pseudo inverse
% computed by the SVD
% x is the input matrix, y is the target vector
% nsm the number of submodels in the model
% knotVs are knot vectors for all submodels arranged in columns
% nKnots is a vector specifying the number of knots in each knotVs
% inDims are the input dimensionanlity of all submodels
% inVars are the input variables used for all submodels arranged in
rows
% degs are the spline degrees for different input variables
% Return variables are:
% c is the computed vector of spline coefficients
% est is the vector of target estimates
% B is a matrix with the basis function values for input vectors in
rows
% Binv is the pseudoinverse of B computed by the SVD
% Modified by John Gan

% Initialize the B matrix with a constant basis function
B = ones ( size( x(:,1) ) );

% Compute and add to B the B matrix for the submodels
for n = 1:nsm;
      knotIndcs = findKVIndices(n, inDims);
      vars = inVars(n,:);
      vars = vars(find(vars > 0));
      b = computeB ( x(:,vars)', knotVs(:,knotIndcs),
nKnots(knotIndcs), degs(vars))';
      B = [B, b(:,1:size(b,2))];
%     B = [B, b(:,2:size(b,2))]; % Don't add the first coefficient
                                  % (avoids linear dependency in
basis functions);
end
% Compute the  pseudoinv(B), the coefficient vector and the estimates
c = zeros(size(B,2),1);
ind = find(max(B) > 0.01);
Binv = pinv(B(:,ind));
c(ind) = Binv * y;
est = B * c;
B=B(:,ind);
```

```
% testModel.m
% for fuzzy local linearisation model

function [rmse, nrmse, ev, mae, maxe, est] = testModel(x, y)
% [rmse, nrmse, ev, mae, maxe, est] = testModel(x, y);;
% Test the model accuracy on data set x,y
% Returns the root means squre error (rmse), the rmse scaled by the
standard
% deviation of y (nrmse), the explained variance (ev), the mean
% abolute error (mae) and the maximum error (maxe)
% est is estimated output vector
% Modified by John Gan

global  noSm_  knVectors_ noKnots_ degrees_ inDim_ inVars_ c_

[nS, nV] = size(x);
if (nV ~= length(degrees_))
      error('ERROR in testModel: No. of columns in x is not the same
as no. input Vars.')
end

%disp('Computing new test matrices')
% Initialize the B matrix with a constant basis function
B = ones ( size( x(:,1) ) );

% Compute and add to B the B matrix for the submodels
for n = 1:noSm_;
      knotIndcs = findKVIndices(n, inDim_);
      vars = inVars_(n,:);
      vars = vars(find(vars > 0));
      b = computeB ( x(:,vars)', knVectors_(:,knotIndcs), ...
            noKnots_(knotIndcs), degrees_(vars))';
      B = [B, b(:,1:size(b,2))];
%     B = [B, b(:,2:size(b,2))];    % Don't add the first coefficient
                                    % (avoids linear dependency in
basis functions);
end

est = B * c_;
res = y - est;
rmse = sqrt( (res' * res) / length( res));
nrmse = rmse / std (y);
ev = (1 - nrmse * nrmse) * 100;
mae = mean(abs(res));
maxe = max(abs(res));
```

```
% tmult.m

function z = tmult (x, y)
% function z = tmult (x, y);
% Computes the tensor product of row vectors x and y and
% rearrange into a long vector
z = reshape(x' * y,1,length(x)*length(y));
```

```
% train.m


function [traces, est] = train(x, y, xTest, yTest, criterion, maxRef)
%[traces, est] = train(x, y, xTest, yTest, criterion, maxRef)
%
% Refine the model repeatedly until optimisation
% criterion increases or after maxRef refinement steps
% Then prune the model repeatedly until optimisation
% criterion increases or after maxRef pruning steps
% The optimisation criterion is also used for selecting
% candidate models
% The currently available optimisation criteria are:
% RMS - Root mean square error on training data
% CV = 2 - Cross validation
% SRM - Structural risk minimisation
% FPE - Akaike's  final prediction error
% AIC - Akaike's information criterion
% MDL - The minimum description length criterion
%
% Train returns a matrix with traces of the different optimization
criteria
% in the columns 1 to 6, and the RMS error for the test data
(optional) in col 7
% for the sequence of refinements and prunings.
% Trace column 8 contains the model degree of freedom (DOF)
% Aslo returns the final estimates for the training data
% See also trainPrune and trainRefine

