# Ένας Αυξητικός Αλγόριθμος Εκπαίδευσης Νευρωνικών Δικτύων Ταξινόμησης

Η ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

υποβάλλεται στην
ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Νικόλαο Τσάπανο

ως μέρος των Υποχρεώσεων για τη λήψη του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ
ΠΛΗΡΟΦΟΡΙΚΗ
ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ
ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΚΑΙ ΤΙΣ ΕΦΑΡΜΟΓΕΣ

Νοέμβριος 2005

# Περιεχόμενα

# Κεφάλαιο 1

# Εισαγωγή

Το θέμα αυτής της εργασίας είναι η διατύπωση, μελέτη και αξιολόγηση ενός αλγορίθμου εκπαίδευσης νευρωνικών δικτύων που βασίζεται στις γεωμετρικές ιδιότητες των σημείων ενός $d$-διάστατου χώρου, όταν αυτά προβάλονται σε κάποια κυρτή επιφάνεια μιας παραπάνω διάστασης. Η επιφάνεια που χρησιμοποιούμε είναι η προβολή των σημείων στο παραβολοειδές μιας παραπάνω διάστασης[1], αλλά οποιαδήποτε άλλη κυρτή επιφάνεια έχει τις επιθυμητές ιδιότητες στις οποίες βασίζεται ο αλγόριθμός μας.

Παραδοσιακά η εκπαίδευση ενός νευρωνικού δικτύου γίνεται με τον ορισμό μιας συνάρτησης σφάλματος και την ελαχιστοποίησή της με κάποια μέθοδο βελτιστοποίησης. Ο αλγόριθμος που παρουσιάζουμε σε αυτήν την εργασία είναι, στο έπακρο της γνώσης μας, ο πρώτος αποδοτικός (τόσο ως προς το πλήθος, όσο και τη διάσταση των δεδομένων) αυξητικός αλγόριθμος εκπαίδευσης νευρωνικών δικτύων που εγγυάται, θεωρητικά τουλάχιστον, τέλεια εκπαίδευση και τρέχει σε πολυωνυμικό χρόνο. Μια άλλη διαφορά με την εκπαίδευση παραδοσιακών δικτύων, είναι ότι στον αλγόριθμό μας οι νευρώνες έχουν διαφορετική προτεραιότητα, αλλά στην ενότητα 3.11 στη σελίδα 35 θα δείξουμε πως μπορούμε από ένα δίκτυο με προτεραιότητα που εκπαίδευσε ο αλγόριθμός μας να κατασκευάσουμε ένα αντίστοιχο παραδοσιακό δίκτυο που είναι ακριβώς ισοδύναμο.

Η ιδέα της προβολής στο παραβολοειδές μιας παραπάνω διάστασης για την επίλυση προβλημάτων πρωτοεμφανίστηκε από τους Edelsbrunner και Seidel [1], όταν βρήκαν μια συσχέτιση ανάμεσα στη Delaunay διαίρεση σε τρίγωνα (η οποία είναι δυική του διαγράμματος Voronoi) ενός συνόλου σημείων και το κυρτό περίβλημα των ίδιων σημείων, όταν αυτά προβάλονται στο παραβολοειδές μιας παραπάνω διάστασης.

[1] Το σημείο $[x_1, x_2, \ldots, x_d]^T$ προβάλεται στο σημείο $[x_1, x_2, \ldots, x_d, x_{d+1} = \sum_{i=1}^{d} x_i^2]^T$

Όσον αφορά στις γεωμετρικές προσεγγίσεις για προβλήματα μηχανικής μάθησης, ο Omohundro [2] χρησιμοποιώντας kd-trees έδωσε έναν γρήγορο γεωμετρικό αλγόριθμο ($O(log(n))$) για την απάντηση σε ερωτήσεις πλησιέστερου γείτονα και μια παραλλαγή σύμφωνα με την οποία μπορούσε αυτός ο γρήγορος αλγόριθμος να εφαρμοστεί στην Delaunay διαίρεση σε τρίγωνα των σημείων. Βέβαια ο υπολογισμός της διαίρεσης σε τρίγωνα καθεαυτής (όπως αναφέραμε προηγουμένως [1]) ανάγεται σε υπολογισμό του κυρτού περιβλήματος, ο οποίος γίνεται βέλτιστα σε χρόνο $O(n^{\lfloor \frac{d}{2} \rfloor} + nlog(n))$ και άρα δεν είναι αποδοτική η κατασκευή της Delaunay διαίρεσης σε τρίγωνα, καθώς η διάσταση του χώρου $d$ εμφανίζεται στον εκθέτη του πλήθους σημείων $n$.

Σχετικά πρόσφατα οι **Ridella, Rovetta και Zunino** [3] πρότειναν την προσθήκη ενός επιπλέον χαρακτηριστικού σε κάθε δεδομένο, το οποίο έπαιρνε την τιμή του αθροίσματος των τετραγώνων όλων των άλλων χαρακτηριστικών του συγκεκριμένου δεδομένου (στην ουσία την προβολή του δεδομένου στο παραβολοειδές μιας παραπάνω διάστασης) και την εκπαίδευση του δικτύου με τη χρήση backpropagation. Παρουσιάζουν αποτελέσματα που δείχνουν την αυξημένη ικανότητα μάθησης και γενίκευσης αυτής της μεθόδου και επίσης αποδεικνύουν κάποιες καλές θεωρητικές ιδιότητες που έχει αυτή η προβολή με βάση κριτήρια που παρουσιάζονται στα [4] και [5].

Για την παρουσίαση του δικού μας αλγορίθμου αρχικά θα δώσουμε τα μαθηματικά θεμέλια, τα οποία αποτέλεσαν την έμπνευση και τη βάση του αλγόριθμού μας. Στη συνέχεια θα περιγράψουμε αναλυτικά τα βήματά του και θα δώσουμε μια εκτίμηση της πολυπλοκότητάς του. Μετά θα παρουσιάσουμε και θα σχολιάσουμε τις επιδόσεις του σε διάφορα datasets και τέλος θα επεκταθούμε πάνω σε διάφορους τρόπους με του οποίους θα μπορέσουμε ίσως να βελτιώσουμε τις επιδόσεις του αλγόριθμού μας.

# Κεφάλαιο 2

# Μαθηματικό υπόβαθρο

Σε αυτό το κεφάλαιο, αφού πρώτα δούμε σε γενικές γραμμές πώς λειτουργεί ένας βιολογικός νευρώνας, θα ασχοληθούμε με τη μαθηματική ερμηνεία της λειτουργίας του και την εφαρμογή που βρίσκουν τα τεχνητά νευρωνικά δίκτυα στην επίλυση προβλημάτων ταξινόμησης. Θα δούμε επίσης για ποιούς λόγους η προβολή των δεδομένων στο παραβολοειδές μιας παραπάνω διάστασης (καμπυλώνοντας ουσιαστικά το χώρο των δεδομένων) δίνει σε ένα τεχνητό νευρωνικό δίκτυο μεγαλύτερη ευελιξία διευκολύνοντας την ταξινόμηση των δεδομένων.

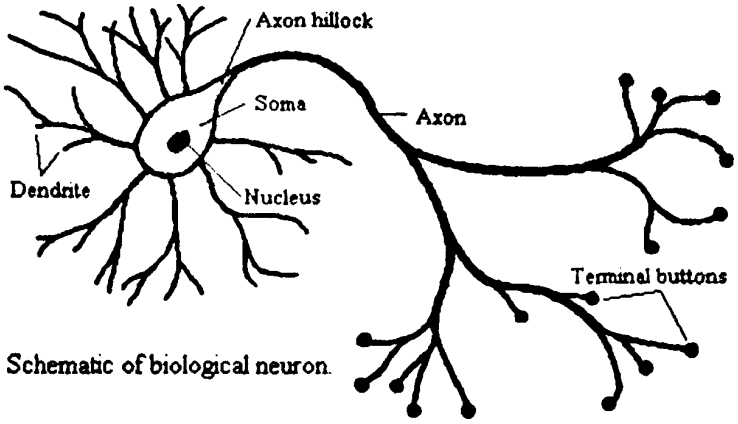
## 2.1 Ο βιολογικός νευρώνας

Ένας νευρώνας είναι μια βιολογική μονάδα αριθμητικών και λογικών πράξεων του εγκεφάλου (ένας ηλεκτρονικός υπολογιστής έχει μια αντίστοιχη ηλεκτρονική ALU - Arithmetic Logic Unit). Μπορεί να εκτελέσει όλες τις βασικές αριθμητικές πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση), να ελέγξει μια ανισότητα και ανάλογα με το αποτέλεσμα του ελέγχου να προχωρήσει ή όχι σε μια ενέργεια.

Ο νευρώνας είναι ένα κύτταρο με δενδρική μορφολογία (Σχήμα 2.1). Στον ανθρώπινο εγκέφαλο υπάρχουν περίπου 10 δισεκατομμύρια νευρώνες, ο καθένας από τους οποίους δέχεται σήματα από περίπου χίλιους γείτονές του και στέλνει σήματα σε περίπου χίλιους άλλους γείτονες. Εκτιμάται ότι αυτό που οι άνθρωποι αντιλαμβανόμαστε ως νοήμουσα σκέψη είναι το αποτέλεσμα της διάδοσης ηλεκτροχημικών σημάτων μεταξύ των διασυνδεδεμένων νευρώνων του εγκεφάλου μας.

Ο *πυρήνας* του κυττάρου είναι υπεύθυνος για τη συντήρηση και καλή λειτουργία του κυττάρου και δε λαμβάνει ενεργό μέρος στους υπολογισμούς.

Ένας νευρώνας δέχεται σήματα από τους γείτονές του μέσω των *δενδριτών*. Ως βασική λειτουργία του νευρώνα, όλα τα σήματα από κάθε γείτονα αθροίζο-




Σχήμα 2.1: Ένας βιολογικός νευρώνας [9].

νται ποσοτικά (πολλά μικρά σήματα μετατρέπονται σε ένα μεγάλο) και χρονικά (πολλοί βραχυπερίοδοι παλμοί μετατρέπονται σε ένα συνεχές σήμα). Στο άθροισμα αυτό προτίθεται και η *πόλωση* του νευρώνα που δεν εξαρτάται από τα εισερχόμενα σήματα.

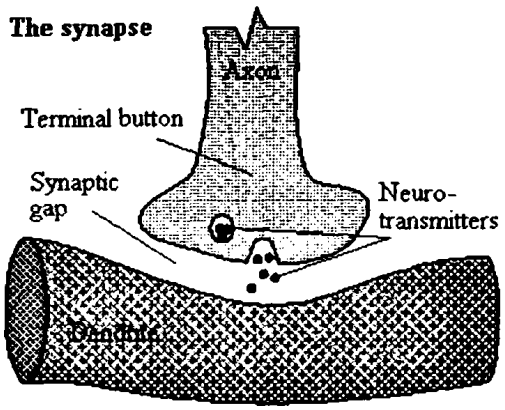
Στη συνέχεια το αποτέλεσμα της παραπάνω άθροισης συγκρίνεται με την τιμή του κατωφλίου του *λοφίσκου*. Αν το αποτέλεσμα ξεπερνά το κατώφλι, ο νευρώνας *πυροδοτείται* και στέλνει ένα σήμα στον *άξονα*.

Ο *άξονας* μεταδίδει ένα σταθερό σήμα, η ένταση του οποίου δεν εξαρτάται από το κατα πόσο μεγαλύτερο ήταν το άθροισμα των σημάτων που δέχτηκε ο νευρώνας από το κατώφλι του *λοφίσκου* του. Επίσης, το σήμα διατηρεί την έντασή του κατά μήκος του *άξονα*, χωρίς να διαφοροποιείται οσεσδήποτε διακλαδώσεις και άν υπάρχουν σε μια διαδρομή.

Στις απολήξεις του *άξονα* βρίσκονται οι *τερματικοί κόμβοι* του, οι οποίοι συνδέουν το νευρώνα με τους *δενρίτες* κάποιου γείτονά του. Αυτή η σύνδεση ονομάζεται *σύναψη*. Οι *τερματικοί κόμβοι* δεν ακουμπάν τους *δενρίτες*, αλλά υπάρχει ένα μικρό κενό ανάμεσά τους.

Η νευροχημική σύσταση ενός *τερματικού κόμβου* καθορίζει πόσο ενισχύεται ή αποδυναμώνεται το σήμα (πολλαπλασιασμός και διαίρεση), καθώς και αν το σήμα είναι διεγερτικό ή ανασταλτικό (αν το σήμα έχει θετικό ή αρνητικό πρόσημο). Ο αριθμός με τον οποίο πολλαπλασιάζεται στο μεταδιδόμενο σήμα ονομάζεται *βάρος* της *σύναψης*.

Η ευελιξία και η μαθησιακή ισχύς του εγκεφάλου οφείλονται στο γεγονός ότι τα βάρη των συνάψεων μπορούν να επηρρεάζονται από εξωτερικά ερεθίσματα




Σχήμα 2.2: Σύναψη μεταξύ κόμβου και δενδρίτη [9].

που δέχεται ο εγκέφαλος από το περιβάλλον. Ο εγκέφαλος μαθαίνει αλλάζοντας τις τιμές των βαρών των συνάψεων.

## 2.2 Το μαθηματικό μοντέλο

Όπως είδαμε, αυτό που κάνει ουσιαστικά ένας βιολογικός νευρώνας είναι να δέχεται τιμές στις εισόδους του, να πολλαπλασιάζει κάθε τιμή με το κατάλληλο βάρος και μετά να αθροίζει όλες τα βεβαρημένες τιμές και επιπλέον να προσθέτει και την πόλωση του ίδιου του νευρώνα. Αν υποθέσουμε ότι ο νευρώνας έχει $d$ εισόδους, ο υπολογισμός που εκτελείται είναι ο:

$$\sum_{i=1}^{d} x_i w_i + w_0$$

όπου $x_i$ είναι η είσοδος από τον $i$-οστό γείτονα, $w_i$ το βάρος της αντίστοιχης σύναψης και $w_0$ η πόλωση του νευρώνα.

Αν θεωρείσουμε το $\vec{x} = [x_1, x_2 \ldots, x_d]^T$ ως το διάνυσμα της εισόδου και το $\vec{w} = [w_1, w_2, \ldots, w_d]^T$ ως το διάνυσμα των βαρών ο υπολογισμός:

$$\sum_{i=1}^{d} x_i w_i = [x_1 \quad x_2 \quad \ldots \quad x_d] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = \vec{x} * \vec{w}$$

είναι το εσωτερικό γινόμενο των δυο διανυσμάτων, ή αλλιώς την προβολή του διανύσματος $\vec{x}$ στο $\vec{w}$.

Αν τώρα λάβουμε υπόψη μας την πόλωση του νευρώνα $w_0$, μπορούμε να δούμε ότι ο τύπος:

$$\frac{\vec{x} * \vec{w} + w_0}{|\vec{w}|}$$

μας δίνει την προσημασμένη απόσταση του σημείου $[x_1, x_2, \ldots, x_d]^T$ από το $d$-διάστατο υπερεπίπεδο με κανονικό διάνυσμα[1] το $\vec{w}$ που απέχει $w_0/|\vec{w}|$ από την αρχή των αξόνων και έχει γενική εξίσωση $w_1x_1 + w_2x_2 + \ldots + w_dx_d + w_0 = 0$. Αν το σημείο ανήκει στο υπερεπίπεδο, η απόσταση είναι 0. Αν βρίσκεται από την ίδια πλευρά του υπερεπιπέδου με το διάνυσμα $\infty * \vec{w}$, τότε η απόσταση είναι θετική, ενώ αν βρίσκεται από την αντίθετη πλευρά αρνητική.

Μπορεί εύκολα κάποιος να δει ότι ένα υπερεπίπεδο έχει άπειρες γενικές εξισώσεις της μορφής $c(w_1x_1 + w_2x_2 + \ldots + w_dx_d + w_0) = 0$, όπου $c \neq 0$ είναι ένας παράγοντας που πολλαπλασιάζει μια οποιαδήποτε γενική εξίσωση του υπερεπιπέδου. Αν μάλιστα τα βάρη κανονικοποιηθούν έτσι ώστε $|\vec{w}| = 1$, τότε η απόσταση του σημείου $\vec{x}$ από το υπερεπίπεδο είναι ακριβώς $\vec{x} * \vec{w} + w_0$.

Επανερχόμενοι στο βιολογικό νευρώνα, παρατηρούμε ότι ένας νευρώνας πυροδοτείται, όταν η είσοδός του απέχει από το υπερεπίπεδο με γενική εξίσωση που καθορίζεται από τα βάρη και την πόλωση του νευρώνα μεγαλύτερη απόσταση από μια συγκεκριμένη τιμή. Συνεπώς η λειτουργία του νευρώνα είναι να αποφασίζει από ποιά πλευρά ενός $d$-διάστατου υπερεπιπέδου βρίκεται η είσοδός του.

## 2.3 Τεχνητά νευρωνικά δίκτυα

Με σκοπό να γίνει καλύτερα κατανοητή η χρήση των νευρωνικών δικτύων σε προβλήματα ταξινόμησης και οι διαφορές του αλγορίθμου μας από τους παραδοσιακούς θα καλύψουμε επιγραμματικά τη δομή των νευρωνικων δικτύων καθώς επίσης και μερικές μεθόδους εκπαίδευσης.

### Το πρόβλημα της ταξινόμησης

Έστω $S$ ένα σύνολο δεδομένων καθένα από τα οποία αποτελείται από $d$ χαρακτηριστικά $x_1, x_2, \ldots, x_d$ και ανείκει σε μια κατηγορία $c_k$ από τις $C$ δυνατές κατηγορίες. Παρατηρούμε ότι τα δεδομένα μπορεί να θεωρηθούν σημεία στον $d$-διάστατο χώρο. Το ζητούμενο στο πρόβλημα της ταξινόμησης είναι, με βάση το σύνολο $S$, να βρεθεί μια απεικόνιση από ολόκληρο το χώρο των δεδομένων στο χώρο των κατηγοριών τέτοια, ώστε όσο το δυνατόν περισσότερα σημεία να απεικονίζονται στη σωστή κατηγορία.

