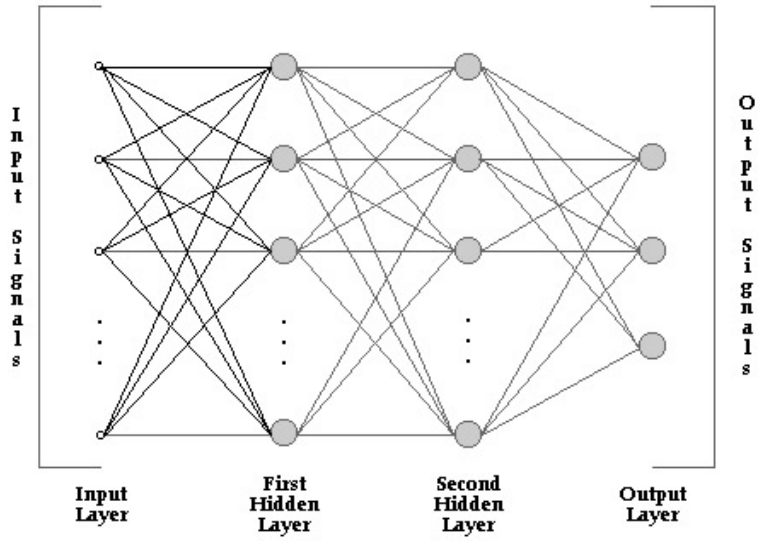
Τσουχνικά Μαρία

ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ

Επιβλέπων: Dr Γ. Βουγιατζής



1. Εισαγωγή

Τα τεχνητά νευρωνικά δίκτυα είναι ένα χρήσιμο υπολογιστικό εργαλείο, με πολλαπλές εφαρμογές. Στόχος της εργασίας αυτής είναι να παρουσιαστούν τα κύρια χαρακτηριστικά τους, οι δυνατότητες και οι αδυναμίες τους, καθώς και κάποιες βασικές τροποποιήσεις, που έπαιξαν σημαντικό ρόλο στην εξέλιξή τους, από την πρώτη εμφάνισή τους μέχρι και σήμερα.

2. Γενικές έννοιες

2.1 Η έννοια του τεχνητού νευρωνικού δικτύου

Τα *τεχνητά νευρωνικά δίκτυα* (*artificial neural networks*) αρχικά προτάθηκαν ως ένα μαθηματικό μοντέλο προσομοίωσης της πολύπλοκης λειτουργίας του ανθρώπινου εγκεφάλου. Η δομή του εγκεφάλου είναι τέτοια ώστε να επιτρέπει την παράλληλη επεξεργασία δεδομένων και τη δυνατότητα συνεχούς μάθησης μέσω της αλληλεπίδρασης με το περιβάλλον. Τα δύο αυτά βασικά χαρακτηριστικά συμβάλλουν στην ικανότητα, αφενός, να εκτελεί δύσκολα καθήκοντα, όπως ταχύτατη αναγνώριση μορφών, ταξινόμηση κ.ά., αφετέρου, να εξελίσσεται συνεχώς, μαθαίνοντας από το περιβάλλον του κατά την αλληλεπίδρασή του με αυτό.

Η δομή του τεχνητού νευρωνικού δικτύου μιμείται κατά το δυνατό εκείνη του βιολογικού νευρωνικού δικτύου, ώστε να εμφανίζει παρόμοιες ιδιότητες. Κατ' αναλογία επομένως με ένα δίκτυο νευρώνων εγκεφάλου, ένα τεχνητό δίκτυο αποτελείται από ένα σύνολο τεχνητών νευρώνων που αλληλεπιδρούν, συνδεόμενοι μεταξύ τους με τις λεγόμενες *συνάψεις* (*synapses*). Ο βαθμός αλληλεπίδρασης είναι διαφορετικός για κάθε ζεύγος νευρώνων και καθορίζεται από τα λεγόμενα *συναπτικά βάρη* (*synaptic weights*).

Συγκεκριμένα, καθώς το νευρωνικό δίκτυο αλληλεπιδρά με το περιβάλλον και μαθαίνει από αυτό, τα συναπτικά βάρη μεταβάλλονται συνεχώς, ενδυναμώνοντας ή αποδυναμώνοντας την ισχύ του κάθε δεσμού. Όλη η εμπειρική γνώση που αποκτά επομένως το νευρωνικό δίκτυο από το περιβάλλον κωδικοποιείται στα συναπτικά βάρη. Αυτά αποτελούν το χαρακτηριστικό εκείνο που δίνει στο δίκτυο την ικανότητα για εξέλιξη και προσαρμογή στο περιβάλλον.

Υπάρχουν δύο τρόποι να εκπαιδεύσουμε ένα δίκτυο. Κατά τον πρώτο τρόπο, η εκπαίδευση γίνεται με εποπτεία. Στην περίπτωση αυτή το δίκτυο τροφοδοτείται με ένα σύνολο γνωστών παραδειγμάτων, δηλαδή ένα σύνολο καταστάσεων στις οποίες μπορεί να περιέλθει το δίκτυο, μαζί με τα αποτελέσματα που θέλουμε να δίνει το δίκτυο για τις καταστάσεις

αυτές. Για να μάθει το δίκτυο τα παραδείγματα αυτά, χρησιμοποιούμε έναν αλγόριθμο εκπαίδευσης. Ο αλγόριθμος εκπαίδευσης που θα χρησιμοποιηθεί εξαρτάται από το εκάστοτε πρόβλημα και από τη δομή του δικτύου που επιλέγουμε για να το αντιμετωπίσουμε. Κατά το δεύτερο τρόπο, η εκπαίδευση γίνεται χωρίς εποπτεία. Στην περίπτωση αυτή το δίκτυο καλείται να αναγνωρίσει ομοιότητες και μοτίβα σε δεδομένα που του έχουμε τροφοδοτήσει. Τα δεδομένα παρουσιάζονται στο δίκτυο και αυτό οφείλει να προσαρμοστεί έτσι ώστε να τα χωρίσει σε ομάδες. Η διαδικασία αυτή επαναλαμβάνεται, ώσπου δεν παρατηρείται μεταβολή στην ταξινόμηση των δεδομένων.

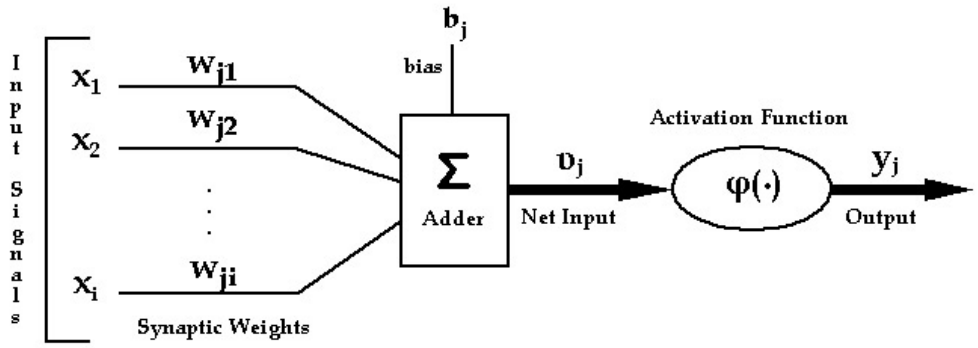
Το βασικό πλεονέκτημα των νευρωνικών δικτύων είναι ότι μπορούν να αποθηκεύσουν γνώση και εμπειρία από το περιβάλλον, την οποία μπορεί στη συνέχεια να ανακαλέσει. Επιπλέον, έχει τη δυνατότητα να γενικεύει, δηλαδή να εξάγει τα βασικά χαρακτηριστικά ενός συστήματος, ακόμα και όταν αυτά είναι κρυμμένα σε θορυβώδη δεδομένα.

2.2 Η δομή του νευρώνα

Σε αναλογία με το βιολογικό νευρώνα του εγκεφάλου, ο *τεχνητός νευρώνας* (*artificial neuron*) είναι η δομική μονάδα του τεχνητού νευρωνικού δικτύου. Σε αυτόν συντελείται όλη η επεξεργασία της πληροφορίας. Κάθε νευρώνας δέχεται πληροφορία, την επεξεργάζεται και δίνει μία τιμή εξόδου. Οι είσοδοί του είναι είτε οι έξοδοι άλλων νευρώνων, είτε το πρωταρχικό σήμα εισόδου του δικτύου.

Υπάρχουν διάφορα είδη νευρώνα. Το είδος που θα επιλεχθεί για να δομηθεί ένα συγκεκριμένο τεχνητό νευρωνικό δίκτυο, εξαρτάται από τη φύση του εκάστοτε προβλήματος που εξετάζουμε. Σε πολλές περιπτώσεις χρησιμοποιείται συνδυασμός διαφορετικών ειδών νευρώνα.

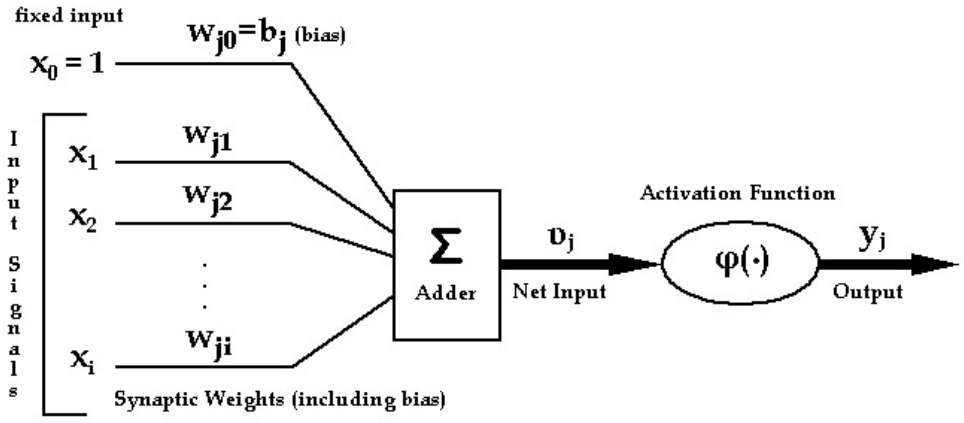
Στο Σχήμα 1, παρακάτω, παρουσιάζεται το βασικό μοντέλο του νευρώνα που χρησιμοποιείται κατά κόρον σε υλοποιήσεις τεχνητών νευρωνικών δικτύων.



Σχήμα 1: Σχηματική αναπαράσταση μη γραμμικού νευρώνα

Στο νευρώνα αυτό, η πληροφορία ρέει πάντα προς μία κατεύθυνση, από αριστερά προς τα δεξιά, δεν υπάρχει δηλαδή κανένας βρόχος ανάδρασης. Βάσει αυτού, μπορούμε να διακρίνουμε τρεις βασικές φάσεις της λειτουργίας του:

Κατά την πρώτη φάση, κάθε είσοδος πολλαπλασιάζεται με το συναπτικό βάρος που της αντιστοιχεί. Στη δεύτερη φάση οι σταθμισμένες πλέον είσοδοι και ένας εξωτερικά εφαρμοζόμενος παράγοντας, η *μεροληψία* ή *πόλωση* ή *κατώφλι* (*bias*, *thresold*), αθροίζονται και δίνουν το *τοπικό πεδίο* (*net input*, *induced local field*, *activation potential*). Για λόγους απλούστευσης, η μεροληψία μπορεί να θεωρηθεί ως μία επιπλέον είσοδος, με συναπτικό βάρος ίσο προς την τιμή του και πάγια τιμή εισόδου ίση προς τη μονάδα. Στην περίπτωση αυτή ο νευρώνας παίρνει τη μορφή που φαίνεται στο Σχήμα 2 που ακολουθεί.



Σχήμα 2: Εναλλακτική σχηματική αναπαράσταση μη γραμμικού νευρώνα

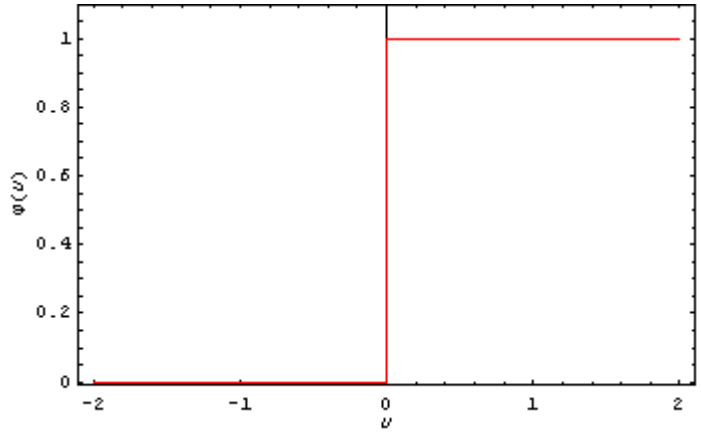
Ως εδώ, ο νευρώνας δεν κάνει τίποτα άλλο από το να δίνει έναν γραμμικό συνδυασμό των εισόδων, με συντελεστές τα προσαρμοζόμενα συναπτικά βάρη. Αν η λειτουργία του λοιπόν σταματούσε εδώ, τότε θα είχαμε έναν γραμμικό νευρώνα, που θα έδινε ένα *γραμμικό προσαρμοζόμενο φίλτρο* (*linear adaptive filter*). Ένα τεχνητό νευρωνικό δίκτυο που αποτελείται από τέτοιους νευρώνες θα είναι γραμμικό.

Τέλος, στην τρίτη φάση, εφαρμόζεται η *συνάρτηση ενεργοποίησης* ή *συνάρτηση μεταφοράς* (*activation function* ή *squashing function*) στο τοπικό πεδίο και το αποτέλεσμα δίνει την έξοδο του νευρώνα.

Στα πρώτα μοντέλα νευρώνα, η συνάρτηση ενεργοποίησης ήταν μία *βηματική συνάρτηση* (*step function*).

$$\varphi(\upsilon) = \begin{cases} 0, & \upsilon \le \theta \\ 1, & \upsilon > \theta \end{cases} \qquad (1)$$

Μία τέτοια συνάρτηση φαίνεται παρακάτω στο Σχήμα 3. Αν το ενδιάμεσο αποτέλεσμα ήταν μικρότερο μιας τιμής κατωφλίου, η έξοδος του νευρώνα ήταν ίση προς 0 (αδρανής νευρώνας), αλλιώς ήταν ίση προς 1 (ενεργοποιημένος νευρώνας).



Σχήμα 3: Βηματική συνάρτηση για τιμή κατωφλίου ίση προς μηδέν

Το παραπάνω μοντέλο αναφέρεται συχνά ως μοντέλο McCulloch-Pitts προς τιμή αυτών που το πρότειναν.

Αργότερα, η εξέλιξη στο θεωρητικό υπόβαθρο των τεχνητών νευρωνικών δικτύων φανέρωσε ότι η παράγωγος της συνάρτησης ενεργοποίησης μπορεί να δώσει χρήσιμες πληροφορίες για το νευρωνικό δίκτυο και να χρησιμοποιηθεί στην εκπαίδευσή του, γεγονός που υποδεικνύει ότι είναι προτιμότερο να χρησιμοποιηθεί μία παραγωγίσιμη συνάρτηση και όχι η βηματική συνάρτηση, που είναι προφανώς μη παραγωγίσιμη.