[traces, est] = trainRefine(x, y, xTest, yTest, criterion, maxRef);
[traces, est] = trainPrune(x, y, xTest, yTest, criterion, maxRef);
```

```
% trainPrune.m
```

```
function [traces, est] = trainPrune(x, y, xTest, yTest, criterion,
maxRef)
%[traces, est] = trainPrune(x, y, xTest, yTest, criterion, maxRef)
%
% Fit and prune the model repeatedly until optimisation
% criterion increases or until maxRef pruning steps are done
% The optimisation criterion is also used for selecting
% candidate models
% RMS - Root mean square error on training data
% CV = 2 - Cross validation
% SRM - Structural risk minimisation
% FPE - Akaike's  final prediction error
% AIC - Akaike's information criterion
% MDL - The minimum description length criterion
%
% Return a matrix with traces of the different optimization criteria
% in the columns 1 to 6, and the RMS error for the test data
(optional) in col 7
% for the sequence of refinements and prunings.
% Trace column 8 contains the model degree of freedom (DOF)
% Aslo returned is the final estimates for the training data
% See also train and trainRefine
% Modified by John Gan

global  noSm_  knVectors_ noKnots_ inDim_ inVars_ c_ degrees_
global traces_

if strcmp(criterion, 'RMS') cr = 1;
elseif strcmp(criterion, 'CV')  cr = 2;
elseif strcmp(criterion, 'SRM') cr = 3;
elseif strcmp(criterion, 'FPE') cr = 4;
elseif strcmp(criterion, 'AIC') cr = 5;
elseif strcmp(criterion, 'MDL') cr = 6;
else error ('Unrecognized criterion');
end

if ( isempty (traces_) )
      [c_, est, B, Binv] = SVDFit(x, y, noSm_, knVectors_, noKnots_,
inDim_, inVars_, degrees_);
      traces_ = [traces_;compCriteria(y, est, B, Binv)];
      if (~isempty(xTest))
            traces_(7) = testModel(xTest, yTest);
      end
end

refineSteps = 0;
refineFlag = 1;
while (refineSteps < maxRef & refineFlag == 1)
% while (refineSteps < maxRef)
      [nsm,knV, noKn, inDim, inVars, c, crit, degs, refineFlag,est]=
...
            findBestPruned(x, y, cr);
      if (refineFlag == 1)
            refineSteps = refineSteps +1;
            noSm_ = nsm;
            knVectors_ = knV;
            noKnots_ = noKn;
            inDim_ = inDim;
            inVars_ = inVars;
            degrees_ = degs;
            c_ = c;
            if (~isempty(xTest))
```

```
                crit(7) = testModel(xTest, yTest);
            end
            traces_ = [traces_;crit];
            printModelStruct;
            plot(traces_(:,8), traces_(:,1), '-', traces_(:,8),
traces_(:,7), '--', traces_(:,8), traces_(:,cr), '-.')
            xlabel('Degrees of Freedom')
            ylabel('RMS errors')
            title('MASMOD Traces: Train - Solid, Test - Dashed,
Criterion(SRM) - Dashdot')
            drawnow;
      end
end

traces = traces_;
```

```
% trainRefine.m

function [traces, est] = trainRefine(x, y, xTest, yTest, criterion,
maxRef)
%[traces, est] = trainRefine(x, y, xTest, yTest, criterion, maxRef)
%
% Train and refine the model repeatedly until optimisation
% criterion increases or after maxRef refinement steps
% The optimisation criterion is also used for selecting
% candidate models
% RMS - Root mean square error on training data
% CV = 2 - Cross validation
% SRM - Structural risk minimisation
% FPE - Akaike's  final prediction error
% AIC - Akaike's information criterion
% MDL - The minimum description length criterion
%
% Return a matrix with traces of the different optimization criteria
% in the columns 1 to 6, and the RMS error for the test data
(optional) in col 7
% for the sequence of refinements and prunings.
% Trace column 8 contains the model degree of freedom (DOF)
% Aslo returns the final estimates for the training data
% See also train and trainPrune
% Modified by John Gan