[1] Διάνυσμα κάθετο στο υπερεπίπεδο

Όπως είναι φανερό από το γεγονός ότι το σύνολο $S$ είναι πεπερασμένο ενώ τα σημεία ολόκληρου του χώρου των δεδομένων άπειρα, δεν είναι ξεκάθαρο ποια κατηγορία είναι η 'σωστή' για κάθε σημείο. Προφανώς για τα σημεία που ανείχουν στο $S$ η απάντηση είναι γνωστή και απαιτούμε σε αυτά τα σημεία οι απαντήσεις να είναι σωστές, αλλά για όλα τα υπόλοιπα η απεικόνιση θα πρέπει να κάνει μια πεπαιδευμένη εικασία (να μαντέψει τη σωστή κατηγορία). Η ικανότητα μιας απεικόνισης να μαθαίνει τα δεδομένα ονομάζεται μαθησιακή ικανότητα, ενώ η ικανότητα να μαντεύει σωστά την κατηγορία άγνωστων σημείων αναφέρεται ως γενικευτική ικανότητα.

## Το Perceptron

Στην ενότητα 2.1 παρουσιάσαμε πως λειτρουργεί ένας βιολογικός νευρώνας. Το Perceptron είναι ακριβώς ένας αντίστοιχος τεχνητός νευρώνας που εκτελεί την ίδια λειτουργία. Τα βάρη $w_1, w_2, \ldots, w_d$ ενός Perceptron πολλαπλασιάζονται με τις εισόδους $x_1, x_2, \ldots, x_d$ και αθροίζονται κατά παράγοντα. Στο άθροισμα προστίθεται και η πόλωση $w_0$ του Perceptron και το τελικό αποτέλεσμα περνάει μέσα από μια συνάρτηση ενεργοποίησης $g(\alpha)$ που καθορίζει αν θα πυροδοτειθεί το Perceptron.

Για το απλό Perceptron η πιο συνηθισμένη συνάρτηση ενεργοποίησης είναι η βηματική

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

Μια επίσης δημοφιλής συνάρτηση ενεργοποίησης που έχει παρόμοια συμπεριφορά με τη βηματική είναι η λογιστική σιγμοειδής

$$g(\alpha) = \frac{1}{1 + e^{-\alpha}}$$

Η οποία σε σχέση με τη βηματική έχει το πλεονέκτημα ότι είναι παραγωγίσιμη και μαλιστα ισχύει ότι

$$g'(\alpha) = g(\alpha)(1 - g(\alpha))$$

Ένας τρόπος με τον οποίο μπορούμε να αυξήσουμε την ευελιξία του μοντέλου είναι να αντικαταστίσουμε τις $d$ γραμμικές εισόδους ενός Perceptron με $M$ μη γραμμικούς συνδυασμούς των $d$ αρχικών εισόδων $\phi_j(x)$. Η μορφή που θα έχει κάθε γραμμικός συνδυασμός $\phi_j(x)$ είναι προεπιλεγμένη, δεν επηρεάζεται από τα δεδομένα και δε μεταβάλεται κατά την εκπαίδευση. Επιπλέον ορίζουμε ως $y(x^{(i)}; w)$ την έξοδο του Perceptron με βάρη $w = [w_0, w_1, \ldots, w_d]^T$ για το δεδομένο $x^{(i)}$.

Όπως αναφέραμε στην παραπάνω ενότητα ο διαχωρισμός που αντιπροσωπεύει ένας νευρώνας είναι ένα $d$-διάστατο υπερεπίπεδο (το υπερεπίπεδο αυτό αναφέρεται και ως επιφάνεια απόφασης) και άρα είναι γραμμικός στο χώρο των δεδομένων. Εύκολα διαπιστώνει κάποιος ότι ένα Perceptron μπορεί να διαχωρίσει δυο κατηγορίες. Μπορούμε να διαχωρίσουμε περισσότερες κατηγορίες χρησιμοποιώντας ισάριθμα Perceptrons το καθένα από τα οποία αναγνωρίζει μόνο μια κατηγορία απορρίπτοντας όλες τις άλλες και ορίζοντας ότι η τελική απάντηση θα προκύπτει από το Perceptron που αναγνωρίζει πιο ισχυρά το δεδομένο (έχει τη μεγαλύτερη έξοδο).

Με βάση όλα τα παραπάνω ορίζουμε

$$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(x) + w_{k0}$$

την έξοδο του Perceptron υπεύθυνου για την αναγνώριση της κατηγορίας $k$, αν το δεδομένο είναι το $x$ και υπάρχουν $M$ μη γραμμικοί συνδυασμοί $\phi_j(x)$ στις εισόδους του Perceptron με βάρη $w_{kj}$.

Η αδυναμία του Perceptron φαίνεται από το γεγονός ότι, αν τα δεδομένα δεν είναι γραμμικά διαχωρίσιμα, ένα Perceptron δεν μπορεί να μάθει αυτά τα δεδομένα. Κλασικό παράδειγμα μη γραμμικά διαχωρίσιμων σημείων είναι το πρόβλημα της ισοτιμίας (parity), που στις δυο διαστάσεις ταυτίζεται με το πρόβλημα του αποκλειστικού ή (XOR).

Για την εκπαίδευση ενός Perceptron είναι χρήσιμο να ορίσουμε την τετραγωνική συνάρτηση σφάλματος εκπαίδευσης

$$E(w) = \frac{1}{2}\sum_{i=1}^{|S|}\sum_{k=1}^{C}(y_k(x^i; w) - t_k^i)^2$$

Όπου $t_k^i$ είναι η επιθυμητή έξοδος για το δεδομένο $x^i$ (1 αν το $x^i \in c_k$, 0 διαφορετικά). Θα παρουσιάσουμε δυο μεθόδους που εκπαιδεύουν ένα Perceptron προσπαθώντας να ελαχιστοποιήσουν το σφάλμα εκπαίδευσης.

### Gradient descent

Η πιο απλή μέθοδος ελαχιστοποίησης είναι να βρούμε προς τα που 'δείχνει' η παράγωγος και να κινηθούμε προς την αντίθετη κατεύθυνση, ξεκινώντας από ένα τυχαίο σημείο. Κάτι τέτοιο φυσικά απαιτεί η συνάρτηση σφάλματος να είναι παραγωγίσιμη. Για να το πετύχουμε αυτό αρκεί να χρησιμοποίσουμε μια παραγωγίσιμη συνάρτηση ενεργοποίησης, όπως για παράδειγμα η γραμμική και η σιγμοειδής.

Θεωρούμε ότι το συνολικό σφάλμα προκύπτει από το άθροισμα του επιμέρους σφάλματος για κάθε δεδομένο

$$E(w) = \sum_{i=1}^{|S|} E^i(w)$$

Άρα για να μάθει το Perceptron το δεδομένο θα πρέπει το διάνυσμα των βαρών του να μετακινηθεί από την προηγούμενη θέση του $t$ στην καινούρια θέση $t+1$ ως εξής

$$w^{(t+1)} = w^{(t)} - \nabla_w E(w)$$

επομένως η αλλαγή για κάθε βάρος θα είναι

$$w_{kj}^{(t+1)} = w_{kj}^{(t)} - \eta \frac{\partial E^i(w)}{\partial w_{kj}}$$

όπου $\eta$ είναι η σταθερά μάθησης του Perceptron. Αν αυτή η σταθερά είναι πολύ μικρή, τότε το Perceptron δε μαθαίνει αρκετά γρήγορα τα δεδομένα. Αν πάλι είναι πολύ μεγάλη μπορεί να οδηγήσει σε ταλαντώσεις γύρω από ένα τοπικό ελάχιστο, χωρίς να σημειώνεται πρόοδος.

Η μερική παράγωγος για γραμμική συνάρτηση ενεργοποίησης είναι

$$\frac{\partial E^i(w)}{\partial w_{kj}} = (y_k(x^i) - t_k^i)\phi_j(x^i)$$

για λόγους ευκολίας ορίζουμε

$$\delta_k^i = y_k(x^i) - t_k^i$$

και η μεταβολή των βαρών προκύπτει να είναι η εξής

$$\Delta w_{kj} = -\eta \delta_k^i \phi_j(x^i)$$

Σε περίπτωση που χρησιμοποιούμε τη σιγμοειδή ως συνάρτηση ενεργοποίησης, η μερική παράγωγος έχει τη μορφή

$$\frac{\partial E^i(w)}{\partial w_{kj}} = (g(\alpha)(1 - g(\alpha)))^2 \delta_k^i \phi_j(x^i)$$

Αποδεικνύεται ότι ελαχιστοποιώντας σε κάθε βήμα τη συνάρτηση σφάλματος $E(w)^i(w)$ για κάθε δεδομένο ελαχιστοποιείται και η ολική συνάρτηση σφάλματος $E(w) = \sum_{i=1}^{|S|} E^i(w)$.

## Το πολυεπίπεδο Perceptron (MLP)

Το Perceptron που εξετάσαμε παραπάνω έχει μόνο ένα επίπεδο νευρώνων. Είδαμε πως αν τα δεδομένα δεν είναι γραμμικά διαχωρίσιμα, πράγμα που είναι αναμενόμενο στις πρακτικές εφαρμογές, τότε ένα Perceptron δεν μπορεί να τα μάθει. Είδαμε όμως ότι στον εγκέφαλο οι νευρώνες συνδέονται μεταξύ τους. Θα μπορούσαμε λοιπόν να κατασκευάσουμε ένα πολύ πιο ισχυρό δίκτυο περνώντας τη βεβαρημένη έξοδο ενός νευρώνα στην είσοδο ενός άλλου νευρώνα.

Η πιο απλή αρχιτεκτονική ενός τέτοιου δικτύου είναι να διατάξουμε τους νευρώνες σε τρια ή περισσότερα επίπεδα και να συνδέσουμε βεβαρημένα κάθε νευρώνα του προηγούμενου επιπέδου με το επόμενο επίπεδο. Αυτά τα δίκτυα ονομάζονται δίκτυα πρόσθιας τροφοδότησης (feed-forward networks).

Τυπικά το πρώτο επίπεδο ονομάζεται επίπεδο εισόδου και το τελευταίο επίπεδο εξόδου. Τα ενδιάμεσα επίπεδα οσαδήποτε κι αν είναι ονομάζονται κρυμμένα επίπεδα. Έχει αποδειχτεί ότι ένα δίκτυο με μόνο ένα κρυμμένο επίπεδο μπορεί να μάθει οποιοδήποτε σύνολο δεδομένων, αρκεί ο αριθμός νευρώνων του κρυμμένου επιπέδου να είναι αρκετά μεγάλος.

Σε αυτό το σημείο αξίζει να σημειώσουμε ότι ο αριθμός των κρυμμένων επιπέδων καθώς και ο αριθμός νευρώνων ανά επίπεδο θα πρέπει να οριστεί πριν την εκπαίδευση. Αυτό έρχεται σε αντιδιαστολή με τον αλγόριθμο μας, τον οποίο θα παρουσιάσουμε αργότερα και που όπως θα δούμε προσθέτει όσους νευρώνες χρειάζεται κάθε φορά κατά τη διάρκεια της εκπαίδευσης.

Η εκπαίδευση ενός πολυεπίπεδου Perceptron μπορεί να γίνει παρόμοια με την εκπαίδευση ενός απλού Perceptron, ορίζοντας τη συνάρτηση σφάλματος

$$E(w) = \sum_{i=1}^{|S|} E^i(w)$$

και προσπαθώντας να την ελαχιστοποιήσουμε, π.χ. χρησιμοποιώντας gradient descent. Η παρουσία πολλών επιπέδων όμως περιπλέκει τον υπολογισμό της παραγώγου ως προς κάθε βάρος.

### Error Backpropagation

Ο κανόνας του Error Backpropagation είναι ένας τρόπος να υπολογίζουμε σχετικά εύκολα την πάραγωγου της συνάρτησης σφάλματος ως προς κάθε βάρος.

Θεωρούμε ότι η έξοδος του νευρώνα $j$ είναι $g(a_j)$ με

$$a_j = \sum_i w_{ji} z_i$$

όπου $z_i$ είναι η είσοδος από τον $i$-οστό νευρώνα του προηγούμενου επιπέδου, $w_{ji}$ το αντίστοιχο βάρος της σύναψης και $g$ η συνάρτηση ενεργοποίησης, η οποία στο εξής θα υποθέτουμε ότι είναι η σιγμοειδής.

Χρησιμοποιώντας τον κανόνα της αλυσίδας μπορούμε να δούμε ότι για κάθε βάρος $w_{ji}$ ισχύει ότι

$$\frac{\partial E^{(n)}}{\partial w_{ji}} = \frac{\partial E^{(n)}}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

Εύκολα μπορούμε να παρατηρήσουμε ότι $\partial a_j / \partial w_{ji} = z_i$. Για λόγους ευκολίας ορίζουμε $\delta_j = \partial E^{(n)} / \partial a_j$, επομένως $\partial E^{(n)} / \partial w_{ji} = \delta_j z_i$.

Χρησιμοποιώντας πάλι τον κανόνα της αλυσίδας μπορούμε να δούμε ότι για τους νευρώνες του επιπέδου εξόδου

$$\delta_j = g'(a_j) \frac{\partial E^{(n)}}{\partial y_j} = y_j - t_j$$

όπου $y_j$ η έξοδος του νευρώνα εξόδου $j$, ενώ για τους χρυμμένους νευρώνες

$$\delta_j = \sum_k \frac{\partial E^{(n)}}{\partial a_k} \frac{\partial a_k}{\partial a_j} = g'(a_j) \sum_k w_{kj} \delta_k = z_j (1 - z_j) \sum_k w_{kj} \delta_k$$

όπου $a_k$ αφορά τους νευρώνες του επόμενου επιπέδου που συνδέονται με το νευρώνα $j$.

Ο υπολογισμός της παραγώγου της συνάρτησης σφάλματος ως προς κάθε βάρος γίνεται περνώντας τα δεδομένα κατά την ορθή φορα στην αρχή και κρατώντας την έξοδο κάθε νευρώνα $k$, η οποία είναι είσοδος σε κάποιο νευρώνα $j$ το επόμενου επιπέδου ($y_k = z_j$) και στη συνέχεια κινούμενοι προς την αντίθετη φορά υπολογίζουμε την παράγωγο με βάση τον κανόνα Back Propagation.

$$\frac{\partial E}{\partial w_{ji}} = \delta_j z_i$$

με τα κατάλληλα $\delta_j$ και $z_i$ για κάθε $w_{ji}$.

Αν στο σύνολο δεδομένων $S$ υπάρχουν $n$ δεδομένα διάστασης $d$, τότε η πολυπλοκότητα μιας εποχής Error Backpropagation (της μεταβολής των βαρών για κάθε δεδομένο) είναι $O(nd)$.

Έχοντας έναν υπολογιστικά αποδοτικό τρόπο να βρίσκουμε την παράγωγο της συνάρτησης σφάλματος ως προς κάθε βάρος μπορούμε να εφαρμόσουμε διάφορες μεθόδους βελτιστοποίησης, από απλή Gradient Descent, Conjugate Gradient ή και μεθόδους Newton ή quasi-Newton (DFP, BFGS). Για της ανάγκες της εργασίας μας χρησιμοποιήσαμε τη μέθοδο Levenberg-Marquardt (όπως είναι υλοποιημένη στη MATLAB 7) για την εκπαίδευση παραδοσιακών δικτύων MLP.

## Το μοντέλο CBP

Το Circular Backpropagation (CBP) μοντέλο μελετήθηκε από τους Ridella, Rovetta και Zunino. Στην εργασία τους ([3]) πρότειναν την προσθήκη μιας επιπλέον εισόδου σε κάθε νευρώνα που θα παίρνει κάθε φορά την τιμή του αθροίσματος των τετραγώνων των τιμών όλων των άλλων εισόδων (ουσιαστικά την προβολή στο παραβολοειδές).

Μια από τις ιδιότητες του μοντέλου, όπως διατυπώνονται στην προαναφερθήσα εργασία, είναι ότι ο διαχωρισμός που υλοποιεί κάθε νευρώνας προβάλλεται σε κύκλο (εξ ου και η ονομασία circular ή γενικότερα σε υπερσφαίρα, εκτός από την περίπτωση όπου το βάρος της επιπλέον εισόδου είναι μηδενικό, οπότε και ο διαχωρισμός του νευρώνα γίνεται γραμμικός. Άμεσο συμπέρασμα αποτελεί το γεγονός ότι το CBP μοντέλο είναι υπερσύνολο του MLP. Δίνονται επίσης οι αντιστοιχίες μεταξύ των βαρών ενός νευρώνα και των παραμέτρων της υπερσφαίρας (κέντρο και ακτίνα).

Ενδιαφέρον επίσης προκαλεί η ισοδυναμία που μπορεί να υπάρξει μεταξύ των δικτύων CBP και των Γκαουσιανών RBF δικτύων. Αυτό είναι λογικό, αν σκεφτεί κανείς πως όταν ο πίνακας συμμεταβλητότητας της Γκαουσιανής κατανομής είναι της μορφής $\sigma I$ όπου $I$ ο μοναδιαίος πίνακας η 'καμπάνα' της κατανομής έχει κυκλική βάση.

Τέλος, σημαντική ιδιότητα του μοντέλου είναι ότι μπορεί να χρησιμοποιηθεί ο κανόνας Backpropagation για τον υπολογισμό της παραγώγου της συνάρτησης σφάλματος ως προς κάθε βάρος, αφού απλά προστίθεται μια επιπλέον είσοδος που πολλαπλασιάζεται γραμμικά με το επιπλέον βάρος. Αυτό σημαίνει ότι η εκπαίδευση CBP δικτύων είναι υπολογιστικά αποδοτική.

Εμείς, από την πλευρά μας, προτείνουμε την προβολή των δεδομένων σε μια οποιαδήποτε κυρτή επιφάνεια μιας παραπάνω διάστασης, αλλά αναγνωρίζουμε τα πλεονεκτήματα που έχει το παραβολοειδές, ακριβώς επειδή ο διαχωρισμός που υλοποιεί προβάλλεται σε υπερσφαίρα στον αρχικό χώρο. Για τη σύγκριση των δυο μεθόδων χρησιμοποιήσαμε είτε εικόνες από την παραπάνω εργασία ή αποτελέσματα που έδωσαν CBP δίκτυα εκπαιδευμένα με τη μέθοδο Levenberg-Marquardt της MATLAB.

## 2.4 Κυρτές επιφάνειες

Ο αλγόριθμος που θα παρουσιάσουμε βασίζεται στην προβολή των $d$-διάστατων δεδομένων σε κάποια κυρτή επιφάνεια του $(d+1)$-διάστατου χώρου. Πριν περάσουμε στη μελέτη της διαχωριστικής ικανότητας ενός $(d+1)$-διάστατου υπερεπιπέδου στα δεδομένα ενός $d$-διάστατου χώρου, όταν αυτά προβάλλονται σε κυρτη επιφάνεια μιας παραπάνω διάστασης, θα μελετήσουμε κάποιες ιδιότητες των κυρτών επιφανειών που θα μας φανούν χρήσιμες στην πορεία.