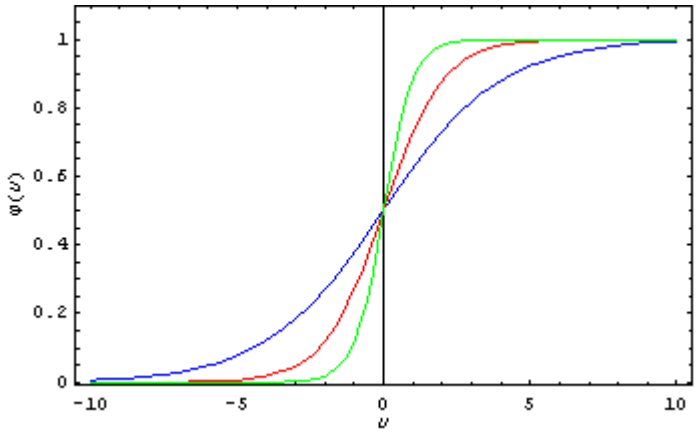
Σήμερα, στα περισσότερα μοντέλα η συνάρτηση ενεργοποίησης είναι μία σιγμοειδής συνάρτηση. Αυτή είναι γενικά μία πραγματική, συνεχής και φραγμένη συνάρτηση, της οποίας η παράγωγος είναι θετική. Το πεδίο ορισμού της μπορεί θεωρητικά να είναι όλο το σύνολο των πραγματικών αριθμών, αλλά στην πράξη μπορεί να περιοριστεί, θέτοντας όρια στις τιμές των συναπτικών βαρών. Το σύνολο τιμών είναι συνήθως το διάστημα [0,1] ή [-1,1].

Ένα από τα πιο γνωστά παραδείγματα σιγμοειδούς συνάρτησης που χρησιμοποιείται ως συνάρτηση ενεργοποίησης είναι η *λογιστική συνάρτηση* (*logistic function*), που δίνεται από τον τύπο

$$\varphi(\upsilon) = \frac{1}{1+e^{-\alpha\upsilon}} \qquad (2)$$

όπου α η παράμετρος κλίσης. Μεταβάλλοντας την παράμετρο κλίσης, παίρνουμε συναρτήσεις με διαφορετικές κλίσεις. Όσο το α τείνει στο άπειρο, η λογιστική συνάρτηση τείνει προς τη βηματική συνάρτηση και έχουμε και πάλι το μοντέλο McCulloch-Pitts. Στο Σχήμα 4 που

ακολουθεί, φαίνεται η γραφική παράσταση της λογιστικής συνάρτησης για διάφορες τιμές της παραμέτρου κλίσης α .



Σχήμα 4: Η λογιστική συνάρτηση, για α = 0.5 (μπλε), α = 1 (κόκκινο) και α = 2 (πράσινο)

Άλλα παραδείγματα σιγμοειδών συναρτήσεων που χρησιμοποιούνται ως συναρτήσεις ενεργοποίησης είναι η υπερβολική συνάρτηση (hyperbolic function) και η συνάρτηση τόξου εφαπτομένης (arctangent function).

Με την εισαγωγή της συνάρτησης ενεργοποίησης, ο νευρώνας γίνεται μη γραμμικός. Αντίστοιχα, ένα τεχνητό νευρωνικό δίκτυο που αποτελείται από τέτοιους νευρώνες θα είναι μη γραμμικό. Αυτή η εγγενής μη γραμμικότητα των νευρωνικών δικτύων είναι ένα πλεονέκτημα έναντι άλλων γνωστών μεθόδων αντιμετώπισης πολλών προβλημάτων. Για παράδειγμα, όταν σε ένα πρόβλημα πρόβλεψης το σύστημα που μελετάμε είναι μη γραμμικό και ιδιαίτερα όταν παρουσιάζει χαοτική συμπεριφορά, τα γνωστά γραμμικά μοντέλα πρόβλεψης αδυνατούν να δώσουν σωστά αποτελέσματα. Σε αυτές τις περιπτώσεις, τα μη γραμμικά τεχνητά νευρωνικά δίκτυα είναι προτιμότερα.

Επειδή πολλές φορές στην πράξη χρειάζεται κάποιοι από τους νευρώνες ενός μη γραμμικού νευρωνικού δικτύου να είναι γραμμικοί, μία ακόμη συνάρτηση ενεργοποίησης που χρησιμοποιείται είναι η γραμμική, όπως η

$$\boxed{\varphi(\upsilon) = \upsilon} \tag{3}$$

ή κάποια άλλη γραμμική συνάρτηση.

Σύμφωνα με τα παραπάνω, η λειτουργία του νευρώνα περιγράφεται από τις σχέσεις

$$\boxed{u_j = \sum_{i=1}^{m} w_{ji} x_i} \tag{4}$$

και

$$\boxed{y_j = \phi(\upsilon_j)} \tag{5}$$

με

$$\boxed{\upsilon_j = u_j + b_j} \tag{6}$$

όπου

m ο αριθμός των εισόδων που δέχεται ο νευρώνας, x_i η i-οστή είσοδος του νευρώνα, w_{ji} το βάρος της σύναψης που συνδέει τον νευρώνα j με την είσοδο i, u_j το σταθμισμένο άθροισμα των εισόδων, b_j η μεροληψία του νευρώνα, υ_j το τοπικό πεδίο και y_j η τελική έξοδος.

Οι παραπάνω σχέσεις περιγράφουν τον νευρώνα του Σχήματος 2. Για την εναλλακτική μορφή του νευρώνα όπως αυτός του Σχήματος 3, οι σχέσεις (4), (5) και (6) συνδυάζονται σε μία σχέση και έχουμε

$$\boxed{\upsilon_j = \sum_{i=0}^{m} w_{ji} x_i} \tag{7}$$

και

$$\boxed{y_j = \phi(\upsilon_j)} \tag{8}$$

με

$$\boxed{x_0 = +1} \tag{9}$$

και

$$\boxed{w_{j0} = b_j} \tag{10}$$

Τέλος, ένας ακόμα τύπος νευρώνα είναι ο λεγόμενος *στοχαστικός νευρώνας* (*stochastic neuron*). Στον νευρώνα αυτό η συνάρτηση ενεργοποίησης είναι πιθανοκρατική. Συγκεκριμένα, ο νευρώνας μπορεί να δώσει δύο πιθανές εξόδους, π.χ. 0 και 1 σύμφωνα με τη σχέση

$$\boxed{y_j = \begin{cases} 0, & \mu\varepsilon\ \pi\iota\theta\alpha\nu\acute{o}\tau\eta\tau\alpha\ 1 - P(\upsilon) \\ 1, & \mu\varepsilon\ \pi\iota\theta\alpha\nu\acute{o}\tau\eta\tau\alpha\ P(\upsilon) \end{cases}} \tag{11}$$

όπου συνήθως η $P(\upsilon)$ είναι

$$\boxed{P(\upsilon) = \frac{1}{1+e^{-\upsilon/T}}} \tag{12}$$

όπου T είναι μία παράμετρος, η οποία χρησιμοποιείται για να ελέγχεται το επίπεδο του θορύβου και δεν έχει καμία φυσική σημασία. Στο όριο $T \to 0$ το μοντέλο αυτό μετατρέπεται στο μοντέλο McCulloch-Pitts.

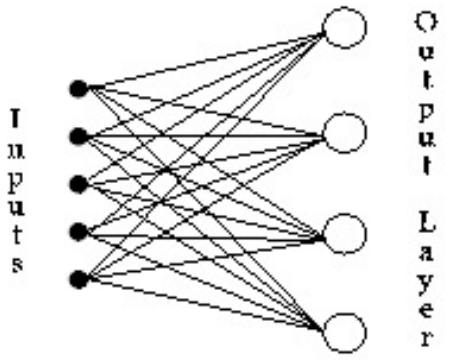
2.3 Παραδείγματα νευρωνικών δικτύων

Ένα τεχνητό νευρωνικό δίκτυο καθορίζεται από τον τύπο ή τους τύπους των νευρώνων που το απαρτίζουν, από τον τρόπο με τον οποίο είναι συνδεδεμένοι οι νευρώνες του, από τον αλγόριθμο που χρησιμοποιείται για την εκπαίδευσή του, από το αν η εκπαίδευση γίνεται με εποπτεία ή χωρίς κ.ά.

Στην εργασία αυτή μελετήσαμε δίκτυα των οποίων οι νευρώνες είναι δομημένοι σε επίπεδα και που η εκπαίδευσή τους γίνεται με εποπτεία. Η πιο απλή μορφή ενός τέτοιου δικτύου είναι αυτή του απλού *αισθητήρα* (*perceptron*), ο οποίος ήταν το πρώτο τεχνητό νευρωνικό δίκτυο το οποίο εισηγήθηκε ο Rosenblatt το 1958.

2.3.1 Ο αισθητήρας (Perceptron)

Ο *αισθητήρας* (*perceptron*) είναι ένα δίκτυο με δύο επίπεδα. Το πρώτο επίπεδο απαρτίζεται από τις εισόδους του δικτύου, δεν έχει νευρώνες και επομένως δεν γίνεται καμία επεξεργασία πληροφορίας σε αυτό. Το δεύτερο επίπεδο αποτελείται από νευρώνες τύπου McCulloch-Pitts και είναι το επίπεδο εξόδου του δικτύου. Ένα παράδειγμα αισθητήρα, με έξι εισόδους και τέσσερις νευρώνες στο επίπεδο εξόδου, φαίνεται στο Σχήμα 5 που ακολουθεί.



Σχήμα 5: Παράδειγμα αισθητήρα (perceptron), με 6 εισόδους και 4 νευρώνες εξόδου

Ο στόχος του απλού αισθητήρα είναι να μάθει να λύνει προβλήματα ταξινόμησης, να αντιστοιχεί δηλαδή κάθε σετ εισόδων που δέχεται στη σωστή κλάση. Ο αισθητήρας μπορεί να λύσει πολλά τέτοια προβλήματα με επιτυχία. Ένα από τα πλεονεκτήματα του δικτύου αυτού είναι ότι υπάρχει ένας σαφής αλγόριθμος βάσει του οποίου μπορεί να εκπαιδευτεί, ώστε να δίνει σωστά αποτελέσματα. Ο αλγόριθμος αυτός, για την πιο απλή περίπτωση για την οποία τα σετ των εισόδων προέρχονται από δύο κλάσεις, έχει ως εξής:

$$\bar{w}_j(n+1) = \begin{cases} \bar{w}_j(n) & \text{όταν η } y_j \text{ είναι σωστή} \\ \bar{w}_j(n) - \eta(n)\bar{x}_j(n) & \text{όταν } y_j = 1, \text{ ενώ θα επρεπε να ειναι } 0 \\ \bar{w}_j(n) + \eta(n)\bar{x}_j(n) & \text{όταν } y_j = 0, \text{ ενώ θα επρεπε να ειναι } 1 \end{cases} \quad (13)$$

όπου $\bar{x}_j(n)$ το διάνυσμα εισόδου του νευρώνα j, $\bar{y}_j(n)$ το διάνυσμα εξόδου και $\bar{w}_j(n)$ το διάνυσμα των βαρών στο βήμα n του αλγορίθμου, $\bar{w}_j(n+1)$ το διάνυσμα των βαρών στο βήμα $n+1$ και η θετική σταθερά που ονομάζεται *παράμετρος ρυθμού εκπαίδευσης* (*learning-rate parameter*).

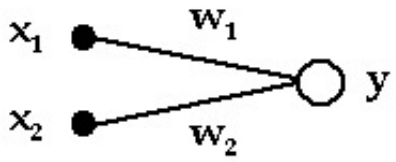
Το ερώτημα που προκύπτει άμεσα είναι κατά πόσο ο αλγόριθμος αυτός συγκλίνει σε μία σωστή λύση του προβλήματος. Όπως ο ίδιος ο Rosenblatt απέδειξε, αν οι κλάσεις του προβλήματος είναι διαχωρήσιμες από τον αισθητήρα, τότε ο παραπάνω αλγόριθμος συγκλίνει και δίνει σωστά αποτελέσματα και μάλιστα σε πεπερασμένο αριθμό βημάτων. Το θεώρημα αυτό ονομάζεται *θεώρημα σύγκλισης του αισθητήρα* (*perceptron convergence theorem*). Η ισχύς του θεωρήματος αυτού αποτελεί ένα ακόμα σημαντικό πλεονέκτημα του αισθητήρα.

Αργότερα όμως οι Minsky και Papert έδειξαν ότι τα προβλήματα ταξινόμησης που μπορεί να λύσει ο αισθητήρας είναι εκείνα τα οποία είναι *γραμμικά διαχωρίσιμα* και μόνο. Το συμπέρασμα αυτό γίνεται εύκολα κατανοητό με ένα απλό παράδειγμα. Έστω λοιπόν ότι ο στόχος του αισθητήρα είναι να δίνει σωστά αποτελέσματα σύμφωνα με τη λογική πράξη XOR, η οποία φαίνεται παρακάτω στον Πίνακα 1.

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Πίνακας 1: Η πράξη XOR

Ο κατάλληλος αισθητήρας για το πρόβλημα αυτό είναι εκείνος με δύο εισόδους και έναν νευρώνα εξόδου, όπως φαίνεται στο Σχήμα 6, που ακολουθεί



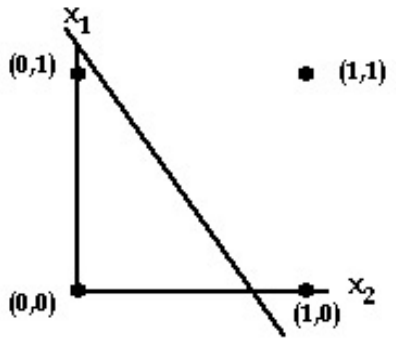
Σχήμα 6: Ο αισθητήρας για την XOR

Η έξοδος του αισθητήρα, όπως είδαμε και παραπάνω, θα δίνεται από τη σχέση

$$\boxed{y = \begin{cases} 0, & \alpha\nu\ w_1 x_1 + w_2 x_2 \le \theta \\ 1, & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}} \qquad (14)$$

όπου w_1 και w_2 τα συναπτικά βάρη και θ το κατώφλι. Το ζητούμενο είναι να βρεθούν οι τιμές για τα βάρη, τέτοιες ώστε το αποτέλεσμα των εισόδων να συνάγει με τη λογική πράξη XOR.

Τα τέσσερα σετ εισόδων του αισθητήρα αντιστοιχούν σε τέσσερα σημεία στο επίπεδο (x_1, x_2). Σύμφωνα με τον Πίνακα 1, τα σημεία (0,0) και (1,1) ανήκουν στην κλάση 0, ενώ τα (0,1) και (1,0) στην κλάση 1. Η εξίσωση $\theta = w_1 x_1 + w_2 x_2$ είναι μία ευθεία που χωρίζει το επίπεδο (x_1, x_2) σε δύο ημιεπίπεδα. Αρκεί λοιπόν να βρεθεί ένα ζεύγος τιμών για τα w_1 και w_2, τέτοιες ώστε να χωρίζουν τις δύο αυτές κλάσεις. Όπως φαίνεται όμως στο Σχήμα 7 παρακάτω, αυτό δεν είναι εφικτό. Δεν υπάρχει καμία ευθεία τέτοια ώστε τα σημεία των δύο κλάσεων να ανήκουν σε διαφορετικά ημιεπίπεδα. Για να το πετύχουμε αυτό χρειαζόμαστε δύο ευθείες. Το πρόβλημα αυτό επομένως δεν είναι γραμμικά διαχωρίσιμο και άρα δεν είναι επιλύσιμο από τον αισθητήρα.