global  noSm_  knVectors_ noKnots_ inDim_ inVars_ c_ degrees_
global traces_

if strcmp(criterion, 'RMS') cr = 1;
elseif strcmp(criterion, 'CV')  cr = 2;
elseif strcmp(criterion, 'SRM') cr = 3;
elseif strcmp(criterion, 'FPE') cr = 4;
elseif strcmp(criterion, 'AIC') cr = 5;
elseif strcmp(criterion, 'MDL') cr = 6;
else error ('Unrecognized criterion');
end

if ( isempty (traces_) )
      [c_, est, B, Binv] = SVDFit(x, y, noSm_, knVectors_, noKnots_,
inDim_, inVars_, degrees_);
      traces_ = compCriteria(y, est, B, Binv);
      if (~isempty(xTest))
            traces_(7) = testModel(xTest, yTest);
      end
end

refineSteps = 0;
refineFlag = 1;
while (refineSteps < maxRef & refineFlag == 1)
      [nsm,knV, noKn, inDim, inVars, c, crit, refineFlag,est]= ...
            findBestRefined(x, y, cr);
      if (refineFlag == 1)
            refineSteps = refineSteps +1;
            noSm_ = nsm;
            knVectors_ = knV;
            noKnots_ = noKn;
            inDim_ = inDim;
            inVars_ = inVars;
            c_ = c;
            if (~isempty(xTest))
                  crit(7) = testModel(xTest, yTest);
            end
```

```
            traces_ = [traces_;crit];
            printModelStruct;
            plot(traces_(:,8), traces_(:,1), '-', traces_(:,8),
traces_(:,7), '--', traces_(:,8), traces_(:,cr), '-.');
            xlabel('Degrees of Freedom')
            ylabel('RMS errors')
            title('MASMOD Traces: Train - Solid, Test - Dashed,
Criterion(SRM) - Dashdot');
            drawnow;
      end
end

traces = traces_;
```

```
% 1_D function approximation: f(x)=x^2+1

rand('seed',11111);
randn('seed',55555);

xTest= 4*rand(100,1)-1;
yTest = xTest.^2+1;

xTrain= 4*rand(200,1)-1;
yTrain = xTrain.^2+1;

degree=1;   % the same spline degree (quadratic) is used for all
variables
% degree=[2 2 2 1 1 1 1 1 1 1];  % individual spline degrees can be
used

resolution=0; % the initial resolution (number of internal knots)
%resolution=[1 1 0 0 0 0 0 0 0 0];  % individual init. resol. can be
used

initSubmodels = []; % Initialize as an empty model

% initSubmodels = [2 1 2; 1 3 0; 1 4 0; 1 5 0]; % Initialize with
      % one 2 dimensional of variables 1 and 2, and three
onedimensional
      % submodels of variables 3, 4 and 5 respectively

xmin = [];xmax = []; % Automatic determination of input domain.
% xmin = [0 0 0 0 0 0 0 0 0 0]; % Setting of input domain lower
limits.
% xmax = [1 1 1 1 1 1 1 1 1 1]; % Setting of input domain upper
limits.

initializeAsmod(xTrain, degree, resolution, xmin, xmax,initSubmodels);
[traces, est] = train(xTrain, yTrain,xTest, yTest, 'SRM', 15);
%plot(traces(:,8), traces(:,1:7))
[rmse, nrmse, ev, mae, maxe, est] = testModel(xTest, yTest);
print -deps masmod_1d.eps;
plot(xTest,yTest,'.',xTest,est,'+');
drawnow;
print -deps masmod_1d_est.eps;
fname='masmod_1d_struct';
saveModel(fname);
```

```
% 2_D Gaussian function approximation.

rand('seed',11111);
randn('seed',55555);

xTest= 2*rand(1000,2)-1;
yTest = exp(-5*(xTest(:,1)-0.5).^2-25*(xTest(:,2)-0.5).^2);

xTrain= 2*rand(4000,2)-1;
yTrain = exp(-5*(xTrain(:,1)-0.5).^2-25*(xTrain(:,2)-0.5).^2);

degree=1;   % the same spline degree (quadratic) is used for all
variables
% degree=[2 2 2 1 1 1 1 1 1 1];  % individual spline degrees can be
used

resolution=0; % the initial resolution (number of internal knots)
%resolution=[1 1 0 0 0 0 0 0 0 0];  % individual init. resol. can be
used

initSubmodels = []; % Initialize as an empty model

% initSubmodels = [2 1 2; 1 3 0; 1 4 0; 1 5 0]; % Initialize with
      % one 2 dimensional of variables 1 and 2, and three
onedimensional
      % submodels of variables 3, 4 and 5 respectively

xmin = [];xmax = []; % Automatic determination of input domain.
% xmin = [0 0 0 0 0 0 0 0 0 0]; % Setting of input domain lower
limits.
% xmax = [1 1 1 1 1 1 1 1 1 1]; % Setting of input domain upper
limits.