Καταρχήν θα ορίσουμε τι είναι μια κυρτή επιφάνεια με βάση τα εφαπτόμενα υπερεπίπεδα πάνω στην επιφάνεια.

**Ορισμός 1** *Μια συνεχής επιφάνεια ονομάζεται κυρτή, αν και μόνο αν κάθε υπερεπίπεδο εφαπτόμενο στην επιφάνεια τέμνει την επιφάνεια μόνο στο σημείο στο οποίο εφάπτεται.*

Μια $(d+1)$-διάστατη επιφάνεια μπορεί να είναι καθολικά κυρτη, π.χ. η προβολή στο παραβολοειδες, αλλά για να μπορούμε να χρησιμοποιήσουμε μια επιφάνεια αρκεί αυτή να είναι τοπικά κυρτή σε μια περιοχή του αρχικού $d$-διάστατου χώρου που να περιέχει όλα τα δεδομένα, π.χ. η προβολή στο $(d+1)$-διάστατο υπερημισφαίριο με ακτίνα αρκετά μεγάλη, ώστε το $d$-διάστατο υπερημισφαίριο με την ίδια ακτίνα να περιέχει όλα τα δεδομένα. Διάφορες κυρτές επιφάνεις φαίνονται στο Σχήμα 2.3 στη σελίδα 16.

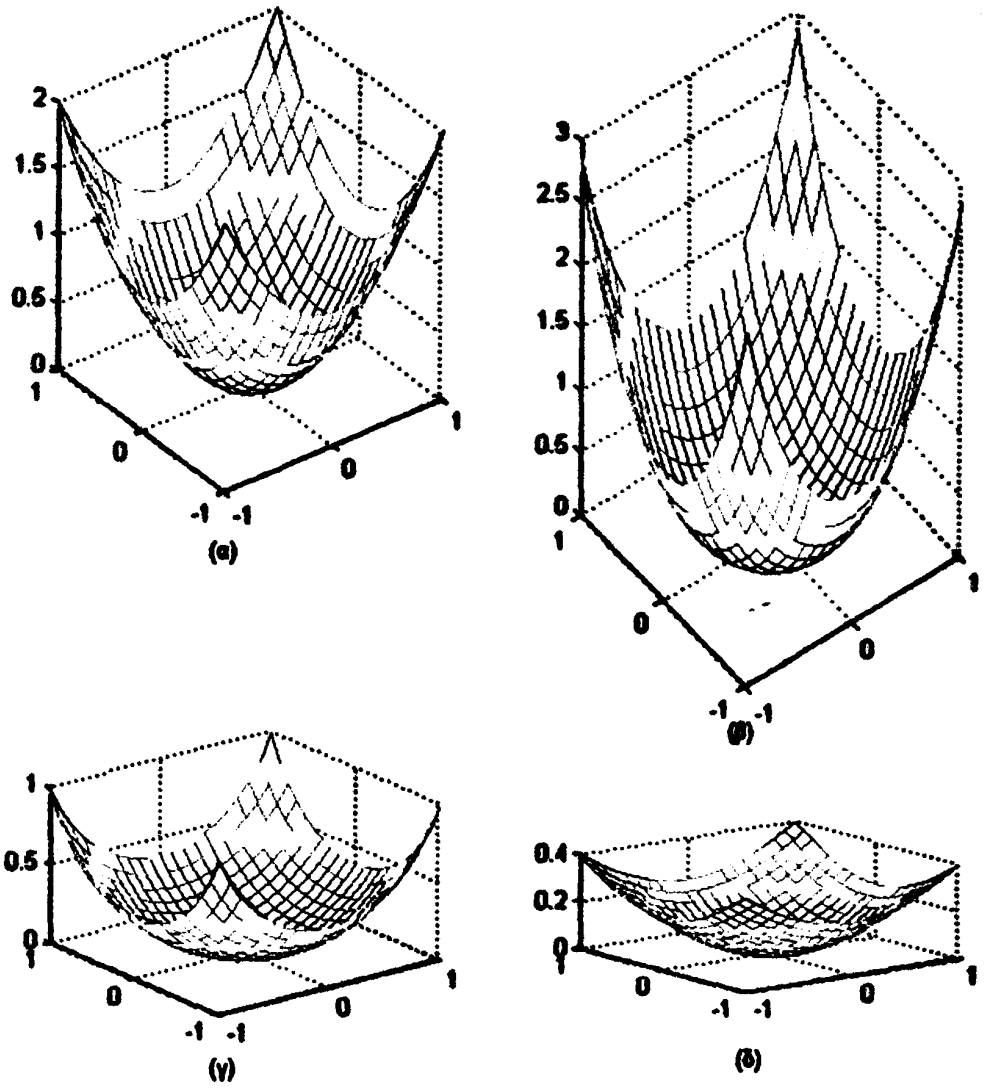
Τώρα θα αποδείξουμε ένα θεώρημα που θα μας εξασφαλίσει αργότερα τον μοναδικό προσδιορισμό ενός $(d+1)$-διάστατου υπερεπιπέδου από $(d+1)$ σημεία.

**Θεώρημα 1** *Όλα τα σημεία μιας κυρτής επιφάνειας είναι ανά τρία μη συνευθειακά.*

**Απόδειξη 1** *Έστω $\vec{p_1}$, $\vec{p_2}$ και $\vec{p_3}$ συνευθειακά σημεία μιας κυρτής επιφάνειας και, χωρίς περιορισμό της γενικότητας, ότι το $\vec{p_2}$ βρίσκεται ανάμεσα στο $\vec{p_1}$ και το $\vec{p_3}$. Έστω ότι το εφαπτόμενο υπερεπίπεδο στο $\vec{p_2}$ έχει γενική εξίσωση $\sum_{i=1}^{d+1} w_i x_i + w_0 = 0$ και κανονικό διάνυσμα το $\vec{w}$. Θεωρούμε το παραμετρικό ευθύγραμμο τμήμα $s(t) = \vec{p_1} + t(\vec{p_3} - \vec{p_1})$, $t \in [0,1]$. Παρατηρούμε ότι καθώς το $t$ κινείται στο $[0,1]$ ένα σημείο διατρέχει το ευθύγραμμο τμήμα $\vec{p_1}\vec{p_3}$. Η απόσταση αυτού του σημείου από το εφαπτόμενο υπερεπίπεδο στο $\vec{p_2}$ είναι*

$$D(s(t)) = \frac{(\vec{p_1} + t(\vec{p_3} - \vec{p_1})) * \vec{w} + w_0}{|\vec{w}|}$$

*Εύκολα παρατηρεί κάποιος ότι η παράγωγος της απόστασης ως προς $t$ είναι σταθερά. Διακρίνουμε τις εξής περιπτώσεις:*




Σχήμα 2.3: Κυρτές επιφάνειες: (α) το παραβολοειδές $z = x^2 + y^2$ (β) η κυβική απόσταση από την αρχή των αξόνων $z = \sqrt{(x^2+y^2)^3}$ (γ) το ημισφαίριο με ακτίνα $\sqrt{2}$ και κέντρο το σημείο $(0,0,2)$ $z = 2^2 - \sqrt{(x^2+y^2)}$ (δ) η Γκαουσιανή $z = \frac{1}{1-e^{\frac{(x^2+y^2)}{4}}}$

- *Αν $D(s(0)) = 0$ ή $D(s(1)) = 0$, τότε η παράγωγος είναι επίσης μηδενική και τουλάχιστον ένα εκ των $\vec{p_1}$ και $\vec{p_3}$ ανήκει στο εφαπτόμενο υπερεπίπεδο στο $\vec{p_2}$ και άρα το υπερεπίπεδο τέμνει την επιφάνεια σε παραπάνω από ένα σημείο.*

- *Έστω ότι $D(s(0)) > 0$, οπότε η παράγωγος είναι μη μηδενική και άρα η συνάρτηση είναι γνησίως μονότονη. Ξέρουμε ότι το υπερεπίπεδο εφάπτεται στο $\vec{p_2}$, άρα $\exists z \in (0,1)$ τέτοιο ώστε $D(s(z)) = 0$ και αφού η συνάρτηση είναι γνησίως μονότονη $D(s(1)) < 0$. Θεωρούμε μια άλλη παραμετρική συνεχή καμπύλη η οποία βρίσκεται πάνω στην επιφάνεια $c(r)$, $r \in [0,1]$ που ενώνει το $\vec{p_1}$ με το $\vec{p_3}$ χωρίς να περνάει από το $\vec{p_2}$ (αν δεν υπάρχει συνεχής καμπύλη, ή αν όλες οι συνεχείς καμπύλες περνάνε από το $\vec{p_2}$, τότε δεν είναι συνεχής η επιφάνεια). Έστω ότι $D(c(r))$ είναι η απόσταση από το εφαπτόμενο υπερεπίπεδο στο $\vec{p_2}$. Παρατηρούμε ότι $D(c(0)) = D(s(0)) > 0$ και $D(c(1)) = D(s(1)) < 0$, άρα $\exists w \in (0,1)$ τέτοιο ώστε $D(c(w)) = 0$ και το εφαπτόμενο υπερεπίπεδο τέμνει την επιφάνεια σε τουλάχιστον δυο σημεία.*

- *Αν $D(s(0)) < 0$ ομοίως.*

*Σε κάθε περίπτωση καταλήγουμε σε άτοπο εξ ορισμού της κυρτής συνεχούς επιφάνειας.*

**Πόρισμα 1** *$d+1$ σημεία μιας $(d+1)$-διάστατης κυρτής επιφάνειας ορίζουν μοναδικά το υπερεπίπεδο που περνάει από όλα αυτά τα σημεία.*

Αυτή η ιδιότητα μας εξασφαλίζει τη μοναδικότητα του υπερεπιπέδου, αφού δεν υπάρχουν τρία συνευθειακά σημεία γύρω από τα οποία θα μπορούσε να περιστραφεί ένα υπερεπίπεδο και να συνεχίσει να περνάει από όλα τα σημεία.

Για λόγους πληρότητας, θα παραθέσουμε και μια απόδειξη της κυρτότητας του παραβολοειδους.

**Θεώρημα 2** *Το $(d+1)$-διάστατο παραβολοειδές είναι κυρτή επιφάνεια.*

**Απόδειξη 2** *Το $(d+1)$-διάστατο παραβολοειδές είναι η επιφάνεια που περιγράφεται από την εξίσωση $x_{d+1} = x_1^2 + x_2^2 + \ldots + x_d^2$. Το εφαπτόμενο υπερεπίπεδο στο σημείο $z$ έχει τη γενική εξίσωση*

$$x_{d+1} = \nabla(x_1^2 + x_2^2 + \ldots + x_d^2) =$$

$$= 2z_1x_1 + 2z_2x_2 + \ldots + 2z_dx_d - z_{d+1} \iff$$

$$\iff 2z_1x_1 + 2z_2x_2 + \ldots + 2z_dx_d - z_{d+1} - x_{d+1} = 0$$

*Ας υποθέσουμε ότι αυτό το υπερεπίπεδο τέμνει το παραβολοειδες και σε ένα δεύτερο σημείο y, άρα*

$$2z_1y_1 + 2z_2y_2 + \ldots + 2z_dy_d - z_{d+1} - y_{d+1} = 0$$

*όμως* $z_{d+1} = z_1^2 + z_2^2 + \ldots + z_d^2$ *και* $y_{d+1} = y_1^2 + y_2^2 + \ldots + y_d^2$ *και η εξίσωση γίνεται*

$$2z_1y_1 + 2z_2y_2 + \ldots + 2z_dy_d \underbrace{-z_1^2 - z_2^2 - \ldots - z_d^2}_{-z_{d+1}} \underbrace{-y_1^2 - y_2^2 - \ldots - y_d^2}_{-y_{d+1}} = 0$$

*αλλάζοντας πρόσημα και τη σειρά των μεταβλητών*

$$(z_1^2 - 2z_1y_1 + y_1^2) + (z_2^2 - 2z_2y_2 + y_2^2) + \ldots + (z_d^2 - 2z_dy_d + y_d^2) = 0$$

*εφαρμόζοντας την ταυτότητα* $(\alpha - \beta)^2 = \alpha^2 - 2\alpha\beta + \beta^2$

$$(z_1 - y_1)^2 + (z_2 - y_2)^2 + \ldots + (z_d - y_d)^2 = 0$$

*ένα άθροισμα τετραγώνων είναι ίσο με το μηδέν αν και μόνο αν κάθε όρος είναι μηδέν*

$$\begin{array}{lll} (z_1 - y_1)^2 = 0 & \Longleftrightarrow & z_1 = y_1 \\ (z_2 - y_2)^2 = 0 & \Longleftrightarrow & z_2 = y_2 \\ & \vdots & \\ (z_d - y_d)^2 = 0 & \Longleftrightarrow & z_d = y_d \\ \hline & & \sum_{i=1}^{d} z_i^2 = \sum_{i=1}^{d} y_i^2 \iff z_{d+1} = y_{d+1} \end{array}$$

*και άρα το σημείο y ταυτίζεται με το z.*

## 2.5 Τομές επιπέδου - κυρτής επιφάνειας

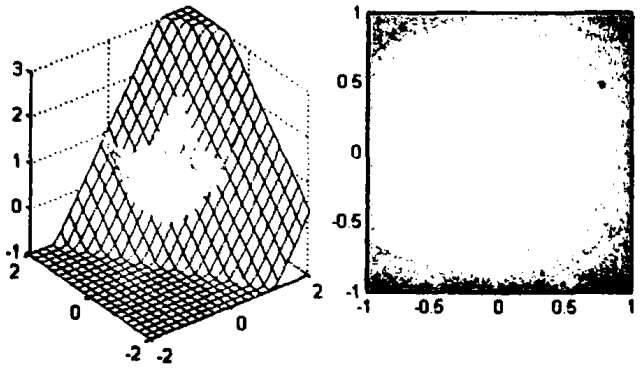
Τώρα θα δούμε τι μορφές μπορεί να πάρει η προβολή του διαχωρισμού που γίνεται από ένα υπερεπίπεδο σε μια $(d+1)$-διάστατη κυρτή επιφάνεια στον αρχικό $d$-διάστατο χώρο. Έτσι θα διαπιστώσουμε ότι έχουμε πραγματικά κατασκευάσει κάτι ισχυρότερο από το γραμμικό διαχωρισμό που γίνεται στα παραδοσιακά δίκτυα.

Όπως έχουμε πει θα μελετήσουμε την προβολή στο παραβολοειδές. Επίσης, θα περιορίσουμε τη μελέτη μας στις δυο διαστάσεις με τήν προβολή να γίνεται στην τρίτη, έτσι ώστε να μπορούμε να έχουμε μια εποπτεία του χώρου.

Από έναν αρχικό χώρο δυο διαστάσεων η προβολή στο παραβολοειδές της τρίτης διάστασης γίνεται ως εξής: $[x, y]^T \rightarrow [x, y, z = x^2 + y^2]^T$. Ένα επίπεδο στον τρισδιάστατο χώρο έχει γενική εξίσωση $\alpha x + \beta y + \gamma z + \delta = 0$ και κανονικό διάνυσμα το $[\alpha, \beta, \gamma]^T$

## Σημείο

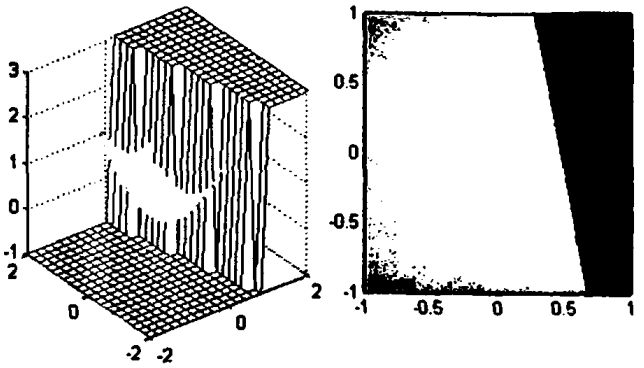
Από τον ορισμό της κυρτής επιφάνειας ξέρουμε ότι ένα εφαπτόμενο υπερεπίπεδο τέμνει την επιφάνεια ακριβώς σε ένα σημείο. Σε κάθε σημείο λοιπόν, το εφαπτόμενο επίπεδο μπορεί να διαχωρίσει το σημείο αυτό από όλα τα άλλα σημεία του αρχικού δισδιάστατου χώρου. Σε αυτό βασίζεται και ο ισχυρισμός ότι ο αλγόριθμός μας μαθαίνει τουλάχιστον ένα δεδομένο κάθε φορά που προσπαθεί να προσθέσει ένα νέο νευρώνα.




Σχήμα 2.4: Σημείο

## Ευθεία

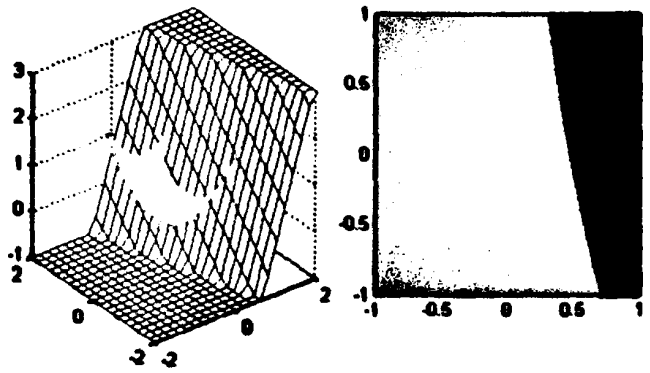
Εύκολα φαίνεται από τη γενική εξίσωση ενός επιπέδου στις τρεις διαστάσεις ότι αν θέσουμε $\gamma = 0$, η εξίσωση $\alpha x + \beta y + \delta = 0$ είναι μια ευθεία στον αρχικό χώρο. Συμπερασματικά οι λύσεις που υπάρχουν χρησιμοποιώντας κυρτή προβολή εμπεριέχουν όλες τις δυνατές λύσεις από παραδοσιακά δίκτυα.