Σχήμα 7: Το πρόβλημα XOR είναι μη γραμμικά διαχωρίσιμο

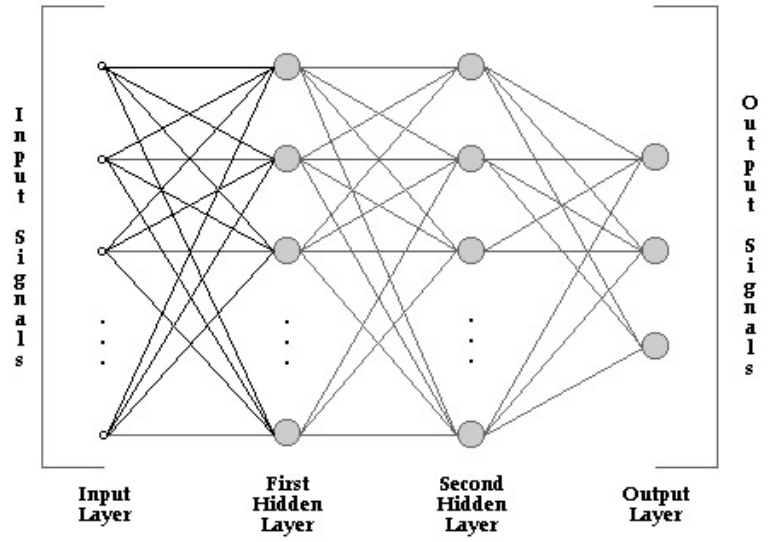
Η αδυναμία του αισθητήρα να επιλύσει το πρόβλημα της XOR, δεν είναι αμελητέα. Όπως επεσήμαναν οι Minsky και Papert, πολλά προβλήματα βασίζονται στην XOR. Η αποτυχία αυτή είχε σαν αποτέλεσμα να

εγκαταλειφθεί η ιδέα των νευρωνικών δικτύων και να σταματήσει σχεδόν κάθε έρευνα πάνω στο αντικείμενο αυτό.

Προκύπτει επομένως το ερώτημα, κατά πόσο θα μπορούσε το αρχικό μοντέλο του αισθητήρα να τροποποιηθεί, ώστε να μπορεί να επιλύει και μη γραμμικά διαχωρίσιμα προβλήματα. Η απάντηση στο ερώτημα αυτό είναι καταφατική. Πράγματι, προσθέτοντας απλά ένα ή περισσότερα επίπεδα νευρώνων μεταξύ του επίπεδου εισόδου και αυτό της εξόδου, ο τροποποιημένος αισθητήρας που προκύπτει μπορεί πλέον να επιλύσει και μη γραμμικά διαχωριζόμενα προβλήματα.

2.3.2 Πολυεπίπεδοι Αισθητήρες (MultiLayer Perceptrons, MLPs)

Το τροποποιημένο αυτό μοντέλο του απλού αισθητήρα ονομάζεται *πολυεπίπεδος αισθητήρας* (*multilayer perceptron* ή *multilayer feedforward networks*). Σε ένα τέτοιο νευρωνικό δίκτυο, μεταξύ των επιπέδων εισόδου και εξόδου, μεσολαβούν και ένα ή περισσότερα επίπεδα ακόμα, τα λεγόμενα *κρυφά επίπεδα* (*hidden layers*). Ένα παράδειγμα τέτοιου δικτύου με δύο κρυφά επίπεδα φαίνεται παρακάτω, στο Σχήμα 8.



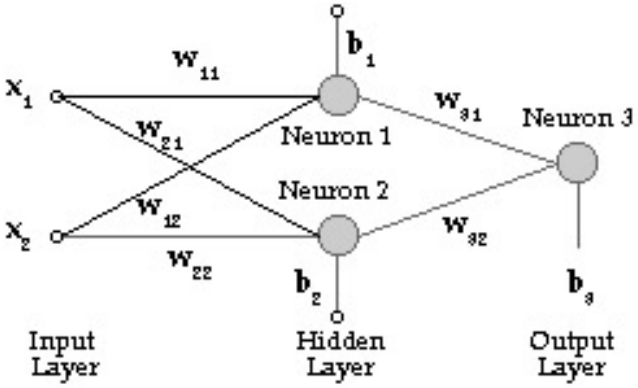
Σχήμα 8: Πολυεπίπεδος αισθητήρας με δύο κρυφά επίπεδα

Η ροή της πληροφορίας σε ένα τέτοιο δίκτυο γίνεται πάντα από τα αριστερά προς τα δεξιά, δεν υπάρχει κανένας βρόχος ανάδρασης. Θεωρούμε επίσης ότι, οι νευρώνες σε κάθε επίπεδο αλληλεπιδρούν μόνο με εκείνους τους νευρώνες που ανήκουν στα άμεσα γειτονικά τους επίπεδα. Δηλαδή το πρώτο κρυφό επίπεδο δέχεται τις τιμές του επιπέδου εισόδου, τα αποτελέσματα του πρώτου κρυφού επιπέδου περνάνε στο δεύτερο κρυφό, του οποίου τα αποτελέσματα στη συνέχεια περνάνε στο επίπεδο εξόδου. Ένα δίκτυο πολυεπίπεδου αισθητήρα στο οποίο υπάρχουν όλες οι επιτρεπτές συνδέσεις μεταξύ των νευρώνων, ονομάζεται

πλήρως συνδεδεμένο (*fully connected*), αλλιώς ονομάζεται *μερικά συνδεδεμένο* (*partially connected*).

Όπως είδαμε προηγουμένως, ένας απλός αισθητήρας δεν μπορεί να δώσει σωστά αποτελέσματα σε μη γραμμικά διαχωρίσιμα προβλήματα, καθώς δεν μπορεί να χωρίσει το επίπεδο σε περισσότερες από δύο περιοχές. Αν λοιπόν μπορούσαμε με κάποιο τρόπο να συνδυάσουμε τα αποτελέσματα δύο διαφορετικών αισθητήρων, τότε θα ήταν δυνατό να χωρίσουμε το επίπεδο σε περισσότερες περιοχές και να είχαμε το επιθυμητό αποτέλεσμα.

Το παραπάνω γίνεται εύκολα κατανοητό ξαναβλέποντας το παράδειγμα της XOR. Θεωρούμε και πάλι ένα νευρωνικό δίκτυο με δύο εισόδους και μία έξοδο, αλλά αυτή τη φορά παρεμβάλλουμε και ένα κρυφό επίπεδο με δύο νευρώνες μεταξύ τους, όπως αυτό που φαίνεται στο Σχήμα 9. Ένας τέτοιος πολυεπίπεδος αισθητήρας αναφέρεται για συντομία ως 2:2:1

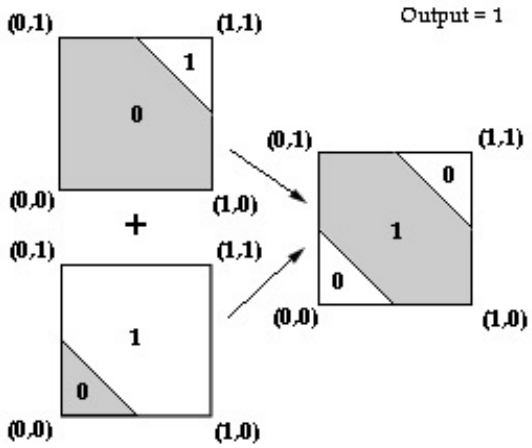


Σχήμα 9: Η XOR με έναν 2:2:1 πολυεπίπεδο αισθητήρα

Εύκολα επαληθεύεται ότι χρησιμοποιώντας το παρακάτω σετ τιμών

$$\begin{cases} w_{11} = w_{12} = w_{21} = w_{22} = w_{32} = +1 \\ w_{31} = -2 \end{cases} \quad \text{και} \quad \begin{cases} b_1 = -\dfrac{3}{2} \\ b_2 = b_3 = -\dfrac{1}{2} \end{cases} \tag{15}$$

είναι μία λύση του προβλήματος. Στο Σχήμα 10, φαίνεται πως λειτουργεί το δίκτυο αυτό, για να δώσει το σωστό αποτέλεσμα. Ο ένας νευρώνας του κρυφού επιπέδου διαχωρίζει το σετ εισόδου (1,1) από τα υπόλοιπα τρία. Ο δεύτερος νευρώνας διαχωρίζει το σετ εισόδου (0,0) από τα υπόλοιπα τρία. Ο νευρώνας του επιπέδου εξόδου συνδυάζει τα αποτελέσματα των δύο προηγούμενων και δίνει το επιθυμητό αποτέλεσμα. Τοποθετεί επιτυχώς τα σετ (0,0) και (1,1) στην κλάση 0 και τα (0,1) και (1,0) στην κλάση 1.



Σχήμα 10: Διαχωρισμός των κλάσεων της XOR στο επίπεδο, με το δίκτυο 2:2:1

Ωστόσο, το γεγονός ότι υπάρχει μία λύση για ένα συγκεκριμένο πρόβλημα δεν έχει καμία πρακτική αξία, αν δεν υπάρχει ένας τυποποιημένος τρόπος να βρίσκεται κάθε φορά μία τέτοια λύση. Το πρόβλημα όμως είναι ότι στο τροποποιημένο αυτό μοντέλο του αισθητήρα, δεν μπορεί να εφαρμοστεί ο αλγόριθμος εκπαίδευσης του Rosenblatt. Ήταν απαραίτητο επομένως να βρεθεί ένας νέος αλγόριθμος, τέτοιος ώστε το νευρωνικό δίκτυο να βρίσκει κάθε φορά το κατάλληλο σετ τιμών για να επιλύει το εκάστοτε πρόβλημα. Με άλλα λόγια, πρέπει να βρεθεί ένας τρόπος να ενισχυθεί η συμμετοχή των νευρώνων που συμβάλλουν θετικά στην κατεύθυνση του σωστού αποτελέσματος και αντίθετα, να αποδυναμωθούν οι δεσμοί εκείνων που συμβάλλουν αρνητικά. Αυτό ακριβώς επιτυγχάνεται με τον αλγόριθμο οπισθοδιάδοσης του σφάλματος (error back-propagation algorithm).

3. Μέθοδοι εκπαίδευσης των τεχνητών νευρωνικών δικτύων

3.1 Γενικά

Υπάρχουν πολλές μέθοδοι με τις οποίες μπορούμε να εκπαιδεύσουμε ένα τεχνητό νευρωνικό δίκτυο. Η κάθε μία από αυτές έχει σαφή πλεονεκτήματα και μειονεκτήματα. Καμία από αυτές δεν είναι πανάκεια, αν και, αυτή που χρησιμοποιείται συνηθέστερα σε πολυεπίπεδους αισθητήρες είναι ο αλγόριθμος οπισθοδιάδοσης του σφάλματος, είτε αυτούσιος, είτε με κάποιες παραλλαγές ή προσθήκες.

Γενικά, επιλέγουμε την κατάλληλη μέθοδο, ανάλογα με τη φύση του εκάστοτε προβλήματος και με τη δομή του συγκεκριμένου δικτύου, που διαλέξαμε για να το μελετήσουμε. Πολλές φορές χρησιμοποιούμε περισσότερες από μία μεθόδους μαζί για να πετύχουμε τη βέλτιστη επίδοση του δικτύου.

3.2 Αλγόριθμος οπισθοδιάδοσης του σφάλματος

Έστω ότι έχουμε έναν πολυεπίπεδο αισθητήρα με ένα ή περισσότερα κρυφά επίπεδα, m τιμές εισόδου και m_0 νευρώνες στο επίπεδο εξόδου του και ότι οι νευρώνες αυτοί λειτουργούν με κάποια παραγωγίσιμη συνάρτηση ενεργοποίησης. Έστω επίσης ότι, για να εκπαιδεύσουμε το δίκτυο αυτό, έχουμε στη διάθεσή μας ένα σύνολο N γνωστών παραδειγμάτων εισόδων – επιθυμητών εξόδων, δηλαδή ένα σύνολο δεδομένων

$$Data:\{\bar{x}(i),\bar{d}(i);i=1,2,\ldots,N\} \tag{16}$$

όπου $\bar{x}(i)=[x_1(i),x_2(i),\ldots,x_m(i)]^T$, το διάνυσμα των τιμών εισόδου και $\bar{d}(i)$ το διάνυσμα των επιθυμητών εξόδων αντίστοιχα, για το δεδομένο παράδειγμα *i*.

Ο στόχος μας είναι να βρούμε έναν κατάλληλο αλγόριθμο για να εκπαιδεύσουμε το δίκτυο αυτό, σύμφωνα με το σύνολο των δεδομένων. Σε κάθε βήμα της εκπαίδευσης, οι τιμές των συναπτικών βαρών μεταβάλλονται και η διαδικασία τερματίζεται, όταν κρίνουμε ότι το δίκτυο έχει "μάθει" τα παραδείγματα σε ικανοποιητικό βαθμό.