initializeAsmod(xTrain, degree, resolution, xmin, xmax,initSubmodels);
[traces, est] = train(xTrain, yTrain,xTest, yTest, 'SRM', 15);
%plot(traces(:,8), traces(:,1:7))
[rmse, nrmse, ev, mae, maxe] = testModel(xTest, yTest);
print -deps masmod_2d.eps;
fname='masmod_2d_struct';
saveModel(fname);
```

```
% This is a 10-dimensional test function designed by Friedman
% for testing the MARS algorithm.

rand('seed',11111);
randn('seed',55555);

xTest = rand(1000,10);
yTest = 10*sin(pi*xTest(:,1).*xTest(:,2))+20*(xTest(:,3)-
0.5).^2+10*xTest(:,4)+5*xTest(:,5);

xTrain = rand(50,10);
yTrain = 10*sin(pi*xTrain(:,1).*xTrain(:,2))+20*(xTrain(:,3)-
0.5).^2+10*xTrain(:,4)+5*xTrain(:,5)+randn(50,1);

degree=2;   % the same spline degree (quadratic) is used for all
variables
% degree=[2 2 2 1 1 1 1 1 1 1];  % individual spline degrees can be
used

resolution=0; % the initial resolution (number of internal knots)
%resolution=[1 1 0 0 0 0 0 0 0 0];  % individual init. resol. can be
used

initSubmodels = []; % Initialize as an empty model

% initSubmodels = [2 1 2; 1 3 0; 1 4 0; 1 5 0]; % Initialize with
      % one 2 dimensional of variables 1 and 2, and three
onedimensional
      % submodels of variables 3, 4 and 5 respectively

xmin = [];xmax = []; % Automatic determination of input domain.
% xmin = [0 0 0 0 0 0 0 0 0 0]; % Setting of input domain lower
limits.
% xmax = [1 1 1 1 1 1 1 1 1 1]; % Setting of input domain upper
limits.

initializeAsmod(xTrain, degree, resolution, xmin, xmax,initSubmodels);
[traces, est] = train(xTrain, yTrain,xTest, yTest, 'SRM', 15);
%plot(traces(:,8), traces(:,1:7))
[rmse, nrmse, ev, mae, maxe] = testModel(xTest, yTest);
print -deps masmod_10d.eps;
fname='masmod_10d_struct';
saveModel(fname);
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Υπολογιστική Νοημοσύνη, *Αριστείδης Λυκάς.*
2. Υπολογιστική Νοημοσύνη, τόμος Α, *Σπύρος Γ. Τζαφέστας.*
3. Ευφυής Αυτόματος Έλεγχος, *Σπύρος Τζαφέστας.*
4. Neuro-Fuzzy Systems, *Detlef Nauck, Frank Klawonn, Rudolf Kruse.*
5. A First Course in FUZZY LOGIC, *Hung T. Nguyen, Elbert A. Walker.*
6. Fuzzy Control and Fuzzy Systems, *Witold Pedrycz.*
7. Multivariable Neural Control, *S.G. Tzafestas, G.G. Rigatos.*
8. Neurocytology, *Stanley Jacobson.*
9. Neural Networks and Simulation methods, *Jian-Kang Wu.*
10. “Linearization and State Estimation of Unknown Discrete – Time Nonlinear Dynamic Systems Using Recurrent Neurofuzzy Networks”, *Qiang Gan and Chris J. Harris.*
11. “Fuzzy Local Linearization and Local Basis Function Expansion in Nonlinear System Modeling”, *Qiang Gan and Chris J. Harris.*

“Underactuated Dynamic Positioning of a Ship-Experimental Results”, *Kristin Y. Pettersen and Thor I. Fossen.*