Σχήμα 2.5: Ευθεία

## Ανοιχτή καμπύλη

Αν το $\gamma$ δεν είναι ακριβώς μηδέν, αλλά πολύ μικρό σε σχέση με τους υπόλοιπους συντελεστές η προβολή του διαχωρισμού για κάθε πρακτικό σκοπό παίρνει τη μορφή μιας ανοιχτής καμπύλης η οποία μπορεί να έχει οσοδήποτε μικρή και μεγάλη καμπυλότητα. Στην πραγματικότητα είναι ένας κύκλος με πολύ μεγάλη διάμετρο.




Σχήμα 2.6: Ανοιχτή καμπύλη

## Κύκλος

Ως υποπερίπτωση της κλειστής καμπύλης και χαρακτηριστικό αποκλειστικά του παραβολοειδούς είναι ο διαχωρισμός που γίνεται από το επίπεδο να προβάλεται ακριβώς σε κύκλο. Αυτό συμβαίνει όταν το εφαπτόμενο επίπεδο σε ένα σημείο ανυψωθεί κατά $r^2$ στον άξονα $z$.

Η ανύψωση δεν αλλάζει το κανονικό διάνυσμα του επιπέδου, απλά αυξάνει την απόσταση του κέντρου των αξόνων από το επίπεδο κατά $r^2$ και αυτό εκφράζεται στη γενική εξίσωση του εφαπτόμενου επιπέδου στο σημείο $[x_1, y_1]^T$ προσθέτοντας τον όρο $r^2$

$$z = 2x_1x + 2y_1y - x_1^2 - y_1^2 + r^2$$

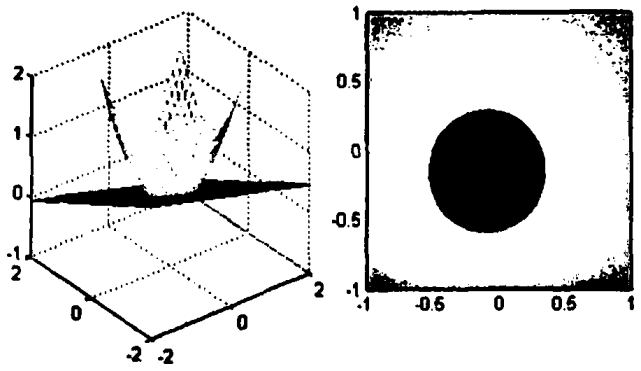
Η τομή του επιπέδου με το παραβολοειδές $z = x^2 + y^2$ γίνεται όταν

$$x^2 + y^2 = 2x_1x + 2y_1y - x_1^2 - y_1^2 + r^2 \iff$$

$$\iff x^2 + y^2 - 2x_1x - 2y_1y + x_1^2 + y_1^2 = r^2 \iff$$

$$(x - x_1)^2 + (y - y_1)^2 = r^2$$

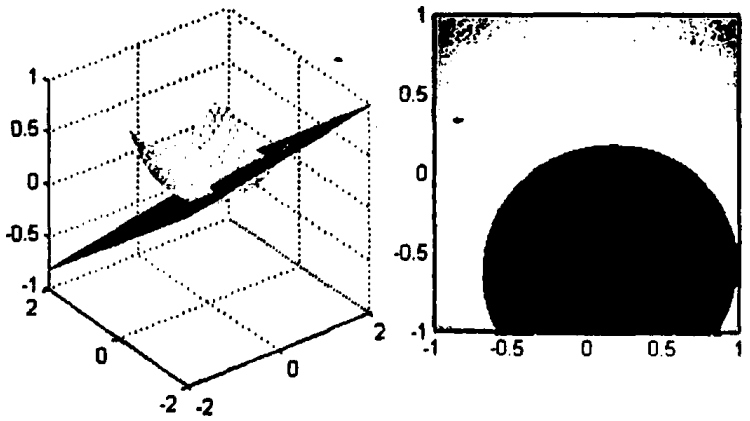
με αποτέλεσμα η τομή να προβάλεται σε κύκλο με κέντρο το σημείο $[x_1, y_1]^T$ και ακτίνα $r$.




Σχήμα 2.7: Κύκλος

## Κλειστή καμπύλη

Τέλος, χρησιμοποιώντας άλλες προβολές μπορούμε να πετύχουμε η προβολή του διαχωρισμού να γίνει μια κλειστή καμπύλη με σχήμα που εξαρτάται από την προβολή. Αυτό είναι ιδιαίτερα χρήσιμο, καθώς με αυτόν τον τρόπο μπορούμε να διαχωρίσουμε μια κλειστή περιοχή των δεδομένων από ολόκληρο τον υπόλοιπο χώρο.




Σχήμα 2.8: Κλειστή καμπύλη

# Κεφάλαιο 3

# Αλγόριθμος εκπαίδευσης

## 3.1 Γενική περιγραφή

Αρχίζουμε ταξινομώντας τα δεδομένα. Για καθεμία από τις $d+1$ συνολικά διαστάσεις έχουμε ένα πίνακα $S_i$ που περιέχει τους δείκτες των δεδομένων, ταξινομημένους σε αύξουσα διάταξη με βάση την τιμή της $i$-οστής συνιστώσας του κάθε δεδομένου. Το πρώτο στοιχείο του πίνακα $S_i$ δεν είναι η μικρότερη τιμή της $i$-οστής συνιστώσας, αλλά δείκτης στο δεδομένο που έχει τη μικρότερη $i$-οστή συνιστώσα.

Στη συνέχεια, και μέχρι το δίκτυο να μάθει όλα τα δεδομένα, προσθέτουμε νευρώνες στο δίκτυο. Ένας νευρώνας καθορίζεται από $d+1$ σημεία του χώρου που ονομάζουμε *σημεία ελέγχου* του νευρώνα και είναι υπεύθυνος για την εκμάθηση μιας κατηγορίας.

Θεωρούμε ότι ένας νευρώνας μαθαίνει ένα δεδομένο, όταν αφήνει το δεδομένο 'πίσω' του όπως γίνεται η διάσχιση της διάστασης. Αυτό σημαίνει ότι το κανονικό διάνυσμα $\vec{w} = [w_1, w_2, \ldots, w_d, w_{d+1}]^T$ του νευρώνα που διασχίζει τη διάσταση $i$ ορίζουμε έχει τη συνιστώσα $w_i$ στην ίδια φορά με τη φορά της διάσχισης και μαθαίνει ένα δεδομένο $\vec{x}$ ανν $\vec{w} * \vec{x} + w_0 < 0$. Στην πραγματικότητα, για την αποφυγή αριθμητικών προβλημάτων δε χρησιμοποιούμε το μηδέν για αυτόν τον έλεγχο, αλλά ένα πολύ μικρό αριθμό.

Στόχος μας είναι χρησιμοποιώντας ως σημεία ελέγχου δεδομένα από το συνόλο εκπαίδευσης να μάθει ένας νευρώνας όσο το δυνατόν περισσότερα δεδομένα της κατηγορίας για την οποία είναι υπεύθυνος, χωρίς να μαθαίνει δεδομένα άλλων κατηγοριών (σφάλμα).

Οι νευρώνες θεωρούμε ότι έχουν διαφορετικές προτεραιότητες. Πιο συγκεχριμένα, ο πρώτος νευρώνας με το id[1] 1 έχει και τη μεγαλύτερη προτεραιότητα

[1] Ο αύξων αριθμός του νευρώνα

και κάθε νευρώνας $v_k$ που προστίθεται έχει μικρότερη προτεραιότητα από τους παλιότερους $v_j, j < k$. Με την προσθήκη του κάθε νευρώνα καλύπτουμε όλα τα μέχρι τώρα ακάλυπτα δεδομένα που αυτός μαθαίνει. Στην ουσία προσπαθούμε να απομονώσουμε χρησιμοποιώντας ένα υπερεπίπεδο δεδομένα μιας κατηγορίας από το υπόλοιπο παραβολοειδές. Αφού λοιπόν 'κόψουμε' το κομμάτι με τα δεδομένα από το παραβολοειδές δε μας απασχολεί αν ένας επόμενος νευρώνας μιας άλλης κατηγορίας συμπεριλάβει αυτά τα δεδομένα, αφού τα έχει ήδη μάθει σωστά ένας νευρώνας με μεγαλύτερη προτεραιότητα.

Η προσθήκη των νευρώνων γίνεται ως εξής:

- Επιλέγουμε κάποια διάσταση.
- Επιλέγουμε τη φορά με την οποία θα διασχίσουμε τη διάσταση. Η διάσχιση γίνεται ακολουθώντας τον πίνακα $S_i$ με τους ταξινομημένους δείκτες. Κατα σύμβαση διατρέχουμε τον $d+1$-οστό πίνακα μόνο κατά την ορθή φορα, από το μικρότερο προς το μεγαλύτερο.
- Βρίσκουμε το πρώτο ακάλυπτο[2] σημείο κατά τη φορά με την οποία διασχίζουμε τη διάσταση.
- Με βάση αυτό το δεδομένο αρχικοποιούμε το νευρώνα χρησιμοποιώντας κάποια βοηθητικά σημεία για να συμπληρώσουμε τα $d+1$ σημεία ελέγχου που χρειαζόμαστε. Αυτό το δεδομένο καθορίζει επίσης την κατηγορία για την οποία είναι υπεύθυνος ο νευρώνας.
- Συνεχίζουμε τη διάσχιση της διάστασης ξεκινώντας από το παραπάνω σημείο. Επισκεπτόμαστε τα σημεία με τη σειρά που αυτά εμφανίζονται στον αντίστοιχο ταξινομημένο πίνακα. Εδώ διακρίνουμε δυο περιπτώσεις:
  - Αν το δεδομένο δεν είναι της κατηγορίας την οποία μαθαίνει ο νευρώνας, το αγνοούμε.
  - Αν το δεδομένο είναι της κατηγορίας για την οποία είναι υπεύθυνος ο νευρώνας, τότε θα δούμε αν μπορούμε να αντικαταστήσουμε ένα από τα υπάρχοντα σημεία ελέγχου του νευρώνα με το δεδομένο που επεξεργαζόμαστε και έτσι να καλύψουμε περισσότερα δεδομένα. Ο έλεγχος γίνεται εξαντλητικά, αντικαθιστώντας καθένα από τα σημεία ελέγχου με το τρέχων σημείο. Επιλέγουμε την αλλαγή που μας οδηγεί στην εκμάθηση περισσότερων δεδομένων προκαλώντας παράλληλα αποδεκτό σφάλμα (αν αυτή υπάρχει).
- Τελειώνουμε, όταν επισκεφτούμε και το τελευταίο δεδομένο στον ταξινομημένο πίνακα.

[2]Καλυμμένα είναι τα σημεία που έχει μάθει ήδη το δίκτυο.

Αφού τελειώσουμε με τη διάσχιση των δεδομένων και βρούμε το νευρώνα που μας δίνει την καλύτερη εκμάθηση, μετακινούμε κατάλληλα το νευρώνα προς τη φορά διάσχισης της διάστασης που επιλέξαμε. Ο νευρώνας έχει πλέον οριστικοποιηθεί και προστίθεται στο δίκτυο.

## 3.2 Επίλυση ομογενούς συστήματος

Βασικό στοιχείο του αλγόριθμού μας είναι ο καθορισμός των βαρών ενός νευρώνα από $d+1$ σημεία ελέγχου του παραβολοειδούς. Όπως θα δούμε, αυτό ανάγεται στην επίλυση ενός ομογενούς συστήματος $d+1$ εξισώσεων με $d+2$ αγνώστους. Ο αλγόριθμος επίλυσης βασίζεται την απαλοιφή Gauss.

Ένας νευρώνας στις $d+1$ διαστάσεις έχει $d+1$ βάρη, έστω $w_1$ έως $w_{d+1}$ και την πόλωσή του $w_0$. Αυτά τα βάρη αντιστοιχούν στο υπερεπίπεδο $h$ με γενική εξίσωση $w_1x_1 + w_2x_2 + \ldots + w_{d+1}x_{d+1} + w_0 = 0$ και κανονικό διάνυσμα το $w = [w_1, w_2, \ldots, w_{d+1}]^T$. Έστω ότι $p^{(i)}$ είναι το $i$-οστό σημείο ελέγχου με συντεταγμένες $[x_1^{(i)}, x_2^{(i)}, \ldots, x_{d+1}^{(i)}]^T$.

Θέλουμε να βρούμε το υπερεπίπεδο που περνάει από όλα τα σημεία ελέγχου. Άρα θέλουμε

$$p^{(i)} \in h \quad \forall i \iff p^{(i)T} * w + w_0 = 0 \quad \forall i$$

και επομένως από $d+1$ σημεία προκύπτει ένα ομογενές σύστημα από $d+1$ εξισώσεις

$$\begin{aligned} x_1^{(1)}w_1 + x_2^{(1)}w_2 + \ldots + x_{d+1}^{(1)} + w_0 &= 0 \\ x_1^{(2)}w_1 + x_2^{(2)}w_2 + \ldots + x_{d+1}^{(2)} + w_0 &= 0 \\ &\vdots \\ x_1^{(d+1)}w_1 + x_2^{(d+1)}w_2 + \ldots + x_{d+1}^{(d+1)} + w_0 &= 0 \end{aligned}$$

ή αν το δούμε με μορφή πινάκων

$$\underbrace{\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \ldots & x_{d+1}^{(1)} & 1 \\ x_1^{(2)} & x_2^{(2)} & \ldots & x_{d+1}^{(2)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{(d+1)} & x_2^{(d+1)} & \ldots & x_{d+1}^{(d+1)} & 1 \\ 0 & 0 & \ldots & 0 & 0 \end{bmatrix}}_{A} * \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{d+1} \\ w_0 \end{bmatrix}}_{W} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Σε αυτό το σημείο πρέπει να κάνουμε ορισμένες παρατηρήσεις. Η τελευταία γραμμή με τα μηδενικά του πίνακα $A$ είναι βοηθητική και εκφράζει το βαθμό ελευθερίας, αφού οι άγνωστοι είναι περισσότεροι από τις εξισώσεις. Τη συμπεριλάβαμε, ώστε ο πίνακας $A$ να είναι τετραγωνικός $(d+2) \times (d+2)$ και η

ορίζουσά του να είναι μηδέν, που σημαίνει ότι το ομογενές σύστημα έχει, εκτός από την τετριμμένη λύση, άπειρες λύσεις.

Όπως ήδη αναφέραμε στην ενότητα 2.2, ένα υπερεπίπεδο έχει άπειρες γενικές εξισώσεις, άρα οι άπειρες λύσεις δεν αποτελούν πρόβλημα, καθώς εκφράζουν το ίδιο υπερεπίπεδο. Επίσης, στην ενότητα 2.4 δείξαμε ότι στο παραβολοειδές δεν υπάρχουν 3 συνευθειακά σημεία και άρα καμιά εξίσωση δεν μπορεί να γραφεί ως γραμμικός συνδυασμός δυο άλλων. Αυτό είναι σημαντικό, γιατί ο βαθμός ελευθερίας είναι ακριβώς ένα και μια λύση μπορεί να βρεθεί αναθέτοντας αυθαίρετα μια τιμή στον άγνωστο που περισσεύει, χωρίς να χρειάζεται να προσδιορίσουμε και άλλον άγνωστο κατάλληλα.

Η επίλυση του συστήματος γίνεται με τη μέθοδο απαλοιφής Gauss. Ο στόχος της μεθόδου είναι μετά από κατάλληλες πράξεις μεταξύ των γραμμών του πίνακα να καταλλήξουμε σε έναν άνω τριγωνικό πίνακα με μονάδες στη διαγώνιο. Με την προσθήκη της τελευταίας γραμμής με τα μηδενικά που κάναμε το τελευταίο στοιχείο της διαγωνίου θα είναι μηδέν. Αυτό αντιστοιχή στην εξίσωση $0w_0 = 0$ και εκφράζει το γεγονός ότι μπορούμε να αναθέσουμε αυθαίρετα τιμή στο $w_0$.

Οι μεθοδολογία που ακολουθούμε για να τριγωνοποιήσουμε τον πίνακα είναι, ξεκινώντας από την πρώτη γραμμη, διαιρούμε όλα τα στοιχεία της γραμμής με το στοιχείο της γραμμής που ανήκει στη διαγώνιο (για την $i$-οστή γραμμή είναι το $i$-οστό στοιχείο). Έτσι εξασφαλίζουμε ότι ένα από τα στοιχεία θα είναι μονάδα. Ο πίνακας που προκύπτει από τον $A$ είναι ο εξής:

$$\begin{bmatrix} 1 & \frac{x_2^{(1)}}{x_1^{(1)}} & \dots & \frac{x_{d+1}^{(1)}}{x_1^{(1)}} & \frac{1}{x_1^{(1)}} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_{d+1}^{(2)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{(d+1)} & x_2^{(d+1)} & \dots & x_{d+1}^{(d+1)} & 1 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

Στη συνέχεια, με τη βοήθεια της μονάδας που μόλις δημιουργήσαμε, προσπαθούμε να κάνουμε τα υπόλοιπα στοιχεία της στήλης μηδέν. Αυτό μπορεί να γίνει πολλαπλασιάζοντας ολόκληρη τη γραμμή που επεξεργαζόμαστε με το πρώτο μη μηδενικό στοιχείο της γραμμής-στόχου στην οποία θέλουμε να μηδενίσουμε το στοιχείο στην κατάλληλη στήλη και αφαιρώντας ολόκληρη την πολλαπλασιασμένη γραμμή από τη γραμμή-στόχο. Γραμμές στόχοι είναι όλες οι γραμμές κάτω από αυτήν που επεξεργαζόμαστε εκτός από την τελευταια, η

οποία είναι βοηθητική.

$$\begin{bmatrix} 1 & \frac{x_2^{(1)}}{x_1^{(1)}} & \dots & \frac{x_{d+1}^{(1)}}{x_1^{(1)}} & \frac{1}{x_1^{(1)}} \\ 0 & x_2^{(2)} - x_1^{(2)}\frac{x_2^{(1)}}{x_1^{(1)}} & \dots & x_{d+1}^{(2)} - x_1^{(2)}\frac{x_{d+1}^{(1)}}{x_1^{(1)}} & 1 - x_1^{(2)}\frac{1}{x_1^{(1)}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & x_2^{(d+1)} - x_1^{(d+1)}\frac{x_2^{(1)}}{x_1^{(1)}} & \dots & x_{d+1}^{(d+1)} - x_1^{(d+1)}\frac{x_2^{(1)}}{x_1^{(1)}} & 1 - x_1^{(d+1)}\frac{x_2^{(1)}}{x_1^{(1)}} \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

Αφού εφαρμόσουμε την παραπάνω μεθοδολογία για όλες τις γραμμές εκτός από την τελευταία ο πίνακας θα έχει την εξής μορφή:

$$\begin{bmatrix} 1 & a_{12} & \dots & a_{1(d+1)} & a_{10} \\ 0 & 1 & \dots & a_{2(d+1)} & a_{20} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & a_{(d+1)0} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Παρατηρούμε ότι η τελευταία γραμμή του πίνακα $A$ αφού εφαρμόσουμε $d+1$ τη μεθοδολογία εκφράζει την εξίσωση

$$0x_1 + 0x_2 + \ldots + 0x_{d+1} + 0w_0 = 0$$

η οποία ισχύει $\forall w_1, w_2, \ldots, w_{d+1}, w_0$ και άρα μας δίνει τη δυνατότητα να αναθέσουμε όποια τιμή θέλουμε στο $w_0$. Επιλέξαμε να αναθέτουμε στο $w_0$ την τιμή 1. Η προτελευταία γραμμή εκφράζει την εξίσωση

$$w_{d+1} + a_{(d+1)0}w_0 = 0 \iff w_{d+1} = -a_{(d+1)0}w_0$$

και βρίσκουμε την τιμή του $w_{d+1}$ για $w_0 = 1$. Ξέροντας τις τιμές των $w_{d+1}$ και $w_0$ μπορούμε να βρούμε την τιμή του $w_d$ από την $d$-οστή γραμμή και διασχίζοντας τον πίνακα από κάτω προς τα πάνω χρησιμοποιούμε τις τιμές που έχουμε ήδη βρει για να υπολογίσουμε την καινούρια τιμή μέχρι να τις βρούμε όλες φτάνοντας στην κορυφή του πίνακα.