Το σφάλμα στην έξοδο του *j*-οστού νευρώνα, του επιπέδου εξόδου, κατά το *n*-οστό βήμα της επανάληψης του αλγόριθμου εκπαίδευσης δίνεται από τη σχέση

$$\boxed{e_j(n)=d_j(n)-y_j(n)} \tag{17}$$

Η στιγμιαία τιμή της *συνάρτησης ενέργειας σφάλματος* (*error energy*) για το σύνολο των νευρώνων του επιπέδου εξόδου του δικτύου, ορίζεται από τη σχέση

$$\boxed{\mathcal{E}(n)=\frac{1}{2}\sum_{j=1}^{n}e_j^2(n)} \tag{18}$$

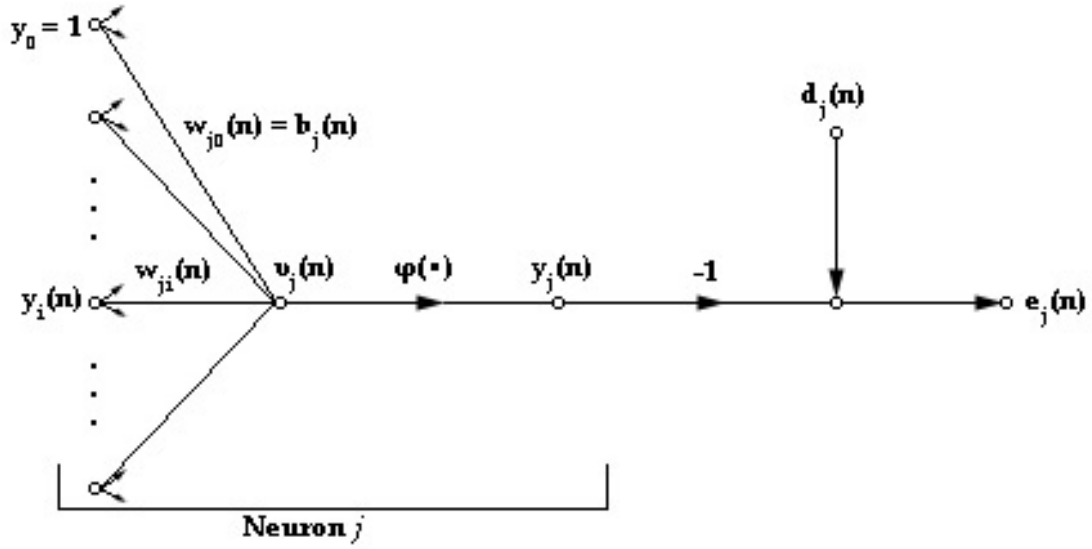
Η μέση τιμή της συνάρτησης ενέργειας σφάλματος για όλο το σύνολο των N δεδομένων, προκύπτει από τη σχέση

$$\boxed{\mathcal{E}_{av}=\frac{1}{N}\sum_{n=1}^{N}\mathcal{E}(n)} \tag{19}$$

Όπως φαίνεται από την παραπάνω σχέση, η $\mathcal{E}_{av}$ είναι μία συνάρτηση όλων των ελεύθερων παραμέτρων (free parameters) του νευρωνικού δικτύου, δηλαδή των συναπτικών βαρών και των μεροληψιών του. Επομένως, για ένα συγκεκριμένο σύνολο δεδομένων, η $\mathcal{E}_{av}$ συνιστά μία

συνάρτηση κόστους (cost function) και ως τέτοια είναι ένα μέτρο του πόσο καλά έχει εκπαιδευτεί το δίκτυο, βάσει του συνόλου αυτού.

Η βασική ιδέα του αλγόριθμου οπισθοδιάδοσης είναι να προσαρμόζονται τα συναπτικά βάρη, σε κάθε βήμα της επανάληψης, έτσι ώστε η συνάρτηση κόστους να μειώνεται. Στην πραγματικότητα επομένως, η προσαρμογή των βαρών γίνεται σύμφωνα με τα σφάλματα που υπολογίζονται σε κάθε βήμα της διαδικασίας, δηλαδή για κάθε δεδομένο παράδειγμα. Ο μέσος όρος της μεταβολής των βαρών σε όλο το σύνολο των σετ δεδομένων, που προκύπτει με αυτόν τον τρόπο, είναι επομένως μία εκτίμηση της μεταβολής που θα προέκυπτε, αν ελαχιστοποιούσαμε τη συνάρτηση κόστους όλου του συνόλου, όπως τη θεωρήσαμε παραπάνω.



Σχήμα 11: Ροή πληροφορίας για το νευρώνα *j*.

Όπως αναφέραμε παραπάνω, ο στόχος είναι, σε κάθε βήμα του αλγορίθμου, να μεταβάλλονται τα συναπτικά βάρη, έτσι ώστε να μειώνεται η συνάρτηση κόστους. Με άλλα λόγια, ξεκινάμε από μία αρχική εκτίμηση των συναπτικών βαρών $w_{ji}(0)$, εφαρμόζοντας μία διόρθωση παίρνουμε ένα νέο σετ βαρών $w_{ji}(1)$, εφαρμόζοντας ξανά μία δεύτερη διόρθωση παίρνουμε ένα δεύτερο σετ $w_{ji}(2)$ κ.ο.κ., έτσι ώστε σε κάθε επανάληψη να ισχύει

$$\boxed{\mathcal{E}(\bar{w}(n+1)) \leq \mathcal{E}(\bar{w}(n))} \qquad (20)$$

όπου $\bar{w}(n)$ ο πίνακας των βαρών στο βήμα n και $\bar{w}(n+1)$ ο διορθωμένος πίνακας στο βήμα $n+1$.

Σύμφωνα με τον αλγόριθμο οπισθοδιάδοσης, για να το πετύχουμε αυτό, θεωρούμε ότι, η μεταβολή στον πίνακα $\bar{w}(n)$ γίνεται στην κατεύθυνση

της *πλέου απότομης κατάβασης* (*steepest descent*), δηλαδή στην κατεύθυνση την αντίθετη προς αυτή του διανύσματος της κλίσης της $\mathcal{E}(w(n))$. Η διόρθωση επομένως στο βάρος $w_{ji}(n)$, που συνδέει το νευρώνα j του επιπέδου εξόδου του δικτύου, με το νευρώνα i, του τελευταίου κρυφού επιπέδου, κατά το βήμα n, δίνεται από τη σχέση

$$\boxed{\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}} \tag{21}$$

όπου η είναι η θετική σταθερά, που ονομάσαμε παράμετρος ρυθμού εκπαίδευσης. Η σχέση αυτή ονομάζεται *κανόνας του δέλτα* (*delta rule*). Η μεταβολή δηλαδή, γίνεται κατά την κατεύθυνση της κλίσης, ενώ το μείον το βάζουμε για να έχουμε μείωση της $\mathcal{E}(n)$. Αρκεί επομένως να βρούμε με τι ισούται η παράγωγος $\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$.

Σύμφωνα με τον κανόνα της αλυσίδας του διαφορικού λογισμού, η παράγωγος αυτή γράφεται

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial \upsilon_j(n)} \frac{\partial \upsilon_j(n)}{\partial w_{ji}(n)} \tag{22}$$

όπου $e_j(n)$ το σφάλμα στην έξοδο του νευρώνα j, $y_j(n)$ η έξοδος του νευρώνα j, και $\upsilon_j(n)$ το τοπικό πεδίο του νευρώνα j, όλα υπολογισμένα κατά το n-οστό βήμα του αλγόριθμου, όπως μπορούμε να δούμε και στο Σχήμα 11.

Αλλά, παραγωγίζοντας τη σχέση (18), παίρνουμε

$$\boxed{\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} = e_j(n)} \tag{23}$$

ενώ από τη σχέση (17), έχουμε

$$\boxed{\frac{\partial e_j(n)}{\partial y_j(n)} = -1} \tag{24}$$

Επιπλέον, όπως φαίνεται και από το Σχήμα11 στη σελ. 14, το τοπικό πεδίο $\upsilon_j(n)$ του νευρώνα j, κατά το βήμα n, δίνεται από τη σχέση

$$\upsilon_j = \sum_{i=0}^{k} w_{ji}(n) y_i(n) \tag{25}$$

όπου k, το πλήθος των νευρώνων στο προτελευταίο επίπεδο του δικτύου, δηλαδή του τελευταίου κρυφού επίπεδου, ενώ η έξοδος $y_j(n)$ του νευρώνα j, κατά το βήμα n, δίνεται από τη σχέση

$$y_j(n) = \varphi_j(\upsilon_j(n)) \tag{26}$$

Από τις σχέσεις (25) και (26), παίρνουμε αντίστοιχα τις

$$\boxed{\frac{\partial y_j(n)}{\partial \upsilon_j(n)} = \varphi'(\upsilon_j(n))} \tag{27}$$

και

$$\boxed{\frac{\partial \upsilon_j(n)}{\partial w_{ji}(n)} = y_i(n)} \tag{28}$$

όπου $y_i(n)$ η έξοδος του νευρώνα i του τελευταίου κρυφού επίπεδου.

Όπως παρατηρούμε από τη σχέση (27), είναι απαραίτητη η γνώση της παραγώγου της συνάρτησης ενεργοποίησης. Αυτός λοιπόν είναι ένας από τους λόγους που χρησιμοποιούμε παραγωγίσιμη συνάρτηση ενεργοποίησης, αντί της βηματικής συνάρτησης.

Βάσει των σχέσεων (23), (24), (27) και (28) επομένως, η σχέση (21) γράφεται

$$\boxed{\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)} \tag{29}$$

όπου

$$\begin{aligned} \delta_j(n) &= -\frac{\partial \mathcal{E}(n)}{\partial \upsilon_j(n)} \\ \dot{\eta}\ \delta_j(n) &= -\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial \upsilon_j(n)} \\ \dot{\eta}\ &\boxed{\delta_j(n) = e_j(n)\varphi_j'(\upsilon_j(n))} \end{aligned} \tag{30}$$

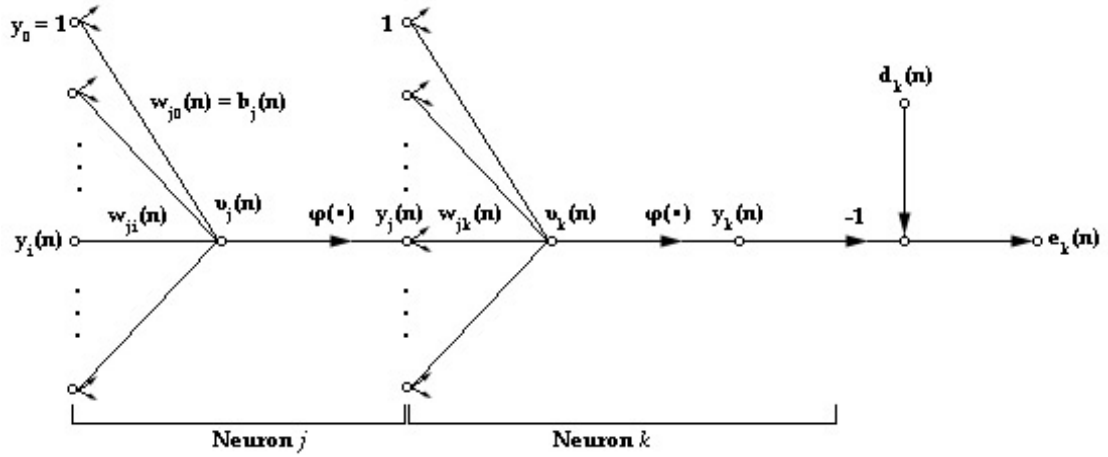
η λεγόμενη *τοπική βαθμίδα κλίσης* (*local gradient*).

Από τη σχέση (30), βλέπουμε ότι, για να υπολογίσουμε τη τοπική βαθμίδα κλίσης και στη συνέχεια τη μεταβολή στο συναπτικό βάρος, πρέπει να ξέρουμε το σφάλμα στην έξοδο του νευρώνα, $e_j(n)$. Όταν ο υπό εξέταση νευρώνας είναι νευρώνας του επιπέδου εξόδου του δικτύου, τότε ο υπολογισμός του σφάλματος γίνεται εύκολα, βάσει της σχέσης (17). Στην περίπτωση όμως που ο νευρώνας ανήκει σε κάποιο κρυφό

επίπεδο, τότε το σφάλμα δεν μπορεί να υπολογιστεί απευθείας, καθώς δεν ξέρουμε ποιο θα ήταν το επιθυμητό αποτέλεσμα για έναν τέτοιο νευρώνα. Άρα, όταν ο υπό εξέταση νευρώνας ανήκει σε κρυφό επίπεδο, η σχέση (30) που δίνει την τοπική βαθμίδα $\delta_j(n)$, πρέπει να τροποποιηθεί. Συγκεκριμένα, επειδή το σφάλμα στην περίπτωση αυτή δεν μπορεί να υπολογιστεί απευθείας, αναγκαστικά θα υπολογιστεί έμμεσα, βάσει των σφαλμάτων του αμέσως επόμενου επιπέδου.

Έτσι λοιπόν, θεωρώντας την περίπτωση που ο νευρώνας ανήκει στο τελευταίο κρυφό επίπεδο, δηλαδή το προτελευταίο επίπεδο του δικτύου, για την οποία η σχέση (30) τροποποιείται και γίνεται

$$\begin{aligned} \delta_j(n) &= -\frac{\partial \mathcal{E}(n)}{\partial \upsilon_j(n)} \\ \dot{\eta}\ \delta_j(n) &= -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial \upsilon_j(n)} \\ \dot{\eta}\ &\boxed{\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} \varphi_j'(\upsilon_j(n))} \end{aligned} \tag{31}$$



Σχήμα 12: Ροή πληροφορίας για το νευρώνα *k* σε σχέση με το νευρώνα *j*.

Αρκεί επομένως να υπολογίσουμε την παράγωγο $\frac{\partial \mathcal{E}(n)}{\partial y_j(n)}$. Από το Σχήμα 12, βλέπουμε ότι είναι

$$\boxed{\mathcal{E}(n) = \frac{1}{2}\sum_{k=1}^{m_o} e_k^2(n)} \tag{32}$$

όπου $e_k(n)$ το σφάλμα του k-οστού νευρώνα του επιπέδου εξόδου του δικτύου. Επομένως, παραγωγίζοντας τη σχέση (32) ως προς $y_j(n)$, έχουμε

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)}$$
$$\dot{\eta}\ \frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial \upsilon_k(n)} \frac{\partial \upsilon_k(n)}{\partial y_j(n)} \tag{33}$$

Αλλά, από το Σχήμα 12, σελ. 17, παρατηρούμε επιπλέον ότι είναι

$$e_k(n) = d_k(n) - y_k(n)$$
$$\dot{\eta}\ \boxed{e_k(n) = d_k(n) - \varphi_k(\upsilon_k(n))} \tag{34}$$

και

$$\boxed{\upsilon_k = \sum_{j=0}^{m_h} w_{kj}(n) y_j(n)} \tag{35}$$

όπου m_h είναι ο αριθμός των εισόδων που δέχεται ο νευρώνας k, δηλαδή ο αριθμός των νευρώνων του τελευταίου κρυφού επιπέδου. Η άθροιση αρχίζει από το μηδέν, για να συνυπολογιστεί και η μεροληψία που εφαρμόζεται στο νευρώνα, συνδεδεμένη με το βάρος w_{k0}.