Πριν αναφέρουμε τη λύση που βρήκαμε κανονικοποιούμε όλες τις τιμές των βαρών έτσι ώστε $\sqrt{w_1^2 + w_2^2 + \ldots + w_{d+1}^2} = 1$. Αυτό γίνεται διαιρώντας κάθε βάρος (συμπεριλαμβανομένης και της πόλωσης $w_0$) με τη ρίζα του αθροίσματος των τετραγώνων που είδαμε παραπάνω χρησιμοποιώντας την αρχική λύση. Σκοπός αυτής της κανονικοποίησης είναι ο υπολογισμός που υλοποιεί ο νευρώνας να αντιστοιχεί ακριβώς στην απόσταση του δεδομένου από το υπερεπίπεδο που αποτελεί την επιφάνεια απόφασης του νευρώνα και επιπλέον το διάνυσμα $\vec{w} = [w_1, w_2, \ldots, w_{d+1}]^T$ να είναι το μοναδιαίο κανονικό διάνυσμα του υπερεπιπέδου του νευρώνα.

### Αριθμητικά θέματα

Ένα πρόβλημα που ενδεχομένως μπορεί να συναντήσουμε στην εφαρμογή της απαλοιφής Gauss είναι η παρουσία ή η εμφάνιση ενός μηδενικού στη διαγώνιο. Αν συμβεί αυτό, τότε προφανώς δεν μπορούμε να διαιρέσουμε όλα τα στοιχεία της γραμμής με το μηδέν. Αντιμετωπίζουμε αυτό το ενδεχόμενο αντικαθιστώντας κάθε προβληματικό μηδενικό με 0.0000000000000001 και συνεχίζουμε κανονικά. Αυτή η αλλαγή μπορεί να προκαλέσει σφάλματα τις τάξεως του $10^{-13}$, το οποίο θεωρείται αποδεκτό.

Η αυθαίρετη ανάθεση της τιμής 1 στο $w_0$ μπορεί επίσης σε συνδυασμό με το παραπάνω να οδηγήσει σε πολύ μεγάλες τιμές των βαρών, πράγμα που γενικά δε θεωρείται καλό στα νευρωνικά δίκτια, αλλά το πρόβλημα εξαλοίφεται με την κανονικοποίηση των βαρών στο τέλος.

Σε πρακτικές εφαρμογές υπάρχει επίσης η πιθανότητα λόγω αριθμητικών σφαλμάτων η λύση που μας δίνει η απαλοιφή Gauss να μην έχει την αναμενόμενη συμπεριφορά. Αυτό κυρίως παρατηρείται, όταν τα δεδομένα έχουν μηδενικά χαρακτηριστικά ή είναι συμμετρικά. Για την πρώτη περίπτωση μπορούμε να μετακινήσουμε λίγο κάθε χαρακτηριστικό ενός δεδομένου μακριά από την αρχή του άξονα του χαρακτηριστικού. Τυπικά αυτό σημαίνει ότι κάθε χαρακτηριστικό αλλάζει ως εξής:

$$x_i = \begin{cases} x_i + 0.00001 & \text{αν } x_i \geq 0 \\ x_i - 0.00001 & \text{αν } x_i < 0 \end{cases}$$

Αυτή η μετατόπιση προκαλεί μια μικρή ασυνέχεια στις καμπύλες όταν αυτές τέμνουν τους άξονες.

## 3.3 Αρχικοποίηση του νευρώνα

Όπως είπαμε νωρίτερα, θα μπορούσαμε να αρχικοποιούμε κάθε νευρώνα στο εφαπτόμενο υπερεπίπεδο στο πρώτο δεδομένο. Ο τρόπος όμως με τον οποίο υπολογίζουμε τα βάρη ενός νευρώνα είναι με βάση κάποια σημεία ελέγχου. Για να βρούμε κατάλληλα σημεία ελέγχου προσπαθούμε να προσεγγίσουμε το εφαπτόμενο υπερεπίπεδο με $d$ βοηθητικά σημεία των οποίων οι πρώτες $d$ συντεταγμένες ταυτίζονται με το δεδομένο εκτός από την $i$-οστή (για το $i$-οστό σημείο) στην οποία προστίθεται θόρυβος (η $(d+1)$-οστή συντεταγμένη προκύπτει από την προβολή). Τώρα πλέον έχουμε όλα τα σημεία ελέγχου που χρειαζόμαστε.

Ο θόρυβος που προσθέτουμε είναι μικρός και έχει φορά αντίθετη με αυτήν που διασχίζουμε τη διάσταση. Για παράδειγμα, αν διασχίζουμε μια διάσταση

από το $-\infty$ προς το $+\infty$ τα σημεία ελέγχου θα είναι:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_d \\ x_{d+1}^{(0)} \end{bmatrix}, \begin{bmatrix} x_1 - \epsilon \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_d \\ x_{d+1}^{(1)} \end{bmatrix}, \ldots \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i - \epsilon \\ \vdots \\ x_d \\ x_{d+1}^{(i)} \end{bmatrix}, \ldots \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_d - \epsilon \\ x_{d+1}^{(d)} \end{bmatrix}, \epsilon = 0.00000001$$

Ένα ακόμα θετικό στοιχείο αυτής της αρχικοποίησης είναι ότι όταν προσπαθήσουμε να μάθουμε το επόμενο δεδομένο ο νευρώνας μπορεί να περιστραφεί γύρω από όλους τους $d$ πρώτους άξονες, αλλά και από τη διαγώνιο.

## 3.4 Ενημέρωση του νευρώνα

Έχοντας ήδη κάποια σημεία ελέγχου να ορίζουν έναν νευρώνα θέλουμε να βρούμε πως μπορούμε να ενημερώσουμε αυτά τα σημεία έτσι ώστε να μάθει ο νευρώνας περισσότερα δεδομένα. Για την ενημέρωση χρησιμοποιούμε σημεία από το σύνολο εκπαίδευσης (training set).

Όπως είπαμε διασχίζουμε τα σημεία με τη σειρά (ορθή ή ανάποδη) που εμφανίζονται στον ταξινομημένο πίνακα της διάστασης που επιλέξαμε. Έστω ότι το σύνολο $\{c_1, c_2, \ldots, c_d, c_{d+1}\}$ είναι τα τρέχοντα σημεία ελέγχου και ότι το επόμενο σημείο στη διάσχιση είναι το $p$. Διακρίνουμε τις εξής περιπτώσεις:

- Το σημείο ανήκει στην κατηγορία που μαθαίνει ο νευρώνας. Θα πρέπει τώρα να ελέγξουμε αν μπορούμε να χρησιμοποιήσουμε το σημείο για να οδηγηθούμε σε εκμάθηση περισσότερων σημείων από το νευρώνα, χωρίς ταυτόχρονα να προκαλούνται απαράδεκτα σφάλματα εκπαίδευσης. Αυτό το πετυχαίνουμε αν αντικαταστήσουμε ένα από τα τρέχοντα σημεία ελέγχου με το καινούριο σημείο. Δυστυχώς δε σταθήκαμε ικανοί να βρούμε έναν πιο έξυπνο τρόπο επιλογής του σημείου ελέγχου προς αντικατάσταση, οπότε εξαντλητικά δοκιμάζουμε κάθεμια από τις $d$ δυνατές αντικαταστάσεις.

$$\{c_1, \ldots, c_i, \ldots, c_{d+1}\} \rightarrow \begin{cases} \{p, c_2, \ldots, c_d, c_{d+1}\} \\ \{c_1, p, \ldots, c_d, c_{d+1}\} \\ \quad \vdots \\ \{c_1, \ldots, p, \ldots, c_{d+1}\} \\ \quad \vdots \\ \{c_1, c_2, \ldots, p, c_{d+1}\} \\ \{c_1, c_2, \ldots, c_d, p\} \end{cases}$$

Αξιολογούμε κάθε αντικατάσταση με μια συνάρτηση που λαμβάνει υπόψη δυο ποσότητες. Το σφάλμα εκπαίδευσης και τα σημεία που μαθαίνει ο νευρώνας συνολικά. Επιλέγουμε την αντικατάσταση που οδηγεί σε αποδεκτό σφάλμα και οδηγεί σε εκμάθηση μεγαλύτερου ή ίσου αριθμό δεδομένων σε σχέση με το τρέχον σύνολο σημείων ελέγχου.

- Το σημείο δεν είναι της κατηγορίας που μαθαίνει ο νευρώνας. Σε αυτήν την περίπτωση απλά προσπερνάμε το σημείο και επεξεργαζόμαστε το επόμενο.

Πρέπει επίσης να παρατηρήσουμε πως, όταν επεξεργαζόμαστε ένα νέο σημείο της ίδιας κατηγορίας, δε μας ενδιαφέρει αν αυτό είναι ήδη καλυμμένο από προηγούμενο νευρώνα και ότι μεταξύ ισόπληθων λύσεων προτιμάμε αυτήν που συμπεριλαμβάνει το πιο πρόσφατο σημείο.

## 3.5 Υπολογισμός σφάλματος

Όπως είπαμε, τα κριτήρια με τα οποία βρίσκουμε αν και πως θα ενημερώσουμε τα σημεία ελέγχου τους τρέχοντος νευρώνα είναι μετρώντας το σφάλμα εκπαίδευσης που προκαλεί ο νευρώνας και τα σημεία της κατηγορίας που μαθαίνει.

Αν υποθέσουμε ότι έχουμε ήδη βρει με την απαλοιφή Gauss τα βάρη του νευρώνα χρησιμοποιώντας κάποιο σύνολο σημείων ελέγχου, τότε πολύ απλά μπορούμε να διατρέξουμε όλα τα δεδομένα και να δούμε πως τα ταξινομεί ο νευρώνας. Ένας τρόπος να γλυτώσουμε πράξεις είναι τα κινηθούμε πάνω στον ταξινομημένο πίνακα της διάστασης που έχουμε επιλέξει. Βρίσκουμε το πρώτο ακάλυπτο σημείο από την κάθε πλευρά του πίνακα και ελέγχουμε μόνο τα ενδιάμεσα σημεία.

Ο έλεγχος γίνεται σταδιακά. Έχουμε δυο μετρητές, έναν για το σφάλμα και έναν για τα δεδομένα που μαθαίνει ο νευρώνας.

- Ελέγχουμε αν το δεδομένο είναι της κατηγορίας που επεξεργαζόμαστε ή οποιασδήποτε άλλης κατηγορίας.
  - Αν η απάντηση είναι θετική, ελέγχουμε αν ο νευρώνας ταξινομεί το δεδομένο στην κατηγορία που μαθαίνει.
    * Ελέγχουμε αν το δεδομένο είναι ήδη καλυμμένο. Αν είναι, τότε δεν κάνουμε τίποτα. Αν δεν είναι, τότε ο νευρώνας μαθαίνει ένα νέο δεδομένο και ενημερώνουμε τον κατάλληλο μετρητή.
  - Αν η απάντηση είναι αρνητική, πάλι ελέγχουμε αν ο νευρώνας ταξινομεί το δεδομένο στην κατηγορία που μαθαίνει.

* Ελέγχουμε αν το δεδομένο είναι ήδη καλυμμένο. Αν είναι, τότε δεν κάνουμε τίποτα. Αν δεν είναι, τότε ο νευρώνας ταξινομεί λάθος ένα δεδομένο και ενημερώνουμε τον μετρητή σφάλματος. Αν αυτός ο μετρητής ξεπεράσει κάποιον αριθμό που έχουμε περάσει ως όρισμα tolerance (θα ασχοληθούμε με το χειρισμό του tolerance αργότερα), τότε θεωρούμε ότι η λύση που μας δίνει ο νευρώνας είναι απαράδεχτη και σταματάμε τη διάσχιση.

## 3.6 Κάλυψη δεδομένων

Η κάλυψη των δεδομένων γίνεται με εντελώς παρόμοιο τρόπο με τον υπολογισμό του σφάλματος. Οι μόνες διαφορές είναι ότι διατρέχουμε όλα τα δεδομένα και η σειρά των ελέγχων σε κάθε δεδομένο.

- Ελέγχουμε αν το δεδομένο είναι ήδη καλυμμένο. Αν είναι τότε το έχει μάθει κάποιος προηγούμενος νευρώνας.
  - Αν η απάντηση είναι αρνητική, ελέγχουμε αν ο νευρώνας ταξινομεί το δεδομένο στην κατηγορία που μαθαίνει. Σημειώνουμε ότι σε αυτό το σημείο δε μας ενδιαφέρει αν η ταξινόμηση γίνεται λάθος, γιατί ξέρουμε ότι το σφάλμα δεν ξεπερνάει το tolerance.
    * Καλύπτουμε το δεδομένο με το id του τρέχοντος νευρώνα.

## 3.7 Χειρισμός κακών περιπτώσεων

Θεωρητικά είναι δυνατόν κάθε νευρώνας να μαθαίνει χωρίς σφάλμα τουλάχιστον ένα δεδομένο. Στην πράξη όμως, λόγω εμφάνισης αριθμητικών λαθών ή ύπαρξης συμμετρικών δεδομένων και ισότιμων χαρακτηριστικών, υπάρχουν περιπτώσεις που αυτό δεν είναι δυνατόν. Αν και συνήθως το πρόβλημα λύνεται επιλέγοντας άλλη διάσταση, προκειμένου να εξασφαλίσουμε ότι ο αλγόριθμος θα τερματίζει πάντα για κάθε σύνολο δεδομένων ορίζουμε το tolerance (ανοχή σε σφάλματα) που μπορούμε να μεταβάλουμε κατάλληλα ώστε ο τερματισμός του αλγορίθμου να είναι εγγυημένος, έστω και με κάποιο σφάλμα εκπαίδευσης μικρότερου ή ίσου του tolerance.

Το tolerance αρχίκα είναι μηδέν, οπότε δεν ανεχόμαστε σφάλματα. Διατηρούμε έναν $(d+1) \times 2$ πίνακα blocked που και αυτός αρχικά περιέχει μηδενικά (εκτός από μια θέση, την τελευταία, η οποία είναι μονάδα) και αναφέρετε στις δυο κατευθύνσεις διάσχισης κάθε διάστασης που μπορούμε να επιλέξουμε. Όπως έχουμε αναφέρει, κατά σύμβαση δε διασχίζουμε ανάποδα την τελευταία διάσταση και για αυτό το λόγο η αντίστοιχη θέση του πίνακα είναι πάντα μονάδα.

Κάθε φορά που προσθέτουμε επιτυχώς ένα νευρώνα στο δίκτυο το tolerance μηδενίζεται και ο πίνακας blocked επαναφέρεται στην αρχική του κατάσταση που αναφέραμε στην προηγούμενη παράγραφο.

Σε περίπτωση που αποτύχουμε να προσθέσουμε ένα νευρώνα χωρίς να προκαλέσουμε σφάλμα, ή ο νευρώνας που βρήκαμε δε μαθαίνει κανένα δεδομένο, τότε σημειώνουμε την αποτυχία μας στον πίνακα blocked, στην κατάλληλη θέση ανάλογα με τη διάσταση και τη φορά που είχαμε επιλέξει. Την επόμενη φορά που θα επιλέξουμε διάσταση και φορά θα ξέρουμε με τη βοήθεια του πίνακα αν έχουμε ήδη αποτύχει με τη συγκεκριμένη επιλογή, την προσπερνάμε και ελέγχουμε την επόμενη επιλογή, καταλήγοντας σε μια επιλογή που δεν είναι 'μπλοκαρισμένη'.

Αν ποτέ ο πίνακας blocked συμπληρωθεί, τότε καμια επιλογή διάστασης δεν πρόκειται να μας δώσει αποδεκτή λύση και έτσι αυξάνουμε το tolerance. Η αύξηση γίνεται ως εξής:

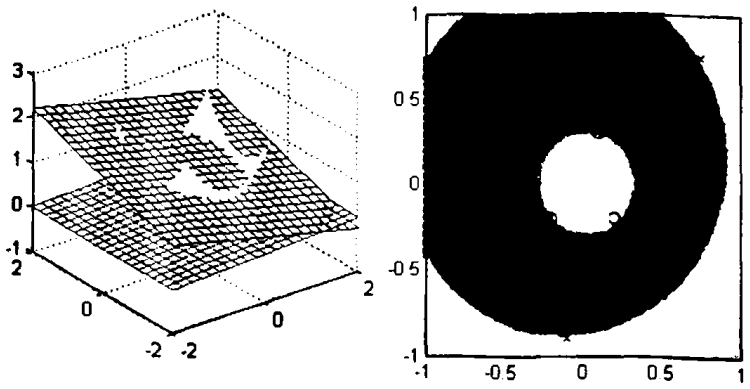
- Αν το tolerance είναι 0, το αλλάζουμε σε 1.
- Αν το tolerance είναι μη μηδενικό, το διπλασιάζουμε.

Κάθε φορά που αυξάνεται το tolerance επαναφέρεται και ο πίνακας blocked στην αρχική του κατάσταση. Υπενθυμίζουμε ότι κάθε επιτυχής προσθήκη νευρώνα μηδενίζει το tolerance και τον πίνακα blocked. Ο διπλασιασμός του tolerance όποτε υπάρχει ανάγκη μας εξασφαλίζει ότι το τελικό σφάλμα που θα προκαλέσει ένας νευρώνας είναι το πολύ διπλάσιο από το ελάχιστο δυνατό που θα μπορούσαμε να πετύχουμε.

## 3.8 Τοπική γενίκευση

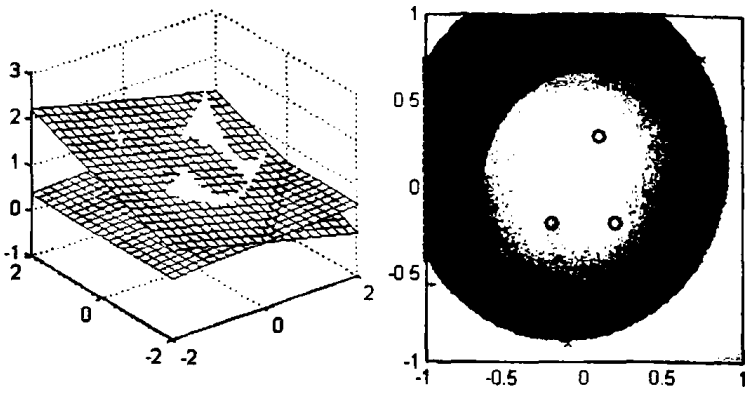
Με τον τρόπο που έχουμε επιλέξει να βρίσκουμε τα βάρη του νευρώνα χρησιμοποιώντας κάποια σημεία ελέγχου η επιφάνεια απόφασης κάθε νευρώνα περνάει ακριβώς από αυτά τα σημεία ελέγχου. Αυτή η λύση έχει προφανώς το ελάχιστο εμβαδο που μπορεί να έχει η επιφάνεια απόφασης που μαθαίνει τα ίδια δεδομένα. Αυτο θεωρούμε ότι δεν είναι το καλύτερο που μπορούμε να πετύχουμε, γιατί κατά πάσα πιθανότητα η περιοχή γύρω από αυτά τα σημεία ανήκει στην ίδια κατηγορία.

Θα ήταν επιθυμητό ο νευρώνας να καλύτει μια ευρύτερη περιοχή γύρω από τα σημεία τα οποία μαθαίνει. Υπενθυμίζουμε ότι ήδη έχουμε το μοναδιαίο κανονικό διάνυσμα του νευρώνα το οποίο δείχνει από την αντίθετη πλευρά των δεδομένων που έχει μάθει στα βάρη $w_1, \ldots, w_{d+1}$. Πριν επιστρέψουμε κάθε νευρώνα, μετακινούμε τα σημεία ελέγχου του νευρώνα κατά την κατεύθυνση




Σχήμα 3.1: Μια λύση χωρίς τοπική γενίκευση

που δείχνει το κανονικό αυτό διάνυσμα μια απόσταση ίση με τη μισή απόσταση του πλησιέστερου σημείου άλλης κατηγορίας.




Σχήμα 3.2: Η ίδια λύση με τοπική γενίκευση

## 3.9 Απάντηση σε ερωτήσεις

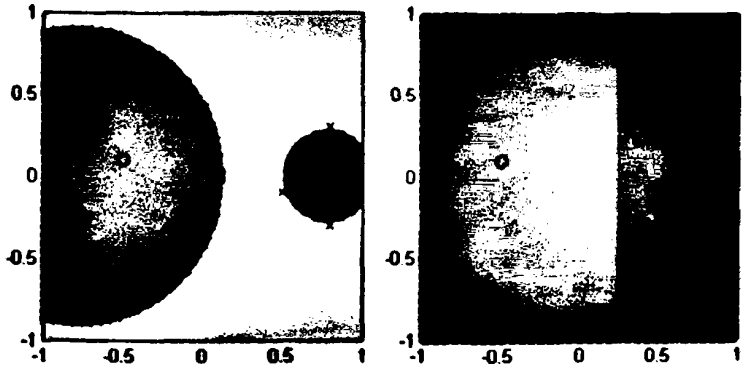
Στο δίκτυο που εκπαιδεύουμε οι νευρώνες έχουν φθίνουσα προτεραιότητα ως προς τη σειρά με την οποία προστέθηκαν. Αυτό σημαίνει πως ό,τι βρισκεται κάτω από τον πρώτο νευρώνα ταξινομείται στην κατηγορία του πρώτου νευρώνα. Θα πρέπει κάτι τέτοιο να μην ισχύει, για να ελέγξουμε τους επόμενους νευρώνες.

Όταν το δίκτυό μας ερωτηθεί για κάποιο νέο δεδομένο, τότε ξεκινώντας από τον πρώτο νευρώνα και συνεχίζοντας μέχρι τον τελευταίο ελέγχουμε αν το δεδομένο βρίσκεται κάτω από κάθε νευρώνα. Όταν το αποτέλεσμα του ελέγχου είναι θετικό, ταξινομούμε το δεδομένο στην κατηγορία αυτού του νευρώνα για τον οποίο πέτυχε ο έλεγχος για πρώτη φορά.

Υπάρχει το ενδεχόμενο να διατρέξουμε όλη τη λίστα των νευρώνων, χωρίς να καταφέρουμε να ταξινομήσουμε το δεδομένο. Σε αυτήν την περίπτωση το δίκτυο ουσιαστικά απαντάει 'Δεν ξέρω'.

Αυτό δεν είναι απαραίτητα αρνητικό, καθώς υποδηλώνει ότι στη συγκεκριμένη περιοχή του χώρου δεν υπάρχουν αρκετά πυκνά δεδομένα, ώστε να καλύψουμε την περιοχή με ένα νευρώνα του οποίου τα σημεία ελέγχου βρίσκονται μέσα σε αυτήν την περιοχή και άρα μας βοηθάει να εξάγουμε συμπεράσματα για τη δομή του συνόλου εκπαίδευσης.

Ένας τρόπος να απαντήσουμε σε τέτοιες περιπτώσεις είναι μέσω της απόστασης από κάθε νευρώνα. Αφού το δεδομένο δεν ταξινομήθηκε από κανέναν νευρώνα, ξέρουμε ότι η απόσταση από όλους τους νευρώνες που προκύπτει απευθείας από την εφαρμογή των συντεταγμένων στα βάρη του νευρώνα (αφού έχουμε το μοναδιαίο κανονικό διάνυσμα στα βάρη $w_1, \ldots, w_{d+1}$) είναι θετική, γιατί αν ήταν αρνητική από κάποιο νευρώνα, τότε θα ήταν από κάτω του και ο νευρώνας θα το ταξινομούσε. Άρα αρκεί να εντοπίσουμε το νευρώνα που απέχει τη μικρότερη (θετική) απόσταση από το δεδομένο και να επιστρέψουμε την κατηγορία αυτού του νευρώνα.




Σχήμα 3.3: Απάντηση με βάση την απόσταση

## 3.10 Επανεκπαίδευση μετά από επεξεργασία

Όπως έχουμε αναφέρει, ο αλγόριθμός μας μπορεί να μάθει ένα δεδομένο μιας κατηγορίας, ακόμα και αν αυτό περιβάλλεται από δεδομένα άλλης κατηγορίας. Επομένως έχει μεγάλη διαχωριστική ικανότητα. Το πρόβλημα που μπορεί να προκύψει είναι ότι, αν τα δεδομένα έχουν θόρυβο, το δίκτυο που θα εκπαιδεύσουμε θα μάθει και το θόρυβο.

Ενδεικτικό αυτής της περίπτωσης είναι ο υπερβολικά μεγάλος αριθμός νευρώνων στο τελικό δίκτυο. Μπορούμε όμως λογικά να παρατηρήσουμε πως οι νευρώνες που έμαθαν το θόρυβο θα είναι μεταξύ αυτών που έχουν μάθει λίγα δεδομένα.

Αντιμετωπίζουμε αυτό το πρόβλημα ορίζοντας ως χρήστες έναν ελάχιστο αριθμό δεδομένων που απαιτούμε να μαθαίνει ένας νευρώνας. Διατρέχουμε όλους τους νευρώνες. Για όσους δεν καλύπτουν τουλάχιστον όσα δεδομένα απαιτούμε, σημειώνουμε τα δεδομένα που καλύπτουν με την τιμή $-1$. Αφου διατρέξουμε τους νευρώνες, τότε ξεκαλύπτουμε όλα τα δεδομένα εκτός από αυτά που σημειώσαμε με $-1$ στο προηγούμενο βήμα. Ουσιαστικά 'σβήνουμε' τα δεδομένα που θεωρούμε θόρυβο. Τέλος, επανεκπαιδεύουμε το δίκτυο αγνοώντας το θόρυβο.

Ο μικρός αριθμός δεδομένων όμως, εκτός από θόρυβο, μπορεί να σημαίνει λεπτομέρεια. Αν ένα σύνολο δεδομένων περιέχει θόρυβο ή λεπτομέρεια μπορούμε να το καταλάβουμε από το κατά πόσο θα μειωθεί ο αριθμός των νευρώνων στο νέο δίκτυο. Αν είναι μικρή η μείωση, τότε πρόκειται για λεπτομέρεια στα σύνορα των κατηγοριών, αλλιώς μάλλον υπάρχει επικάλυψη κατηγοριών (θόρυβος).

## 3.11 Μετατροπή σε παραδοσιακό δίκτυο

Τώρα θα δούμε πως μπορούμε από το δίκτυο με προτεραιότητες που έχουμε εκπαιδεύσει να κατασκευάσουμε ένα ισοδύναμο παραδοσιακό νευρωνικό δίκτυο με επίπεδο εξόδου.

- Κάθε νευρώνας του δικτύου μας έχει ήδη τα κατάλληλα βάρη στις εισόδους του. Αυτοί οι νευρώνες αποτελούν το κρυμμένο επίπεδο.
- Θεωρούμε τη βηματική συνάρτηση ενεργοποίησης

$$g(y) = \begin{cases} 0, \text{ αν } y > 0 \\ 1, \text{ αν } y \leq 0 \end{cases}$$

  για κάθε νευρώνα του κρυμμένου επιπέδου αλλά και του νευρώνα εξόδου.
- Το βάρος κάθε νευρώνα με το νευρώνα εξόδου είναι $-2^{-id}$, αν ο νευρώνας μαθαίνει την κατηγορία 1 ή $+2^{-id}$, αν μαθαίνει την κατηγορία 0.

Ισχυριζόμαστε ότι αυτό το παραδοσιακό νευρωνικό δίκτυο διατηρεί τις προτεραιότητες που έχουμε ορίσει. Πράγματι, από τη συνάρτηση ενεργοποίησης φαίνεται ότι ένας νευρώνας πυροδοτείται, αν το δεδομένο βρίσκεται από χάτω του. Παρατηρούμε επίσης ότι το πρόσημο του αθροίσματος στο νευρώνα εξόδου καθορίζει την κατηγορία στην οποία θα ταξινομηθεί το δεδομένο.

Ας υποθέσουμε ότι ο νευρώνας με το μεγαλύτερο id που πυροδοτείται είναι ο $i$ και ότι μαθαίνει την κατηγορία 1. Άρα ο μεγαλύτερος όρος του αθροίσματος στο νευρώνα εξόδου είναι $-2^i$. Ακόμα κι αν όλοι οι υπόλοιποι νευρώνες πυροδοτείθηκαν με θετικό πρόσημο, ο νευρώνας $i$ θα καθορίσει το πρόσημο του αθροίσματος, καθώς

$$\sum_{j=i+1}^{\infty} 2^{-j} < 2^{-i} \iff -2^{-i} + \sum_{j=i+1}^{\infty} 2^{-j} < 0$$

και άρα η προτεραιότητα διατηρείται.

## 3.12 Ανάλυση

Κλείνοντας αυτό το κεφάλαιο θα δώσουμε μια εκτίμηση της πολυπλοκότητας του αλγόριθμού μας.

Έστω $d$ η διάσταση των δεδομένων, $n$ το πλήθος των δεδομένων και $m$ ο αριθμός των νευρώνων που θα έχει το δίκτυο στο τέλος. Προφανώς ο αλγόριθμος είναι output sensitive, αφού το μέγεθος της εξόδου εμφανίζεται στην πολυπλοκότητα.

- Το δίκτυο θα έχει $m$ νευρώνες
  - Για κάθε νευρώνα θα χρειαστεί να επεξεργαστούμε $O(n)$ σημεία.
    - Κάθε σημείο μας δίνει $O(d)$ επιλογές για την ενημέρωση των σημείων ελέγχου.
      - Η επίλυση του ομογενούς συστήματος μας κοστίζει $O(d^3)$ χρόνο.
      - Ο υπολογισμός του σφάλματος διατρέχει $O(n)$ δεδομένα και εκτελεί $O(d)$ πράξεις για τον υπολογισμό της απόστασης.

Συνολικά η πολυπλοκότητα είναι $O(mnd^4 + mn^2d^2)$. Αν μάλιστα υποθέσουμε ότι $n \gg d$ είναι απλώς $O(mn^2d^2)$. Επισημαίνουμε ότι μια εποχή backpropagation τρέχει σε $O(mnd)$ χρόνο, και άρα ο αλγόριθμός μας τρέχει σε αντίστοιχο χρόνο με $nd$ εποχές backpropagation.

# Κεφάλαιο 4

# Πειραματικά αποτελέσματα

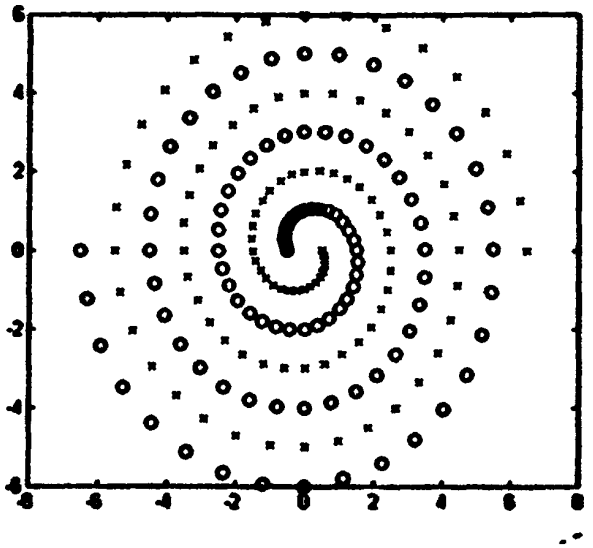
Σε αυτό το κεφάλαιο θα παρουσιάσουμε τα αποτελέσματα διαφόρων πειραμάτων για την αξιολόγηση της επίδοσης του αλγορίθμου μας. Αρχικά θα δείξουμε τη λειτουργία του σε κάποια δισδιάστατα προβλήματα στα οποία μπορούμε να έχουμε εποπτεία. Μετά θα παρουσιάσουμε κάποια πειράματα που θεωρούμε ενδεικτικά για τη λειτουργία και τις ιδιότητες του αλγορίθμου. Τέλος, θα παραθέσουμε τις επιδόσεις του αλγόριθμού μας σε διάφορα σύνολα δεδομένων.

## 4.1 Δισδιάστατα προβλήματα

Στην πρώτη ενότητα του κεφαλαίου θα δούμε πως επιλύει ο αλγόριθμός μας μερικά δισδιάστατα προβλήματα. Αυτό γίνεται έτσι ώστε να μπορούμε να καταλάβουμε οπτικά πώς δουλεύει ο αλγόριθμος και τι διαχωρισμούς επιτυγχάνει. Για κάθε dataset θα περιγράφουμε τη μορφή των δεδομένων και θα περιλαμβάνουμε μια εικόνα του. Στη συνέχεια θα δείχνουμε με μια εικόνα πως ταξινομεί ένα δίκτυο που κατασκεύασε ο αλγόριθμός μας, χρησιμοποιώντας ολόκληρο το dataset, το χώρο γύρω από τα δεδομένα, καθώς και μια εικόνα με τους νευρώνες σε σειρά προτεραιότητας. Ως συμπέρασμα θα σχολιάζουμε το αποτέλεσμα που μας έδωσε ο αλγόριθμος και θα συγκρίνουμε το αποτέλεσμα με μια λύση ενός δικτύου MLP ή CBP εκπαιδευμένου με τη μέθοδο Levenberg-Marquardt.

### 4.1.1 Τα δυο spirals

Πρόκειται για ένα πρόβλημα δυο κατηγοριών, η καθεμία από τις οποίες βρίσκεται τυλιγμένη μέσα στην άλλη. Είναι ένα ιδιαίτερα δύσκολο πρόβλημα, αν η λύση περιέχει μόνο γραμμές ως επιφάνειας απόφασης.

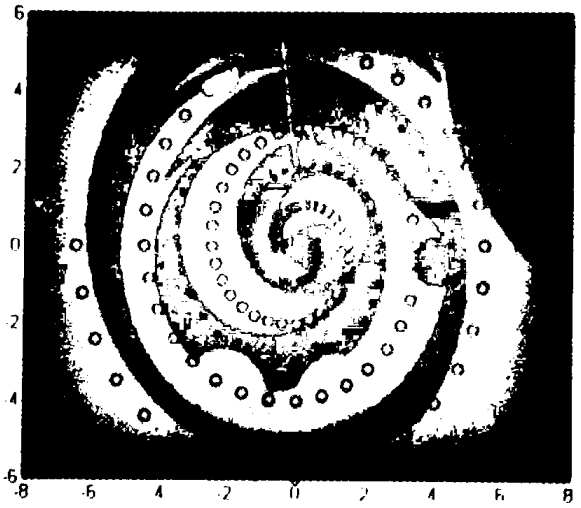



Σχήμα 4.1: Το dataset για τα δυο spirals

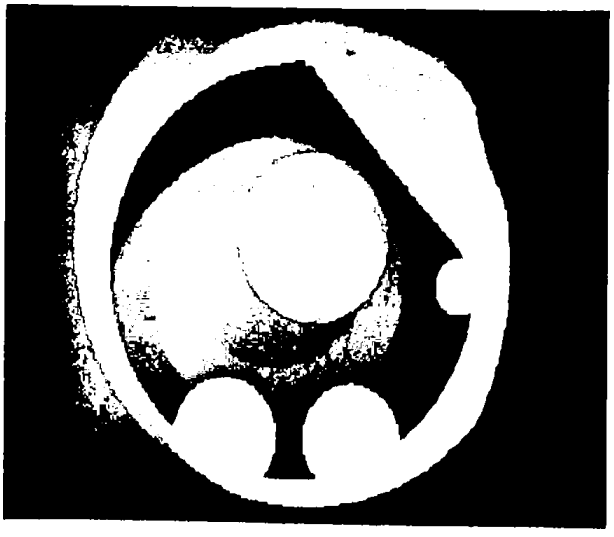
Χρησιμοποιήσαμε όλα τα σημεία για training set. Το δίκτυο έμαθε σωστά όλα τα δεδομένα. Παρατηρούμε ότι οι κλειστές καμπύλες αποτελούνται από την ένωση κυκλικών τόξων, όπως είχαμε αναφέρει στις ιδιότητες του παραβολοειδούς, και άρα η προβολή όντως μας δίνει περισσότερες δυνατότητες.