Από τη σχέση (34), με παραγώγιση ως προς $\upsilon_k(n)$, προκύπτει η

$$\boxed{\frac{\partial e_k(n)}{\partial \upsilon_k(n)} = -\varphi_k'(\upsilon_k(n))} \tag{36}$$

ενώ από τη σχέση (35), με παραγώγιση ως προς $y_j(n)$, παίρνουμε τη

$$\boxed{\frac{\partial \upsilon_k(n)}{\partial y_j(n)} = w_{kj}(n)} \tag{37}$$

Αντικαθιστώντας τις (36) και (37) στη (33), βρίσκουμε ότι η ζητούμενη παράγωγος δίνεται από τη σχέση

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = -\sum_k e_k(n) \varphi_k'(\upsilon_k(n)) w_{kj}(n)$$
$$\dot{\eta}\ \boxed{\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_k \delta_k(n) w_{kj}(n)} \tag{38}$$

όπου χρησιμοποιήσαμε τη σχέση 30) που δίνει την τοπική βαθμίδα $\delta_k(n)$, όταν ο νευρώνας k ανήκει στο επίπεδο εξόδου. Τέλος, αντικαθιστώντας τη σχέση (38) στη σχέση (31), προκύπτει η σχέση για

την τοπική βαθμίδα $\delta_j(n)$ στην περίπτωση που ο νευρώνας βρίσκεται σε κρυφό επίπεδο

$$\boxed{\delta_j(n) = \varphi'_j(\upsilon_j(n))\sum_k \delta_k(n) w_{kj}(n)} \quad (39)$$

Ανακεφαλαιώνοντας, η διόρθωση στο συναπτικό βάρος $w_{ji}(n)$, που συνδέει το νευρώνα *j* με το νευρώνα *i*, δίνεται γενικά από τη σχέση

$$\boxed{\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)} \quad (40)$$

και διακρίνουμε δύο περιπτώσεις για την τοπική βαθμίδα $\delta_j(n)$:

1. Όταν ο νευρώνας *j* ανήκει στο επίπεδο εξόδου του δικτύου, τότε

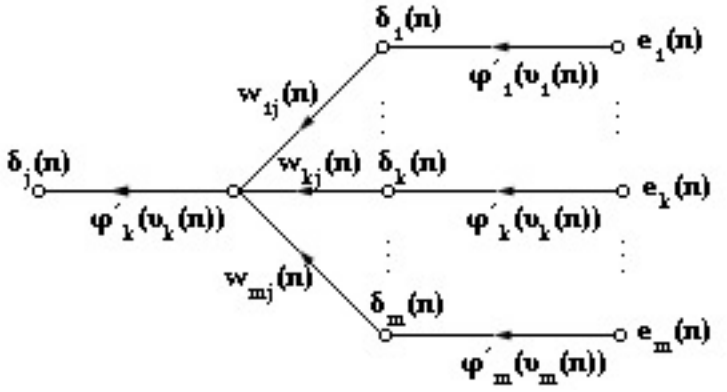
$$\boxed{\delta_j(n) = e_j(n)\varphi'_j(\upsilon_j(n))} \quad (30)$$

2. Όταν ο νευρώνας *j* ανήκει σε κρυφό επίπεδο του δικτύου, τότε

$$\boxed{\delta_j(n) = \varphi'_j(\upsilon_j(n))\sum_k \delta_k(n) w_{kj}(n)} \quad (39)$$

όπου ο δείκτης *k* συμβολίζει τους νευρώνες του αμέσως επόμενου επιπέδου.

Στο Σχήμα 13 παρακάτω, φαίνεται σχηματικά η σχέση που δίνει τη τοπική βαθμίδα $\delta_j(n)$ ενός νευρώνα σε κρυφό επίπεδο, από τα $\delta_k(n)$ του αμέσως επόμενου επιπέδου.



Σχήμα 13: Υπολογισμός του $\delta_j(n)$ για έναν νευρώνα κρυφού επιπέδου

Κατά την εκπαίδευση επομένως ενός πολυεπίπεδου αισθητήρα με τον αλγόριθμο οπισθοδιάδοσης, σε κάθε βήμα της επανάληψης, λαμβάνουν χώρα τα παρακάτω:

Α. Στάδιο 1ο, ευθύ πέρασμα

Τροφοδοτούμε το δίκτυο με το n-οστό σετ τιμών εισόδου. Οι τιμές αυτές περνάνε στο πρώτο κρυφό επίπεδο, οι νευρώνες του οποίου τις επεξεργάζονται και δίνουν τις τιμές εξόδου. Οι τελευταίες περνάνε στο δεύτερο κρυφό επίπεδο – αν υπάρχει – κ.ο.κ. και τελικά το επεξεργασμένο σήμα φτάνει στο επίπεδο εξόδου, το οποίο μας δίνει τις τελικές εξόδους του δικτύου. Συγκρίνουμε τις τιμές της τελικής εξόδου με το επιθυμητό αποτέλεσμα και βρίσκουμε το σφάλμα για κάθε μία από αυτές.

Β. Στάδιο 2ο, ανάποδο πέρασμα

Έχοντας υπολογίσει τα σφάλματα στην έξοδο του δικτύου, υπολογίζουμε τις τοπικές βαθμίδες για το επίπεδο εξόδου. Βάσει των τελευταίων, υπολογίζουμε στη συνέχεια τις τοπικές βαθμίδες του τελευταίου κρυφού επιπέδου, έπειτα αυτές του προτελευταίου κρυφού επιπέδου – αν υπάρχει – κ.ο.κ., μέχρι να υπολογιστούν και αυτές του πρώτου κρυφού επιπέδου. Γνωρίζοντας τις τοπικές βαθμίδες για κάθε νευρώνα του δικτύου, υπολογίζουμε τη διόρθωση σε κάθε συναπτικό βάρος.

Τα δύο αυτά στάδια επαναλαμβάνονται, μέχρι τα αποτελέσματα του δικτύου να συμπίπτουν με τα επιθυμητά με ικανοποιητικό βαθμό ακρίβειας.

Όπως είδαμε παραπάνω, ο αλγόριθμος οπισθοδιάδοσης οδηγεί σε μία τροχιά στον χώρο των βαρών, η οποία αποτελεί μία προσέγγιση αυτής που θα ακολουθούσαμε, αν ελαχιστοποιούσαμε τη μέση τιμή της συνάρτησης κόστους, $\mathcal{E}_{av}$. Η μορφή της τροχιά αυτής εξαρτάται από την παράμετρο ρυθμού εκπαίδευσης η. Όσο μικρότερη είναι η η, τόσο μικρότερες είναι οι διορθώσεις στα βάρη, σε κάθε βήμα, και τόσο πιο ομαλή είναι η τροχιά που ακολουθούμε. Όμως, η σταθερά η επηρεάζει και την ταχύτητα σύγκλισης του αλγόριθμου οπισθοδιάδοσης. Όταν αυτή είναι μικρή ο αλγόριθμος συγκλίνει με αργούς συνήθως ρυθμούς. Αν όμως αυξήσουμε την παράμετρο ρυθμού εκπαίδευσης, ώστε να έχουμε πιθανότατα ταχύτερη σύγκλιση, τότε οι διορθώσεις σε κάθε βήμα μπορεί να γίνουν πολύ μεγάλες και να οδηγήσουν σε αστάθεια του αλγόριθμου.

Ένας τρόπος να μπορέσουμε να αυξήσουμε την ταχύτητα σύγκλισης του αλγόριθμου, αποφεύγοντας όμως την πιθανή αστάθεια, είναι να τροποποιήσουμε τη σχέση (29)

$$\boxed{\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)} \tag{29}$$

και να την αντικαταστήσουμε με τη σχέση

$$\boxed{\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)} \qquad (41)$$

όπου α είναι μία σταθερά, που συνήθως είναι θετική και ονομάζεται *σταθερά της ορμής* (*momentum constant*). Η σχέση (41) ονομάζεται *γενικευμένος κανόνας δέλτα* (*generalized delta rule*) και ανάγεται στη σχέση (29), για $\alpha = 0$.

3.3 Εφαρμογή του αλγόριθμου οπισθοδιάδοσης – Πείραμα XOR

Για να μελετήσουμε τον αλγόριθμο οπισθοδιάδοσης στην πράξη, κατασκευάσαμε ένα τεχνητό νευρωνικό δίκτυο στον υπολογιστή και το εκπαιδεύσαμε, σύμφωνα με τη λογική πράξη XOR. Συγκεκριμένα, υλοποιήσαμε το νευρωνικό δίκτυο του Σχήματος 9, σελ 11. Το δίκτυο επομένως δέχεται δύο εισόδους, έχει ένα κρυφό επίπεδο με δύο νευρώνες και έχει ένα νευρώνα στο επίπεδο εξόδου. Κάθε νευρώνας είναι μη γραμμικός και λειτουργεί με τη λογιστική συνάρτηση ως συνάρτηση ενεργοποίησης.

Οι αρχικές τιμές για τα βάρη και τις μεροληψίες επιλέχθηκαν από μία ομοιόμορφη κατανομή τυχαίων αριθμών στο διάστημα (-1,1). Στη συνέχεια, οι τιμές αυτές τροποποιήθηκαν, βάσει του αλγόριθμου οπισθοδιάδοσης. Για το πείραμα αυτό έχουμε τέσσερα διαφορετικά σετ εισόδων – επιθυμητής εξόδου, που φαίνονται στον Πίνακα 1. Τα σετ αυτά χρησιμοποιήθηκαν πολλές φορές μέχρι να επιτευχθεί ο επιθυμητός στόχος. Κάθε φορά που εξαντλείται η τετράδα των σετ προτύπων, αντιστοιχεί σε μία *εποχή* (*epoch*) της εκπαίδευσης. Η εκπαίδευση τερματίστηκε όταν το σφάλμα της τιμής της εξόδου μειώθηκε στο καθορισμένο όριο.

Ο κώδικας που εφαρμόσαμε βρίσκεται στο αρχείο NNXOR.cpp. Αρχικά, θέσαμε την παράμετρο ρυθμού εκπαίδευσης, $\eta = 0.01$ και τη σταθερά ορμής, $\alpha = 0$. Ο αλγόριθμος συνέκλινε σε 24776 επαναλήψεις και έδωσε, ως λύση του προβλήματος, το παρακάτω σετ τιμών για τα βάρη και τις μεροληψίες του δικτύου

$$\begin{cases} w_{11} = 1.49731 & w_{12} = 1.42548 & b_1 = -1.96077 \\ w_{21} = 3.68129 & w_{22} = 3.30679 & b_2 = -0.539174 \\ w_{31} = -3.32491 & w_{32} = 3.16934 & b_3 = -0.756056 \end{cases} \qquad (42)$$

Στη συνέχεια, επαναλάβαμε το πείραμα με τις ίδιες αρχικές τιμές των βαρών και των μεροληψιών για διάφορες τιμές της παραμέτρου ρυθμού εκπαίδευσης και της σταθεράς της ορμής και παρατηρήσαμε αν και πως αλλάζει ο αριθμός των επαναλαμβανόμενων βημάτων της διαδικασίας εκπαίδευσης και αν ο αλγόριθμος συγκλίνει ή όχι. Τα αποτελέσματα φαίνονται στον Πίνακα 2, που ακολουθεί:

παράμετρος ρυθμού εκπαίδευσης (η)	σταθερά της ορμής (α)	αριθμός επαναλήψεων	συγκλίνει
0.01	0.0	24776	ναι
0.1	0.0	84262	ναι
0.2	0.0	1389	ναι
0.3	0.0	755	ναι
0.4	0.0	518	ναι
0.5	0.0	200.000	όχι
0.5	0.3	302	ναι
0.5	0.9	149	ναι
0.6	0.0	163	ναι
0.7	0.0	111	ναι

Πίνακας 2: Διερεύνηση της επίπτωσης των σταθερών η και α στη σύγκλιση του αλγόριθμου και την ταχύτητά της

Από τον Πίνακα 2, βλέπουμε ότι η δράση τόσο της σταθεράς του ρυθμού εκπαίδευσης η, όσο και της σταθεράς της ορμής α δεν είναι πάντα η ίδια. Δεν υπάρχει ρητή σχέση μεταξύ της η και της ταχύτητας σύγκλισης. Συνήθως, όσο αυξάνεται η η, τόσο ταχύτερη σύγκλιση έχουμε, αλλά όχι πάντα. Αλλά ούτε και η επίδραση της σταθεράς της ορμής είναι σταθερή. Σε κάποιες περιπτώσεις φαίνεται να βοηθάει και σε άλλες δεν χρειάζεται καθόλου. Κάθε φορά λοιπόν, σε κάθε διαφορετικό πρόβλημα, θα πρέπει να διερευνήσουμε τη συμπεριφορά του αλγόριθμου για διάφορες τιμές των παραμέτρων αυτών και να επιλέξουμε αυτές που εμείς θεωρούμε ως πιο κατάλληλες.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η περίπτωση με $\eta = 0.5$ και $\alpha = 0$. Για την περίπτωση αυτή ο αλγόριθμος συγκλίνει σε λάθος αποτελέσματα, τα οποία φαίνονται στον Πίνακα 3.

x_1	x_2	x_1 XOR x_2	Αποτέλεσμα δικτύου
0	0	0	$3.43184\ 10^{-5}$
0	1	1	0.99998
1	0	1	0.333304
1	1	0	0.666681

Πίνακας 3: Αποτελέσματα για XOR με $\eta = 0.5$ και $\alpha = 0$

Σε αυτή την περίπτωση, η εκπαίδευση αποτυγχάνει πλήρως. Όσο και να εκπαιδευτεί το δίκτυο, με αυτή την αρχικοποίηση των συναπτικών βαρών και των μεροληψιών και με αυτή την τιμή της παραμέτρου ρυθμού εκπαίδευσης, ποτέ δε θα φτάσει στο επιθυμητό αποτέλεσμα.

Αν επαναλάβουμε το όλο πείραμα της εκπαίδευσης, αρχίζοντας από πολλά διαφορετικά σετ αρχικών τιμών για τα βάρη και τις μεροληψίες,

θα διαπιστώσουμε ότι υπάρχουν και άλλες τέτοιες περιπτώσεις για τις οποίες το δίκτυο συγκλίνει σε λάθος αποτελέσματα. Αυτό είναι και το βασικό μειονέκτημα του αλγόριθμου οπισθοδιάδοσης. Δεν υπάρχει εγγύηση ότι θα συγκλίνει πάντα σε ένα σετ συναπτικών βαρών και μεροληψιών, για τα οποία το δίκτυο να δίνει σωστά αποτελέσματα.

Η βασική ιδέα του αλγόριθμου εκπαίδευσης είναι, όπως είδαμε, να ελαχιστοποιηθεί η συνάρτηση κόστους, δηλαδή η τετραγωνική ρίζα του αθροίσματος των τετραγώνων των σφαλμάτων στην έξοδο του δικτύου. Η συνάρτηση αυτή έχει όχι μόνο ένα τέτοιο καθολικό ελάχιστο, αλλά πολλά, λόγω συμμετριών. Το πρόβλημα είναι ότι, εκτός από αυτά τα καθολικά ελάχιστα, στα οποία θέλουμε να φτάσουμε, έχει συνήθως και πολλά τοπικά ελάχιστα. Τα τοπικά αυτά ελάχιστα είναι συχνά αρκετά μεγάλα, ώστε αν η τροχιά που ακολουθεί ο αλγόριθμος πέσει μέσα σε ένα από αυτά, να παγιδευτεί εκεί και να πάρουμε λάθος αποτελέσματα. Γι' αυτό και θα πρέπει πάντα όταν εκπαιδεύουμε ένα νευρωνικό δίκτυο με τον αλγόριθμο οπισθοδιάδοσης να φροντίσουμε να ξεκινήσουμε από πολλά διαφορετικά σετ αρχικών τιμών.

Όπως είδαμε όμως στον Πίνακα 2, σελ 22, προσθέτοντας τη σταθερά της ορμής στους υπολογισμούς μας, στο συγκεκριμένο παράδειγμα, το δίκτυο καταφέρνει να φτάσει σε σωστό αποτέλεσμα σε μικρό αριθμό επαναλήψεων. Για $\alpha = 0.3$ συγκλίνει σε σωστή λύση σε 302 επαναλήψεις, ενώ για $\alpha = 0.9$ σε 149. Η σταθερά της ορμής στην περίπτωση αυτή αλλάζει τη τροχιά που ακολουθεί ο αλγόριθμος στο χώρο των βαρών και το τοπικό ελάχιστο στο οποίο παγιδεύονταν αποφεύγεται. Υπάρχουν και άλλοι τρόποι να αποφευχθούν τα τοπικά ελάχιστα. Ένας από τους πιο γνωστούς είναι και η μέθοδος του simulated annealing.