Περνώντας στους νευρώνες βλέπουμε τους επικαλυπτόμενους κύκλους της λύσης. Προφανώς ένας κύκλος έχει μεγαλύτερη προτεραιότητα από αυτούς που καλύπτει και μικρότερη από αυτούς που τον καλύπτουν. Παρατηρούμε επίσης ότι, εκτός από μερικούς κύκλους που υπάρχουν για να μάθουν λίγα δεδομένα που το δίκτυο δε μπορούσε να μάθει αλλιώς, οι κύκλοι ιδιαίτερα στο κέντρο έχουν σπυροειδή τοποθέτηση με αυξανόμενη ακτίνα.

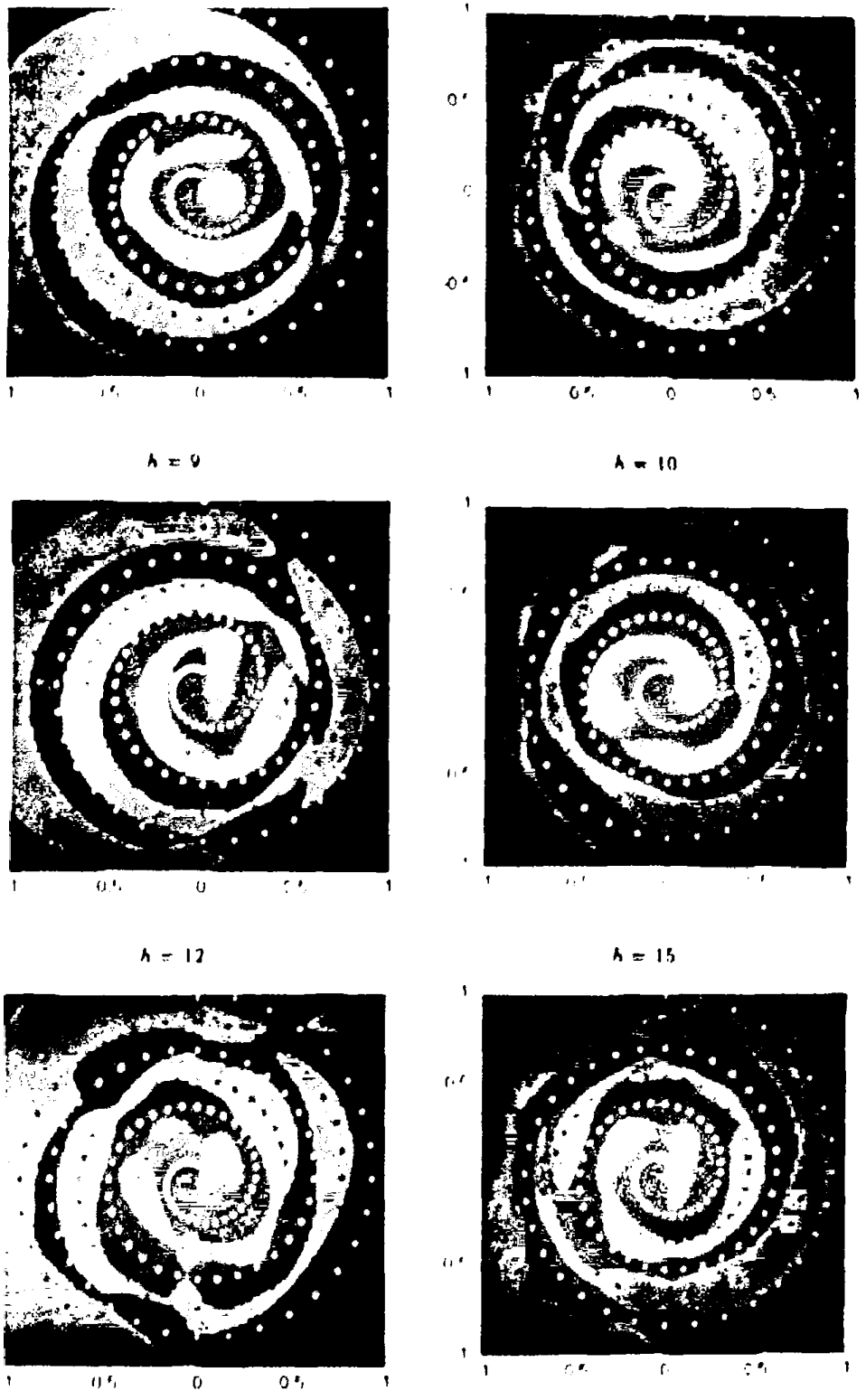
Σε σύγκριση με το δίκτυο MLP βλέπουμε πως οι καμπύλες που έδωσε ο αλγόριθμός μας είναι πολύ πιο ομαλές και μάλιστα χρησιμοποιώντας μικρότερο αριθμό νευρώνων, ενω το δίκτυο MLP έχει επίσης και μη μηδενικό σφάλμα εκπαίδευσης.



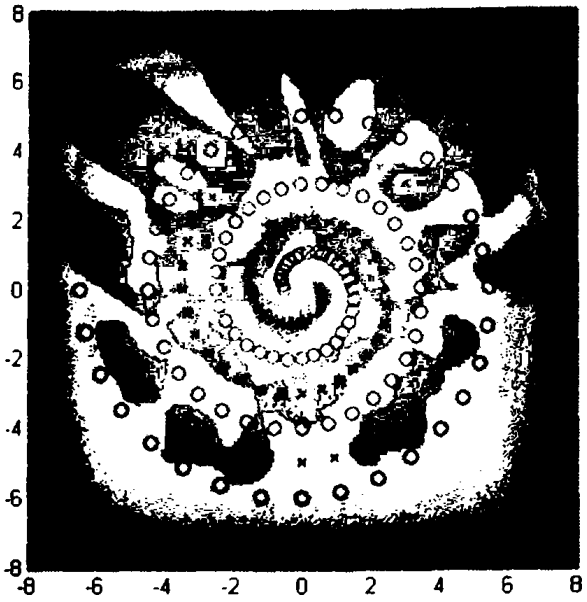

Σχήμα 4.2: Μια λύση για τα δυο spirals



Σχήμα 4.3: Οι νευρώνες της λύσης




1: Διάφορες λύσεις για το τα δυο spirals από δίκτυο CBP όπου $h$ ο
ων κρυμμένων νευρώνων [3]




Σχήμα 4.5: Μια λύση για τα δυο spirals από δίκτυο MLP με 100 νευρώνες

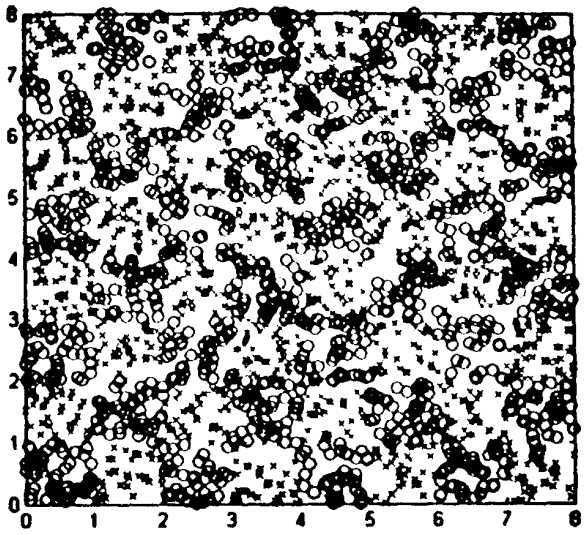
Σε σύγκριση με το δίκτυο CBP παρατηρούμε πως οι λύσεις του CBP με λίγους κρυμμένους νευρώνες παρουσιάζουν αρκετές ομοιότητες με τη δική μας λύση, αλλά όσο ο αριθμός των νευρώνων αυξάνεται το CBP δείχνει τάσεις υπερεκπαίδευσης σε αντίθεση με τον αλγόριθμό μας, οποίος χρησιμοποίησε περισσότερους νευρώνες χωρίς να υπερεκπαιδεύσει το δίκτυο.

### 4.1.2 Η σκακιέρα

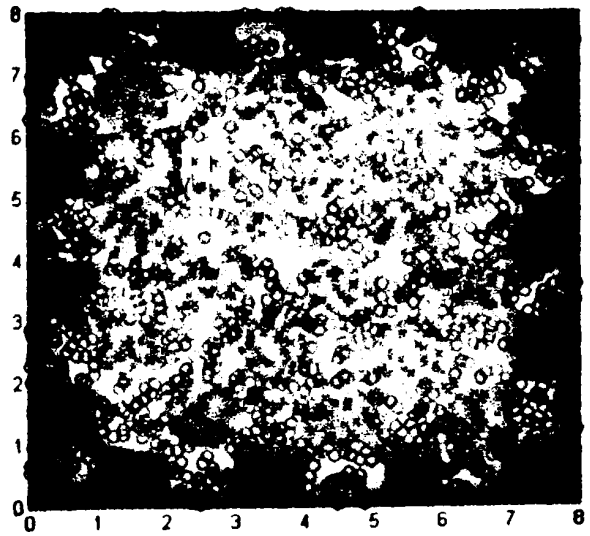
Το πρόβλημα είναι ο διαχωρισμός των λευκών και μαύρων τετραγώνων μιας σκακιέρας. Είναι παρόμοιο με το πρόβλημα των εναλλασσόμενων ετικετών και είναι επίσης δύσκολο να επιλυθεί χρησιμοποιώντας γραμμές.

Χρησιμοποιήσαμε όλα τα σημεία για training set. Το δίκτυο έμαθε πολύ καλά το εσωτερικό κάθε τετραγώνου και το περίγραμμα, αν και δεν είναι τετραγωνικό, είναι χαρακτηριστικό της δειγματοληψίας των σημείων που κάναμε. Αφού δεν έχουμε αρκετά δεδομένα στα σύνορα των τετραγώνων κάτι τέτοιο ήταν αναμενόμενο.

Κοιτώντας τους νευρώνες παρατηρούμε ότι όντως στο εσωτερικό κάθε τετραγώνου υπάρχει ένας κύκλος που περιέχει τα περισσότερα σημεία του τετραγώνου. Επίσης η μάθηση ξεκίνησε από το περίγραμμα της σκακιέρας και στη συνέχεια προχώρησε προς το εσωτερικό της.



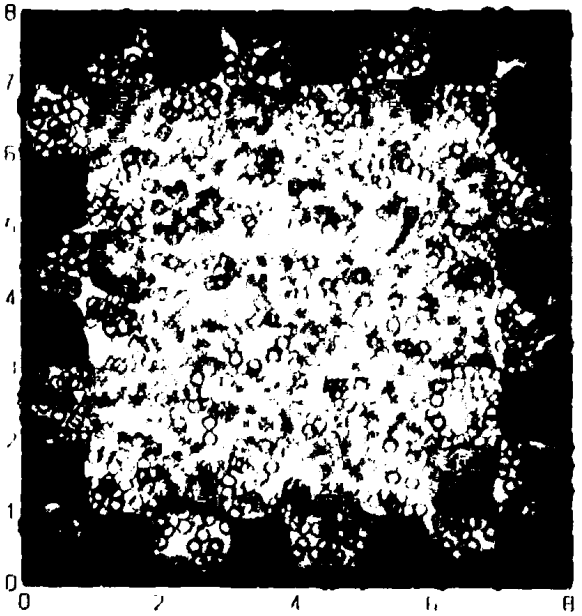

Σχήμα 4.6: Το dataset για τη σκακιέρας




Σχήμα 4.7: Μια λύση για τη σκακιέρα

Βλέπουμε ότι η παραδοσιακή λύση (MLP) φαίνεται να είναι πιο ομαλή, αλλά ούτε αυτό βρήκε την τετραγωνική μορφή μιας ιδανικής λύσης και επισημαίνουμε ακόμα ότι χρησιμοποίησε αρκετά περισσότερους νευρώνες.

Σχήμα 4.8: Οι νευρώνες της λύσης

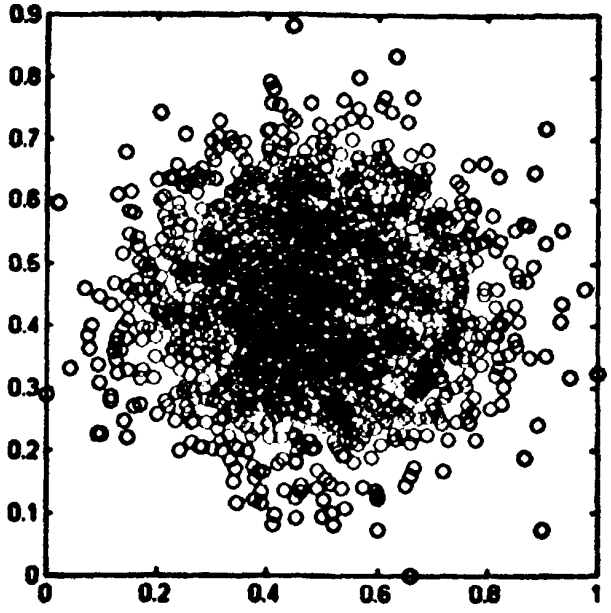



Σχήμα 4.9: Μια λύση για τη σκακιέρα από δίκτυο MLP με 256 νευρώνες

### 4.1.3 Τα τέσσερα νέφη

Πρόκειται ουσιαστικά για τέσσερις Γκαουσιανές κατανομές, τρεις από τις οποίες ανήκουν στη μια κατηγορία και η τέταρτη κατανομή με μεγάλη τυπική

απόκλειση που περιέχει τις πρώτες δυο ανήκει στην άλλη κατηγορία.

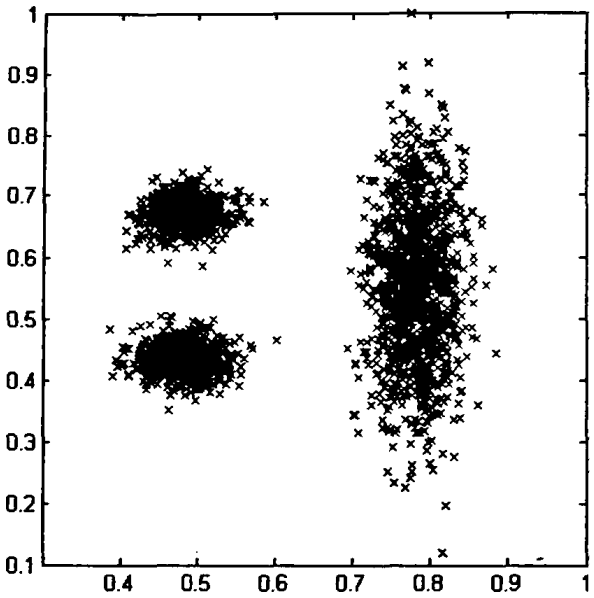



Σχήμα 4.10: Το dataset για την κατηγορία 0

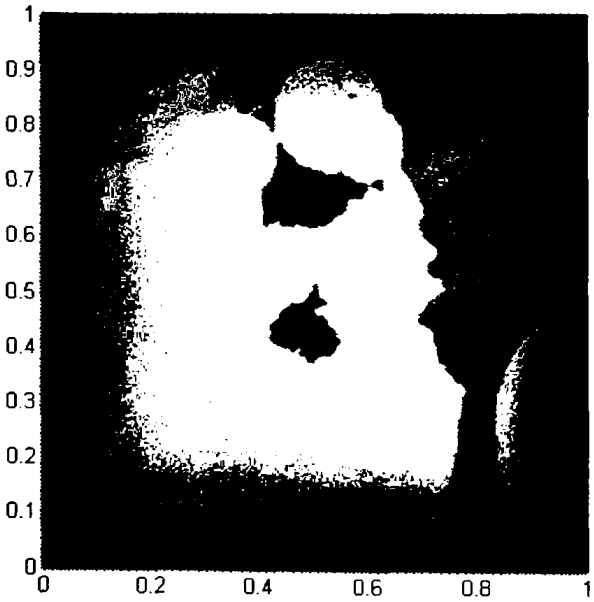
Επειδή το dataset έχει θόρυβο, χρησιμοποιήσαμε την τεχνική του κλαδέματος με επανεκπαίδευση. Προφανώς η λύση δεν έχει μάθει σωστά όλα τα δεδομένα, αλλά τα δυο νέφη που περιέχονται στο άλλο είναι συμπαγή. Αξιοσημείωτο είναι επίσης ότι σε περιοχές όπου δεν υπάρχουν αρκετά σημεία, η λύση μπορεί να μαθαίνει τα δεδομένα, αλλά η μορφή της δεν πλησιάζει το ιδανικό αποτέλεσμα.

Περνώντας στους νευρώνες βλέπουμε ότι λίγοι σχετικά νευρώνες χρησιμοποιήθηκαν για τις δυο κατανομές ενώ οι περισσότεροι χρησιμοποιήθηκαν για να μάθουν την άλλη χωρίς να προκαλέσουν σφάλμα.