4. Προσέγγιση συναρτήσεων με πολυεπίπεδους αισθητήρες

4.1 Γενικά

Θεωρούμε έναν πολυεπίπεδο αισθητήρα που δέχεται m_0 εισόδους και έναν νευρώνα στο επίπεδο εξόδου. Μπορούμε να δούμε τη δράση ενός τέτοιου δικτύου ως μία απεικόνιση του Ευκλείδειου χώρου διάστασης m_0 των εισόδων, στον Ευκλείδειο χώρο των εξόδων. Η απεικόνιση αυτή θα είναι συνεχώς παραγωγίσιμη, αν το ίδιο είναι και η συνάρτηση ενεργοποίησης των νευρώνων που χρησιμοποιούμε.

Εκπαιδεύουμε το δίκτυο αυτό, με τον αλγόριθμο οπισθοδιάδοσης, βάσει ενός συνόλου προτύπων εισόδων – εξόδου, που προέρχονται από μία συνεχή απεικόνιση. Ο στόχος μας είναι η απεικόνιση που θα πάρουμε από το εκπαιδευμένο δίκτυο, να αποτελεί μία προσέγγιση της αρχικής απεικόνισης. Όπως και στις περισσότερες εφαρμογές που αφορούν

νευρωνικά δίκτυα, το ερώτημα που προκύπτει άμεσα είναι αν υπάρχει κάποιο κριτήριο σύμφωνα με το οποίο να επιλεχθεί το κατάλληλο πλήθος κρυφών επιπέδων για να πάρουμε την προσέγγιση αυτή.

Προς την κατεύθυνση αυτή μας βοηθάει το *θεώρημα καθολικής προσέγγισης* (*universal approximation theorem*), σύμφωνα με το οποίο ισχύουν τα παρακάτω:

Έστω $\varphi(\cdot)$ μία μη σταθερή, φραγμένη και μονότονα αύξουσα συνάρτηση. Συμβολίζουμε με I_{m_0} τον $[0,1]^{m_0}$ και το σύνολο των συνεχών συναρτήσεων στον I_{m_0} με $C(I_{m_0})$. Τότε, δοθείσας μίας συνάρτησης $f \in C(I_{m_0})$ και ενός αριθμού $\varepsilon > 0$, υπάρχει ένας ακέραιος M και σύνολα πραγματικών σταθερών α_i, b_i και $w_{i,j}$, όπου $i = 1,\ldots,m_1$ και $j = 1,\ldots,m_0$, τέτοιοι ώστε η συνάρτηση

$$F(x_1,\ldots,x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i\right) \tag{43}$$

να είναι μία προσέγγιση της συνάρτησης f, με άλλα λόγια να είναι

$$\left|F(x_1,\ldots,x_{m_0}) - f(x_1,\ldots,x_{m_0})\right| < \varepsilon \tag{44}$$

για κάθε σύνολο των $x_1, x_2, \ldots, x_{m_0}$ που ανήκουν στο χώρο των εισόδων.

Το θεώρημα αυτό μπορεί να εφαρμοστεί άμεσα στους πολυεπίπεδους αισθητήρες. Πράγματι, παρατηρούμε ότι η λογιστική συνάρτηση, καθώς και όλες οι σιγμοειδείς συναρτήσεις, που χρησιμοποιούνται ως συνάρτηση ενεργοποίησης των νευρώνων, πληροί τις προϋποθέσεις που επιβάλλονται στη $\varphi(\cdot)$, είναι δηλαδή μη σταθερά, φραγμένη και μονότονα αύξουσα. Επιπλέον, η σχέση (43) μπορεί να θεωρηθεί ότι δίνει το αποτέλεσμα στην έξοδο ενός δικτύου, για το οποίο ισχύουν τα ακόλουθα:

1. Δέχεται m_0 εισόδους, τις $x_1,\ldots,x_{m_0}$ και έχει ένα κρυφό επίπεδο με m_1 νευρώνες.
2. Ο νευρώνας *i*, του κρυφού επιπέδου, συνδέεται με τις εισόδους με τα βάρη $w_{i1},\ldots,w_{im_0}$ και έχει μεροληψία b_i.
3. Η έξοδός του είναι ένας γραμμικός συνδυασμός των εξόδων των νευρώνων του κρυφού επιπέδου, με συντελεστές τα βάρη $\alpha_1,\ldots,\alpha_{m_1}$, με τα οποία συνδέεται το επίπεδο εξόδου με το κρυφό επίπεδο του δικτύου.

Το θεώρημα καθολικής προσέγγισης είναι ένα θεώρημα ύπαρξης. Μας εξασφαλίζει ότι, όταν θέλουμε να εκπαιδεύσουμε έναν πολυεπίπεδο

αισθητήρα βάσει ενός σετ τιμών προερχόμενων από συνεχή συνάρτηση, ώστε να προσεγγίζει τη συνάρτηση αυτή, τότε ένα κρυφό επίπεδο είναι επαρκές. Δεν μπορούμε να ξέρουμε όμως αν ένα και μοναδικό κρυφό επίπεδο είναι η καλύτερη λύση από την άποψη της ταχύτητας της εκπαίδευσης, ούτε από την άποψη της ικανότητας που θα έχει το δίκτυο να εξάγει γενικά χαρακτηριστικά της συνάρτησης.

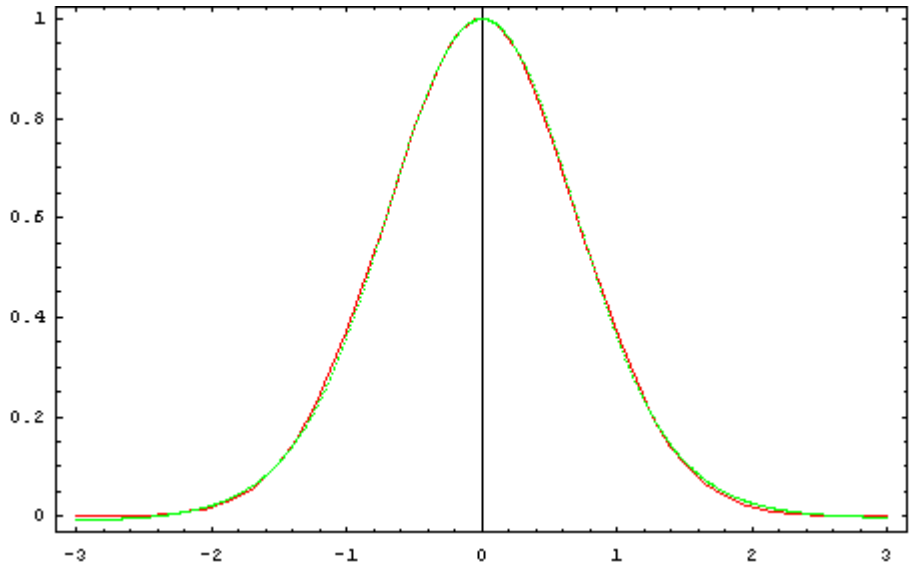
4.2 Προσέγγιση της e^{-x^2} με πολυεπίπεδο αισθητήρα

Με στόχο να προσεγγίσουμε τη συνάρτηση e^{-x^2}, κατασκευάσαμε έναν πολυεπίπεδο αισθητήρα με μία είσοδο, ένα κρυφό επίπεδο με δύο νευρώνες και έναν νευρώνα στο επίπεδο εξόδου. Για να εκπαιδεύσουμε το δίκτυο αυτό, χρησιμοποιήσαμε ένα σύνολο από έντεκα σετ διαφορετικών τιμών εισόδου – εξόδου, οι οποίες φαίνονται στον Πίνακα 4, παρακάτω

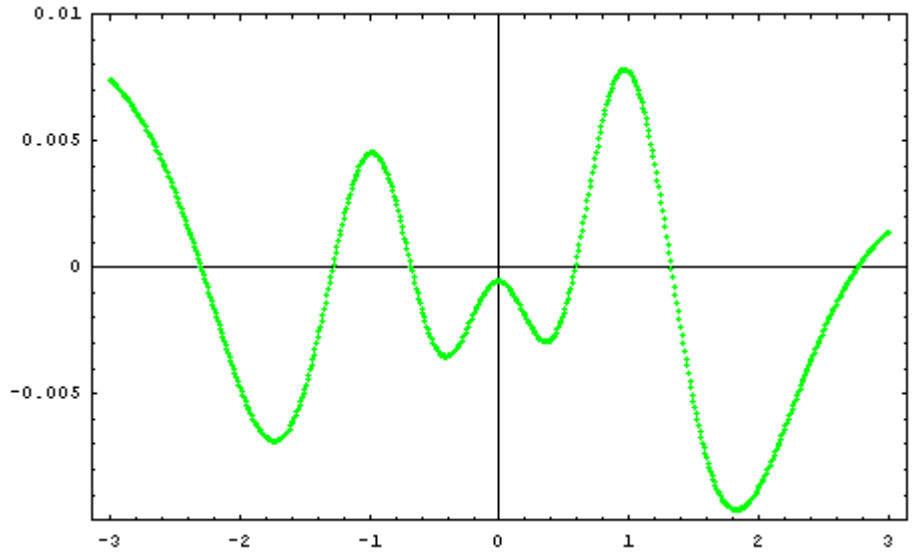
x_{in}	x_{out}	x_{in}	x_{out}
-2.7273	0.00059	0.54546	0.7427
-2.1818	0.0086	1.0909	0.3042
-1.6364	0.0086	1.6364	0.0086
-1.0909	0.3042	2.1818	0.0086
-0.54546	0.7427	2.7273	0.00059
0.0	1.0	-	-

Πίνακας 4: Τα σετ τιμών για την προσέγγιση της e^{-x^2}

Θέσαμε την παράμετρο ρυθμού εκπαίδευσης, $\eta = 0.4$ και τη σταθερά ορμής, $\alpha = 0.3$. Αφού εκπαιδεύσαμε το δίκτυο σε έναν βαθμό επιθυμητής ακρίβειας, για να ελέγξουμε την ποιότητα των αποτελεσμάτων, πήραμε τιμές στο διάστημα [-3,3], με βήμα 0.01 και κάναμε τη γραφική παράσταση. Στο Σχήμα 14 που ακολουθεί φαίνεται η σύγκριση των αποτελεσμάτων της προσέγγισης που δίνει το νευρωνικό δίκτυο, με τις πραγματικές τιμές της συνάρτησης e^{-x^2}, ενώ στο Σχήμα 15, φαίνεται το σφάλμα της μεθόδου.

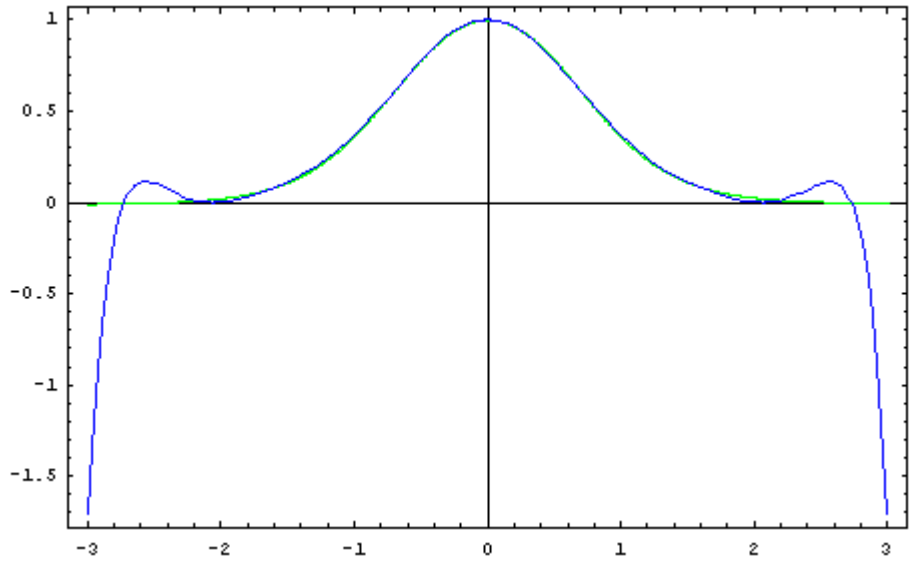


Σχήμα 14: Σύγκριση των αποτελεσμάτων του νευρωνικού δικτύου, με την e^{-x^2}

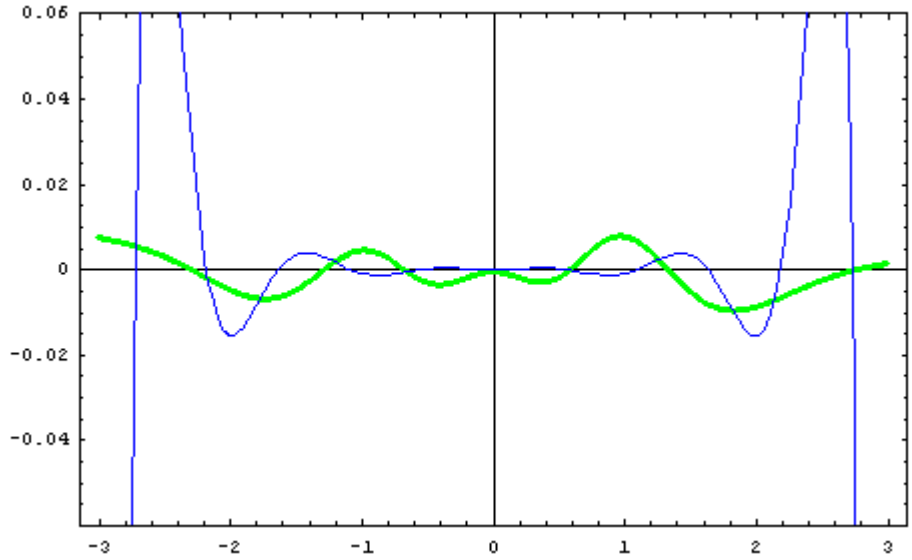


Σχήμα 15: Το σφάλμα της μεθόδου του νευρωνικού δικτύου, για την e^{-x^2}

Τέλος, συγκρίναμε τα αποτελέσματα της μεθόδου του νευρωνικού δικτύου, με τη μέθοδο της πολυωνυμικής προσέγγισης. Η σύγκριση των αποτελεσμάτων και των σφαλμάτων των δύο μεθόδων φαίνονται στα Σχήματα 16 και 17.



Σχήμα 16: Σύγκριση της μεθόδου του νευρωνικού δικτύου, με την πολυωνυμική



Σχήμα 17: Σύγκριση του σφάλματος της μεθόδου του νευρωνικού δικτύου με αυτό της πολυωνυμικής

Παρατηρούμε ότι το δίκτυο προσεγγίζει τη συνάρτηση με ικανοποιητική ακρίβεια, ακόμα και στις περιοχές όπου η πολυωνυμική προσέγγιση δεν είναι πολύ καλή.

Στην πράξη, σχεδόν ποτέ δε χρησιμοποιούνται παραπάνω από δύο κρυφά επίπεδα για την προσέγγιση μιας συνάρτησης. Στις περισσότερες περιπτώσεις χρησιμοποιείται μόνο ένα κρυφό επίπεδο και μόνο σε εκείνες για τις οποίες η συνάρτηση που ζητάμε να προσεγγίσουμε παρουσιάζει μεμονωμένες ασυνέχειες χρειάζεται και δεύτερο. Συμπερασματικά, ένας πολυεπίπεδος αισθητήρας μπορεί να μάθει όλες τις συναρτήσεις, αρκεί να αυτές να είναι ορισμένες σε συμπαγές σύνολο και να είναι ντετερμινιστικές.

Όσο για τον αριθμό των νευρώνων που θα χρησιμοποιήσουμε στο κάθε κρυφό επίπεδο, υπάρχουν κάποιοι εμπειρικοί κανόνες, αλλά δεν αρμόζουν σε όλες τις περιπτώσεις. Πρακτικά, η σίγουρα χρονοβόρα, αλλά πιθανότατα βέλτιστη λύση είναι να αρχίσουμε με τον μικρότερο δυνατό αριθμό νευρώνων, που είναι συνήθως δύο, και να ανεβάζουμε το πλήθος παρατηρώντας τις αλλαγές στη συμπεριφορά του δικτύου. Κρατάμε τον αριθμό για τον οποίο θεωρούμε ότι η επίδοση του δικτύου είναι η καλύτερη.

5. Πρόβλεψη και ανακατασκευή του χώρου φάσεων χρονοσειρών

5.1 Γενικά

Το μεγάλο πλεονέκτημα των νευρωνικών δικτύων όσον αφορά την πρόβλεψη χρονοσειρών είναι το γεγονός ότι η επίδοσή τους σε περιπτώσεις ύπαρξης θορύβου και χαοτικής συμπεριφοράς είναι καλύτερη από αυτή άλλων γνωστών μεθόδων.

Ο πρώτος στόχος μας είναι, έχοντας στη διάθεσή μας ένα σύνολο τιμών μιας χρονοσειράς, να εκπαιδεύσουμε το νευρωνικό δίκτυο, ώστε να πάρουμε μια πρόβλεψη για τις επόμενες τιμές. Ο δεύτερος στόχος μας είναι να προσπαθήσουμε να ανακατασκευάσουμε το χώρο φάσης, ώστε να εξάγουμε γενικότερες πληροφορίες για το σύστημα που γεννά τη χρονοσειρά.

Έστω λοιπόν, ότι έχουμε ένα σύνολο διαδοχικών όρων της χρονοσειράς και θέλουμε να τη μελετήσουμε με έναν πολυεπίπεδο αισθητήρα. Η βασική ιδέα είναι να χρησιμοποιήσουμε m_0 διαδοχικές τιμές της χρονοσειράς για να προβλέψουμε την επόμενη. Το πόσες ακριβώς εισόδους, πόσα κρυφά επίπεδα και πόσους νευρώνες στο κάθε επίπεδο θα έχουμε, εξαρτάται από το εκάστοτε πρόβλημα. Τέλος, στο επίπεδο εξόδου παίρνουμε τόσους νευρώνες, όσες τιμές θέλουμε να προβλέψουμε. Συνήθως προβλέπουμε μόνο την επόμενη τιμή της χρονοσειράς, οπότε έχουμε μόνο έναν νευρώνα στο επίπεδο εξόδου.

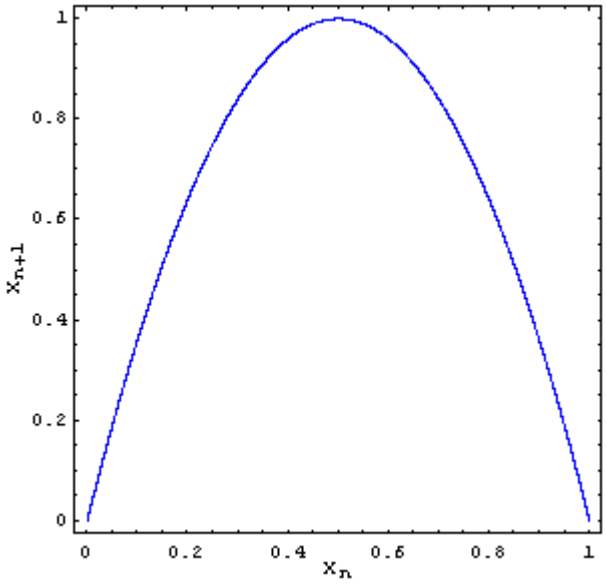
5.2 Παράδειγμα πρόβλεψης και ανακατασκευής του χώρου φάσης για μία χαοτική χρονοσειρά - Logistic map

Θα μελετήσουμε τη χρονοσειρά που προέρχεται από τη λογιστική απεικόνιση, η οποία μαθηματικά περιγράφεται από τον τύπο

$$\boxed{x_{n+1} = r x_n (1 - x_n)} \tag{45}$$

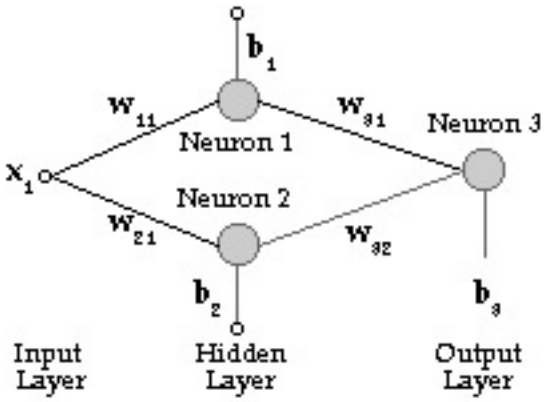
Παίρνουμε ένα σύνολο 10000 τιμών της λογιστικής απεικόνισης, για $r = 4$. Για την τιμή αυτή της παραμέτρου, η απεικόνιση εμφανίζει

χαοτική συμπεριφορά. Κάνοντας τη γραφική παράσταση του x_{n+1} συναρτήσει του x_n, παίρνουμε τον ελκυστή της λογιστικής απεικόνισης, που φαίνεται στο Σχήμα 18.



Σχήμα 18: Ο ελκυστής της λογιστικής απεικόνισης

Για να μελετήσουμε τη χρονοσειρά αυτή με έναν πολυεπίπεδο αισθητήρα, χρησιμοποιούμε μία τιμή εισόδου, ένα κρυφό επίπεδο με δύο νευρώνες και έναν νευρώνα στο επίπεδο εξόδου, όπως αυτό που φαίνεται στο Σχήμα 19.



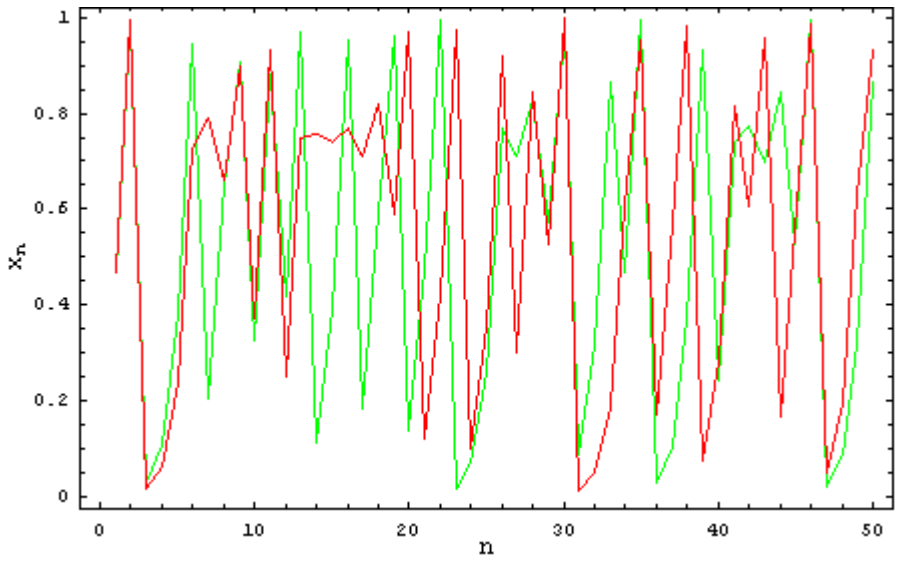
Σχήμα 19: Ο πολυεπίπεδος αισθητήρας για τη λογιστική απεικόνιση

Καταλήξαμε σε αυτή τη δομή του δικτύου, μετά από δοκιμές ως προς την ακρίβεια των αποτελεσμάτων και την ταχύτητα σύγκλισης του αλγόριθμου εκπαίδευσης.

Αρχικά, χωρίζουμε το σύνολο των δεδομένων τιμών σε δύο σύνολα, σε αυτό που θα χρησιμοποιήσουμε για την εκπαίδευση του δικτύου και σε αυτό που θα χρησιμοποιήσουμε για να ελέγξουμε την αξιοπιστία της

εκπαίδευσης αυτής. Ο διαχωρισμός γίνεται με τυχαίο τρόπο για να έχουμε αντιπροσωπευτικό δείγμα κατά την εκπαίδευση.

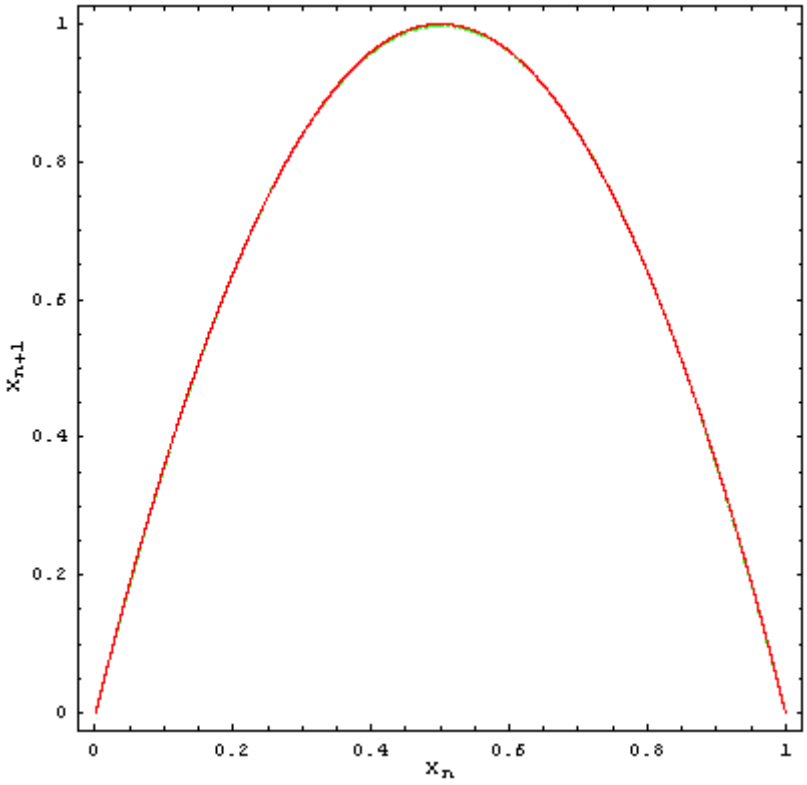
Εκπαιδεύουμε και ελέγχουμε την αξιοπιστία του δίκτυο και στη συνέχεια παίρνουμε πρόβλεψη για τις επόμενες 1000 τιμές. Τα αποτελέσματα της πρόβλεψης φαίνονται στο Σχήμα 20.



Σχήμα 20: Σύγκριση της πρόβλεψης με τη πραγματική τροχιά. Η χρονοσειρά που προβλέπει το δίκτυο φαίνεται με πράσινο, ενώ η πραγματική με κόκκινο

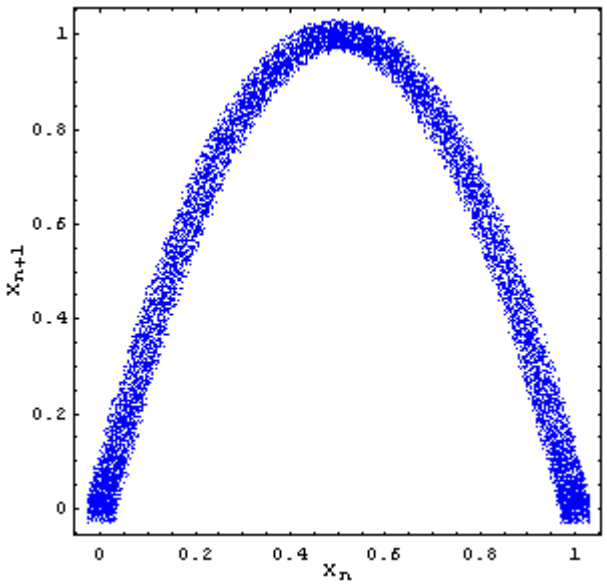
Όπως βλέπουμε από το παραπάνω σχήμα, το δίκτυο καταφέρνει να προβλέψει τις 3-4 πρώτες τιμές, ενώ μετά οι προβλεπόμενες τιμές ξεφεύγουν. Το δίκτυο αυτό επομένως καταφέρνει βραχυπρόθεσμα να κάνει σωστή πρόβλεψη.

Από τις 1000 τιμές που προβλέψαμε, ανακατασκευάσαμε τον ελκυστή της απεικόνισης. Η σύγκριση του πραγματικού ελκυστή με αυτόν που προβλέπει το νευρωνικό δίκτυο φαίνεται στο Σχήμα 21. Από το σχήμα αυτό παρατηρούμε ότι πρακτικά, ο ανακατασκευασμένος ελκυστής συμπίπτει πλήρως με τον ελκυστή της λογιστικής απεικόνισης (Σχήμα 18) από την οποία προέρχονται τα αρχικά δεδομένα μας.



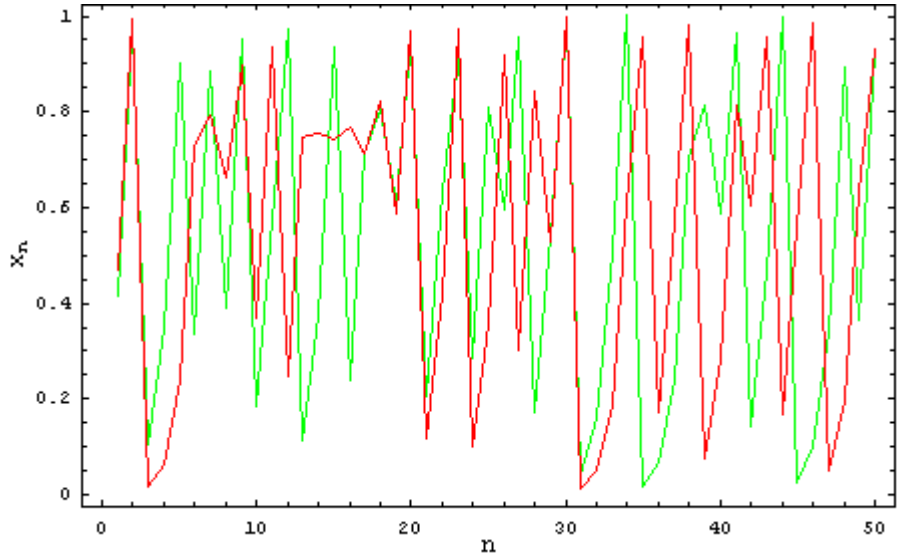
Σχήμα 21: Σύγκριση του ανακατασκευασμένου ελκυστή (πράσινο) με τον πραγματικό (κόκκινο)

Στη συνέχεια, προσθέτουμε στατιστικό θόρυβο στα αρχικά δεδομένα της χρονοσειράς. Στο Σχήμα 22 φαίνεται η γραφική παράσταση του x_{n+1} συναρτήσει του x_n. Από το σχήμα αυτό φαίνεται ότι η μορφή του ελκυστή δεν είναι πλέον τόσο καθαρή, λόγω του θορύβου που προσθέσαμε.



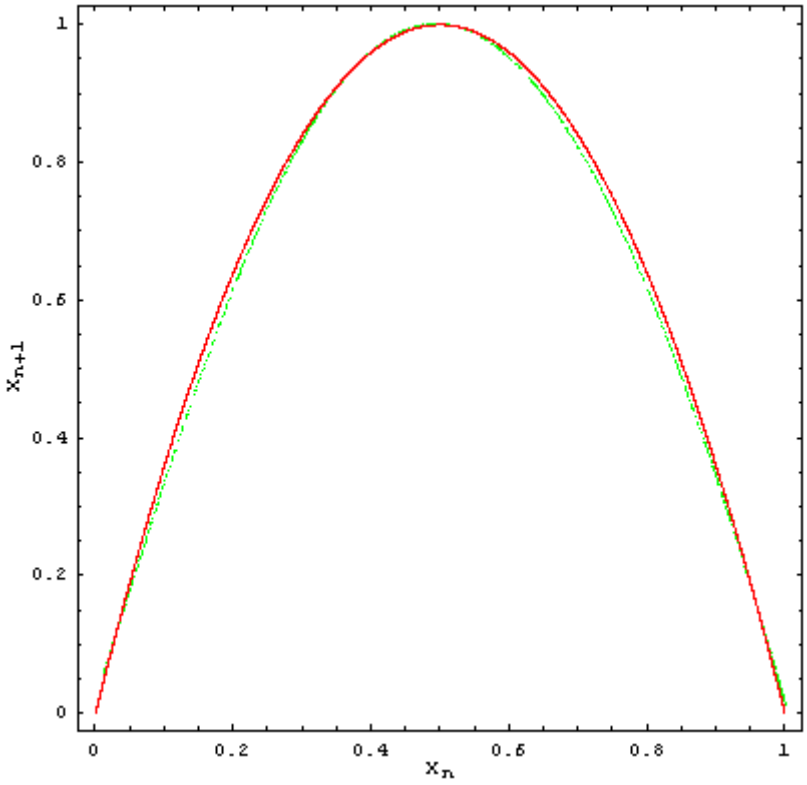
Σχήμα 22: Ο ελκυστής της λογιστικής απεικόνισης με την προσθήκη θορύβου

Επαναλαμβάνουμε το πείραμα της πρόβλεψης, όπως και παραπάνω. Η πρόβλεψη για την περίπτωση αυτή φαίνεται στο Σχήμα 23



Σχήμα 23: Σύγκριση της πρόβλεψης με τη πραγματική τροχιά, για την περίπτωση της προσθήκης θορύβου. Η χρονοσειρά που προβλέπει το δίκτυο φαίνεται με πράσινο, ενώ η πραγματική με κόκκινο

Η ακρίβεια της πρόβλεψης για την περίπτωση με θόρυβο, κινείται σχεδόν στα επίπεδα αυτής χωρίς θόρυβο. Βραχυπρόθεσμα έχουμε μία ταύτιση και στη συνέχεια η πρόβλεψη αποκλίνει. Τέλος, στο Σχήμα 24 φαίνεται η σύγκριση του ανακατασκευασμένου ελκυστή με τον πραγματικό. Η τάυτιση αυτή τη φόρά είναι ελάχιστα χειρότερη από ότι στην περίπτωση χωρίς θόρυβο. Ωστόσο, το αποτέλεσμα είναι πολύ καλό. Είναι προφανές ότι το δίκτυο μπόρεσε να μάθει τα γενικά χαρακτηριστικά της χρονοσειράς και να τα απομονώσει από τον θόρυβο που προσθέσαμε.



Σχήμα 24: Σύγκριση του ανακατασκευασμένου ελκυστή (πράσινο) με τον πραγματικό (κόκκινο), στην περίπτωση που έχουμε προσθέσει θόρυβο

Βιβλιογραφία

Albano, A.M., A. Passamante, T. Hediger and Mary Eileen Farell. 1992. “Using neural nets to look for chaos.” Physica D **58**: 1-9

Αργυράκης, Πάνος (2001). *Τεχνητή Νοημοσύνη – Εφαρμογές*. Ελληνικό Ανοιχτό Πανεπιστήμιο.

Bigus, Joseph P and Bigus, Jennifer (). *Constructing Intelligent Agents Using Java.* Willey Computer Publishing

Haykin, Simon (1999). *Neural Networks, A Comprehensive Foundation*. Prentice Hall International, Inc, New Jersey

Ilachinski, Andrew (2001). *Cellular Automata, A Discrete Universe*. World Scientific Publishing Co. Pte. Ltd.

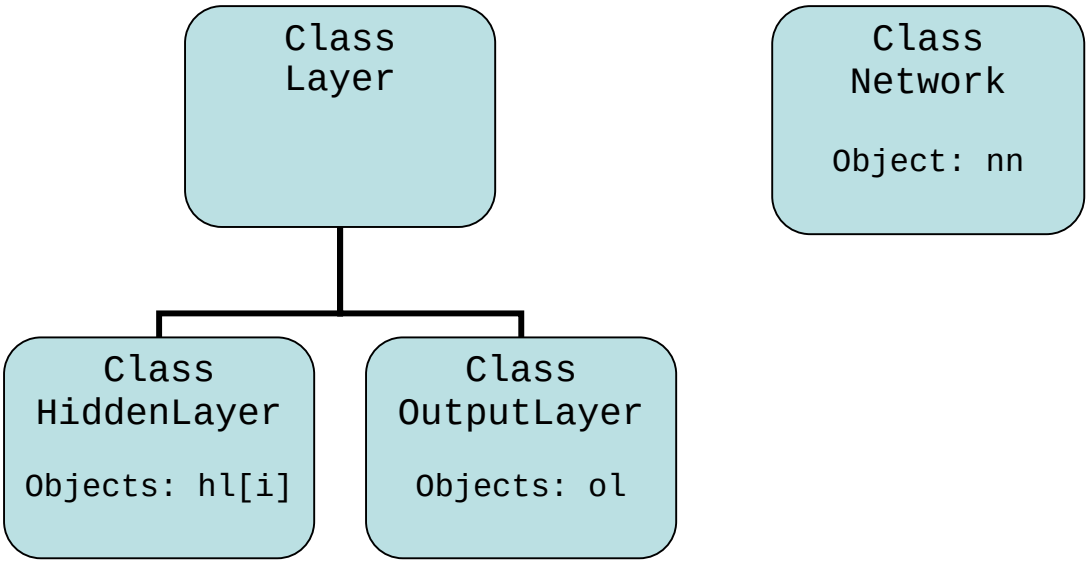
Masters, Timothy (1993). *Practical Neural Network Recipies in C++*. Academic Press

Παράρτημα (Κώδικας)

Πρόγραμμα MultiLayer FeedForward Network

Δομή των δεδομένων.

Στο πρόγραμμα υπάρχουν τρεις τάξεις: `NeuralNetwork`, `HiddenLayer` και `OutputLayer`. Η τάξη `Layer` χρησιμοποιείται ως βασική τάξη των δύο τελευταίων και δεν έχει η ίδια δικά της αντικείμενα.



Οι ορισμοί των παραπάνω καθώς και οι αντίστοιχες συναρτήσεις εγκατάστασης βρίσκονται στο αρχείο `classes.h`
Στο αρχείο `functions.h` βρίσκονται οι ορισμοί των συναρτήσεων των παραπάνω τάξεων. Οι σημαντικότερες από αυτές είναι:

Τάξη NeuralNetwork

`void CreateLayers()`
Δημιουργεί τα αντικείμενα των των τάξεων HiddenLayer και OutputLayer και αρχικοποιεί τις απαραίτητες μεταβλητές τους, όπως τον αριθμό των νευρώνων σε κάθε Layer, τα αντίστοιχα βάρη τους κλπ.

`void Execute(int i)`
Αναθέτει στο πρώτο επίπεδο το i-σετ τιμών εισόδου και διαδοχικά ενεργοποιεί κάθε επίπεδο, μέχρι να φτάσουμε στο Output Layer και να πάρουμε τις τιμές εξόδου.

`void BackPropagation(int i)`
Χρησιμοποιώντας τα αποτελέσματα της Execute υπολογίζει το σφάλμα (Επιθυμητή έξοδος – Πραγματική έξοδος) και το διαδίδει προς τα πίσω σύμφωνα με τον αλγόριθμο Backpropagation και διορθώνει τα βάρη.

`void Train(int i)`
Καλεί διαδοχικά την `Execute` και την `BackPropagation` για το i-σετ τιμών εισόδου.

`double CalcError(int k)`
Υπολογίζει και επιστρέφει το σφάλμα του δικτύου (rms) για να αποφασίσουμε αν είναι ικανοποιητικό.

`double Validate(int)`
Στην περίπτωση που έχουμε σετ εκμάθησης και σετ validation χρησιμοποιεί το validation σετ για τιμές εισόδου στο δίκτυο και υπολογίζει το σφάλμα για να δούμε εάν το δίκτυο εκπαιδεύτηκε σωστά (με την ακρίβεια που απαιτούμε). Επιστρέφει το συνολικό σφάλμα του δικτύου στις τιμές εξόδου. Δέχεται ως είσοδο πλήθος των σετ εισόδου-εξόδου του validation set.

void Anneal(double Tmin, double Tmax, int Tsteps, double tol)
Εφαρμόζει τον αλγόριθμο Simulated Annealing. Δέχεται ως είσοδο τις τιμές της Θερμοκρασιας (αρχική και τελική) και τον αριθμό των βημάτων για τη σταδιακή ψύξη και την επιθυμητή ακρίβεια.

void showTrainResults()
Καλείται μετα την εκπαίδευση για να μας δείξει στην οθόνη τις τιμές εισόδου και τις τιμές επιθυμητής και πραγματικής εξόδου του κάθε σετ τιμών.

Τάξη Layer

Αυτή η τάξη έχει συναρτήσεις που είναι κοινές και στα hidden και στο Output Layer.

void setValues(int, int, int, double**)
Καλείται μετά τη δημιουργία ενός αντκειμένου Layer και αρχικοποιεί τις απαραίτητες μεταβλητες `NumOfNeuronsInCurrentLayer`, `NumOfNeuronsInPreviousLayer`, `NumOfNeuronsInNextLayer` και w (βάρη) από τις αντίστοιχες εισόδους που δέχεται. Με τη βοήθεια των παραπάνω τιμών δημιουργει δυναμικά τους πίνακες `Inputs`, `Outputs` που θα κρατούν αντίστοιχα τις τιμές εισόδου από τους νευρώνες του προηγούμενου Layer και τις τιμές εξόδου από τους δικούς του νευρώνες, τον πίνακα `NetInput` που κρατά την είσοδο σε κάθε νευρώνα δηλ. το σταθμισμένο άθροισμα των εισόδων από το προηγούμενο Layer και μια σειρά από άλλους πίνακες και μεταβλητές που χρησιμοποιούνται στους υπολογοσμούς για την αλλαγή των βαρών.

void CalculateLayerNetInput(double*)

Συνάρτηση που υπολογίζει για κάθε νευρώνα, το άθροισμα των γινομένων των εισόδων από το προηγούμενο Layer επί τα αντίστοιχα βάρη.

double* Activate(void)
Χρησιμοποιεί την προηγούμενη συνάρτηση και με τη βοήθεια της ActivationFunction υπολογίζει και επιστρέφει την έξοδο του κάθε νευρώνα για να χρησιμοποιηθει ως είσοδος στο επόμενο Layer.

Τάξη HiddenLayer

Συναρτήσεις ειδικά για τα hidden layers hl[i].

void HiddenLayer::ActivationFunction(double*c, double*y)
Η μη γραμμική Activation function για τα hidden layers. Δέχεται τον πίνακα με τις τιμές εισόδου σε κάθε νευρώνα (NetInput) και επιστρέφει πίνακα με τις τιμές εξόδου, χρησιμοποιώντας τη λογιστική συνάρτηση. Είναι ίδια για κάθε νευρώνα όλων των hidden layers. Μόνο οι νευρώνες του Output Layer μπορουν να έχουν διαφορετική (πχ. Γραμμική) γι' αυτό και έχουμε διαφορετικη συνάρτηση `OutputLayer::ActivationFunction`.

double HiddenLayer::ActivationFunctionPrime(double c)
Υπολογίζει και επιστρέφει την παράγωγο της activation function που χρησιμοποιείται στον αλγόριθμο BackPropagation.

Οι συναρτήσεις `CalculateSumDelta` και `LocalGradient` χρησιμοποιούνται επίσης για υπολογισμό ενδιάμεσων τιμών απαραίτητων στην BackPropagation.

Τάξη OutputLayer

Ιδιες συναρτήσεις με την HiddenLayer προσαρμοσμένες για το Layer εξόδου.

Ροή του προγράμματος.

Στην συνάρτηση main() του προγράμματος ξεκινάμε θέτοντας τιμές για τον αριθμό των hidden layers, το πλήθος των νευρώνων σε κάθε layer, τον αριθμό των σετ εισόδου – εξόδου, τον αριμό εισόδων και εξόδων του κάθε σετ, το αρχείο που είναι αποθηκευμένα τα δεδομένα και δημιουργούμε με αυτά ένα αντικείμενο της τάξης NeuralNetwork.
Αυτό με τη σειρά του (στην createLayers()) αναλαμβάνει να δημιουργήσει τα αντικείμενα των τάξεων HiddenLayer και OutputLayer (όσα χρειάζονται), να τα αρχικοποιήσει με τα παραπάνω δεδομένα και να αναθέσει τυχαίες τιμές στα βάρη.

Στη συνέχεια εκπαιδεύουμε το δίκτυο καλώντας την train() για κάθε σετ τιμών, μέχρι να επιτευχθει η επιθυμητή ακρίβεια ή να εκτελεστεί το μέγιστο αριθμό επαναλήψεων που ορίσαμε.
Για την περίπτωση XOR σταματάμε εδώ.
Αν έχουμε και validation σετ, όπως στην περίπτωση της πρόβλεψης της χαοτικής τροχιάς, αμέσως μετα την εκπαίδευση καλούμε τη validate() για να δούμε αν το δίκτυο υπολογίζει ικανοποιητικά και τις τιμές για τις οποίες δεν έχει εκπαιδευτεί.
Τέλος, στην περίπτωση της e^{-x2} και του logistic map δίνουμε τιμές εισόδου και παίρνουμε αποτελέσματα που αποθηκεύουμε σε αρχεία για περαιτέρω επεξεργασία και δημιουργία γραφικών παραστάσεων κλπ.