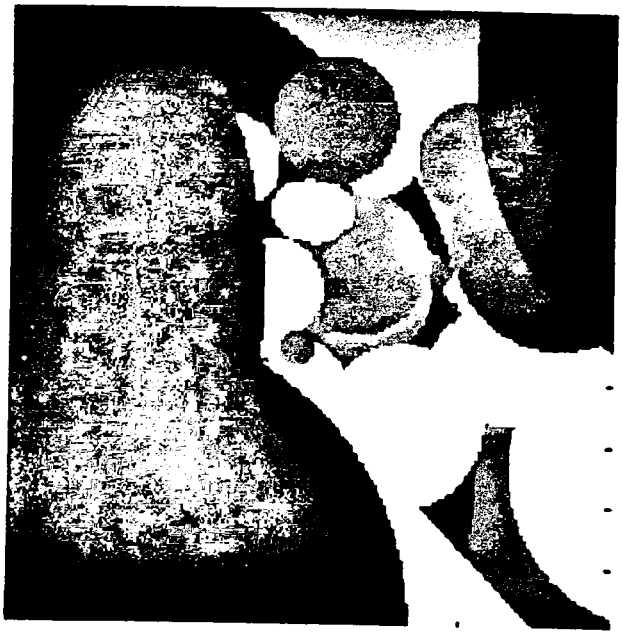
Συκρίνοντας τη λύση που έδωσε το MLP με τη δική μας βλέπουμε ότι είναι πιο ομαλή. Ισχυριζόμαστε όμως ότι οι δυο λύσεις είναι ισοδύναμες, καθώς πετυχαίνουν το θεωρητικό βέλτιστο, όπως θα δούμε παρακάτω.



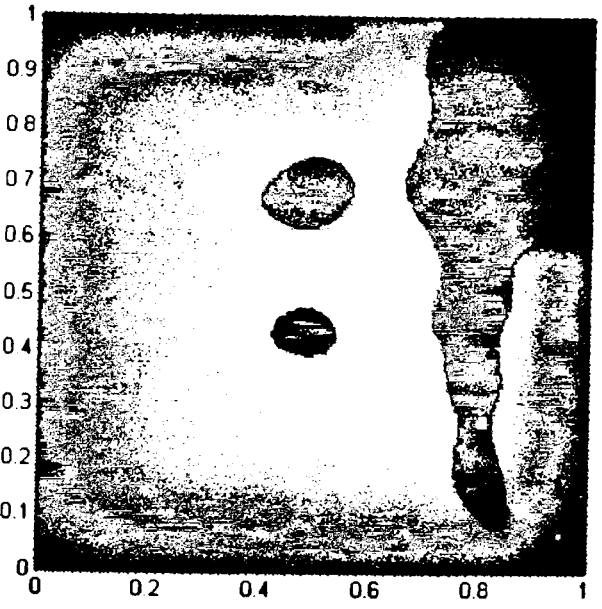

Σχήμα 4.11: Το dataset για την κατηγορία 1




Σχήμα 4.12: Μια λύση για τα τέσσερα νέφη

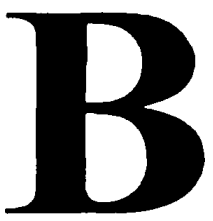


Σχήμα 4.13: Οι νευρώνες της λύσης



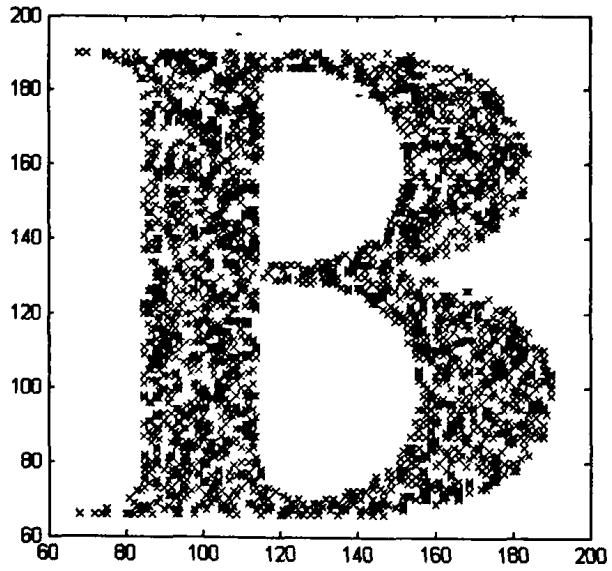

α λύση για τα τέσσερα νέφη από δίκτυο MLP με 20 νευρώνες

### 4.1.4 Το γράμμα Β'



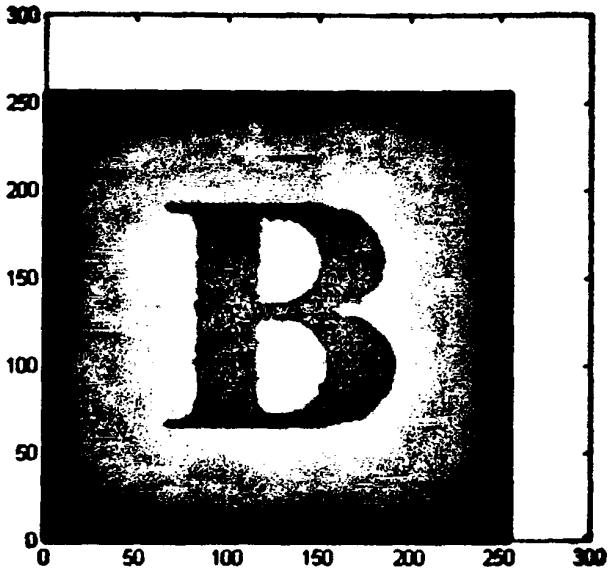

Σχήμα 4.15: Η αρχική εικόνα του Β'

Το γράμμα Β' όπως βλέπουμε περιέχει αρκετά ενδιαφέροντα σημεία από την άποψη του διαχωρισμού. Περιέχει καμπύλες γραμμές, τρεις ευθείες, περιοχές μιας κατηγορίας που περικυκλώνονται από την άλλη (τρύπες) και επίσης μερικές μη κυρτές γωνίες.




Σχήμα 4.16: Η δειγματοληψία που κάναμε από το Β'

Παρατηρούμε ότι η λύση δε δυσκολεύτηκε ιδιαίτερα από τις τρύπες. Προσέγγισε αρκετά καλά τις καμπύλες και τις ευθείες (με λίγο 'θόρυβο' στις γραμμές) ενώ έμαθε πολύ καλά και τις δυο 'ουρές' που προεξέχουν αριστερά από το γράμμα.

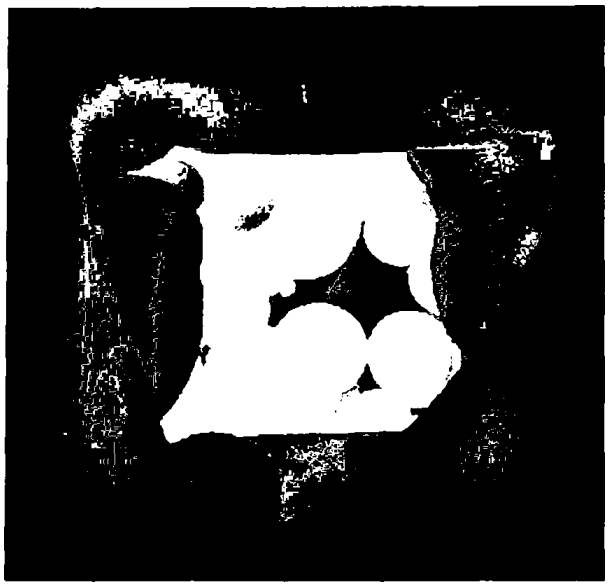



Σχήμα 4.17: Μια λύση για το Β'

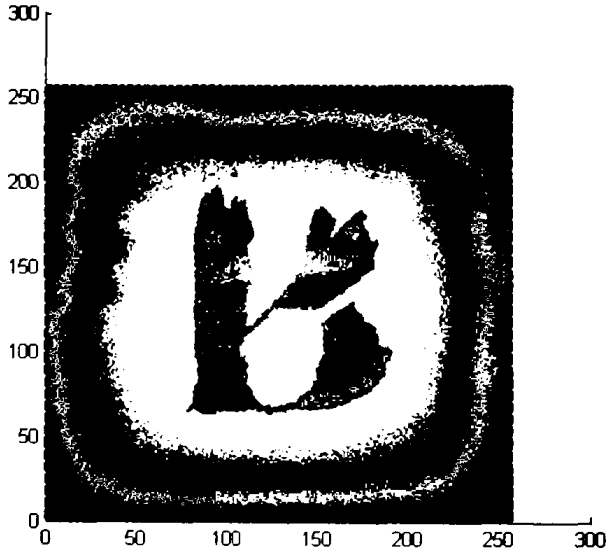
Βλέποντας τους νευρώνες διαπιστώνουμε ότι στην αρχή οι νευρώνες μάθαιναν πολλά σημεία και όσο προχωρούσε η εκπαίδευση προστίθονταν μικρότεροι κύκλοι για να μάθουν τις λεπτομέρειες.

Σε σύγκριση με τη λύση που έδωσε το δίκτυο MLP η λύση που έδωσε ο αλγόριθμός μας είναι εμφανέστατα καλύτερη και πάλι χρησιμοποιώντας λιγότερους νευρώνες.

Περνώντας στο δίκτυο CBP βλέπουμε ότι αποτελεί σαφώς βελτίωση σε σχέση με το MLP, αλλά δεν πλησιάζει την πραγματική λύση τόσο, όσο η λύση που έδωσε ο αλγόριθμός μας. Σημειώνουμε επίσης ότι η εκπαίδευση του CBP ήταν δύσκολη. Αν η αρχικοποίηση ήταν καλή, τότε η μέθοδος levenberg-Marquardt τερμάτιζε γρήγορα, αλλά αν δεν ήταν καλή η αρχικοποίηση τότε σταματούσε πρόωρα η μέθοδος.



Σχήμα 4.18: Οι νευρώνες της λύσης

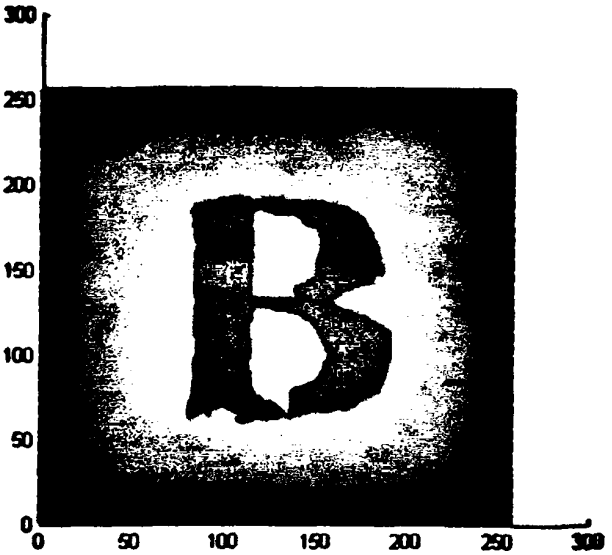



Σχήμα 4.19: Μια λύση για το Β΄ από δίκτυο MLP με 100 νευρώνες

## 4.2 Ειδικά ζητήματα

Τώρα θα δούμε με τη βοήθεια ορισμένων πειραμάτων τη συμπεριφορά του αλγορίθμου μας σε διάφορες ιδιότητες του συνόλου εκπαίδευσης. Αρχικά θα


Σχήμα 4.20: Μια λύση για το Β' από δίκτυο CBP με 50 νευρώνες




Σχήμα 4.21: Μια λύση για το Β' από δίκτυο CBP με 100 νευρώνες

δούμε πώς συμπεριφέρεται σε περίπτωση που υπάρχει επικάλυψη κατηγοριών, μετά θα διαπιστώσουμε πως όταν τα δεδομένα έχουν πολύ μικρές ή μηδενικές τιμές εμφανίζονται αριθμητικά σφάλματα, όπως αναφέραμε ήδη στην ενότητα

3.7, και τέλος θα δούμε πως όσο πιο πυκνή και αντιπροσωπευτική είναι η δειγματοληψία που κάνουμε στα δεδομένα τόσο περισσότερο πλησιάζει την τέλεια λύση ο αλγόριθμος.

### 4.2.1 Επικάλυψη κατηγοριών

Όπως είδαμε στο σύνολο δεδομένων των 'τεσσάρων νεφών' η κατανομή της μιας κατηγορίας περιείχε τις κατανομές τις άλλης. Όπως είναι φυσικό τα σημεία μιας κατηγορίας που σύμφωνα με τις πιθανότητες των κατανομών ανήκουν στην άλλη κατηγορία θεωρούνται θόρυβος. Εφαρμόζουμε λοιπόν την τεχνική κλάδεμα και επανεκπαίδευση και παρατηρούμε τα εξής σε δέκα εκτελέσεις του αλγορίθμου πριν και μετά το κλάδεμα:

| | Νευρώνες αρχικά | Επίδοση στο validation | Νευρώνες μετά το κλάδεμα | Επίδοση στο validation |
|---|---|---|---|---|
| 1. | 429 | 425/500 | 36 | 444/500 |
| 2. | 421 | 410/500 | 59 | 427/500 |
| 3. | 408 | 427/500 | 35 | 439/500 |
| 4. | 412 | 415/500 | 59 | 434/500 |
| 5. | 417 | 411/500 | 39 | 434/500 |
| 6. | 407 | 420/500 | 33 | 439/500 |
| 7. | 405 | 409/500 | 40 | 434/500 |
| 8. | 400 | 414/500 | 43 | 438/500 |
| 9. | 425 | 441/500 | 40 | 448/500 |
| 10. | 417 | 440/500 | 33 | 443/500 |
| Μέσος όρος | 414.1 | 421.2/500 | 41.7 | 438/500 |

Άρα η εφαρμογή της τεχνικής οδήγησε σε λύσεις που είχαν κατά μέσο όρο το 10% του μεγέθους της αρχικής λύσης. Επίσης αξίζει να σημειωθεί ότι η γενικευτική ικανότητα της λύσης όπως αυτή φαίνεται από την επίδοση στο validation αυξήθηκε. Επομένως μπορούμε με ασφάλεια να υποθέσουμε ότι, αν χρησιμοποιώντας επανεκπαίδευση μετά από κλάδεμα μειωθεί δραστικά ο αριθμός των νευρώνων της λύσης αυξάνοντας παράλληλα τη γενικευτική ικανότητα του δικτύου, τότε τα δεδομένα έχουν θόρυβο.

Ας εξετάσουμε τώρα πως θα μεταβληθεί η λύση μας αν εφαρμόσουμε την ίδια τεχνική στο σύνολο δεδομένων pima. Για δέκα εκτελέσεις χρησιμοποιώντας κλάδεμα και επανεκπαίδευση έχουμε:

| | Νευρώνες αρχικά | Επίδοση στο validation | Νευρώνες μετά το χλάδεμα | Επίδοση στο validation |
|---|---|---|---|---|
| 1. | 48 | 35/50 | 47 | 34/50 |
| 2. | 54 | 31/50 | 50 | 36/50 |
| 3. | 51 | 28/50 | 41 | 33/50 |
| 4. | 55 | 33/50 | 49 | 32/50 |
| 5. | 56 | 40/50 | 52 | 36/50 |
| 6. | 58 | 32/50 | 45 | 31/50 |
| 7. | 49 | 33/50 | 44 | 35/50 |
| 8. | 55 | 34/50 | 43 | 30/50 |
| 9. | 59 | 36/50 | 46 | 39/50 |
| 10. | 50 | 37/50 | 53 | 31/50 |
| Μέσος όρος | 53.5 | 33.9/50 | 47 | 33.7/50 |

Βλέπουμε λοιπόν ότι το μέγεθος της νέας λύσης είναι κατα μέσο όρο το 87,8% της αρχικής που μπορεί να θεωρηθεί στην ίδια πολυπλοκότητα, ενώ παράλληλα μειώθηκε ελάχιστα και η γενικευτική ικανότητα της λύσης. Άρα όταν χρησιμοποιώντας χλάδεμα και επανεκπαίδευση δε μειωθεί δραστικά ο αριθμός των νευρώνων και η λύση χάσει σε γενικευτική ικανότητα, τότε το σύνολο δεδομένων μάλλον έχει λεπτομέρειες που θα πρέπει να καλυφθούν από νευρώνες που μαθαίνουν τα λίγα αυτά σημεία κάθε λεπτομέρειας.

### 4.2.2 Αριθμητικά σφάλματα

Θα εξετάσουμε τώρα το πρόβλημα parity, ή αλλιώς το πρόβλημα της ισοτιμίας. Έστω ότι έχουμε $d$ bits. Για κάθε δυνατή $d$-άδα από bits θέλουμε η έξοδος να είναι τέτοια, ώστε αν αθροίσουμε όλα τα bits και την έξοδο το αποτέλεσμα να είναι άρτιος αριθμός (ή περιττός, ανάλογα αν θέλουμε άρτια η περιττή ισοτιμία). Το πρόβλημα XOR που είδαμε ότι δεν είναι γραμμικά διαχωρίσιμο είναι η υποπερίπτωση της άρτιας ισοτιμίας δυο bits. Πρακτικά η λύση του προβλήματος από νευρωνικό δίκτυο δεν έχει καμία εφαρμογή, άλλα θεωρείται μέτρο σύγκρισης της εκπαιδευτικής ικανότητας ενός αλγορίθμου.

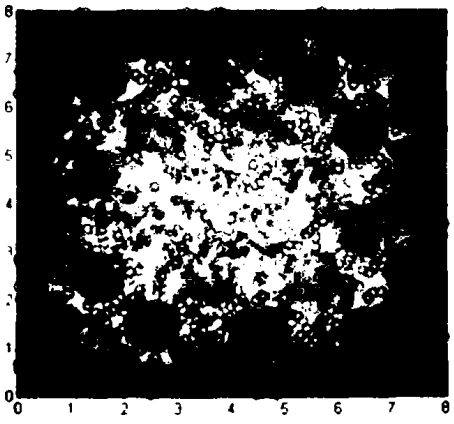
Αρχικά προσπαθούμε να λύσουμε το πρόβλημα αναθέτοντας σε $d = 8$ bits τις τίμές 0 και 1 και μετά δοκιμάζουμε αντί για 0 να χρησιμοποιούμε την τιμή $-1$. Μετά από δέκα εκτελέσεις του αλγορίθμου για κάθε περίπτωση μετράμε το σφάλμα εκπαίδευσης:

| | 0,1 | -1,1 |
|---|---|---|
| 1. | 43 | 0 |
| 2. | 46 | 0 |
| 3. | 43 | 0 |
| 4. | 91 | 0 |
| 5. | 45 | 0 |
| 6. | 70 | 0 |
| 7. | 43 | 0 |
| 8. | 43 | 0 |
| 9. | 88 | 0 |
| 10. | 43 | 0 |
| Μέσος Όρος | 55,5 | 0 |

Παρατηρούμε πως μπορούμε να παρακάμψουμε τις δυσκολίες που προκαλεί η ύπαρξη μηδενικών στα δεδομένα (στοιχείο που θεωρείται απορροφητικό) αν μετακινήσουε τα σημεία κατά τους άξονες μακριά από το μηδέν.

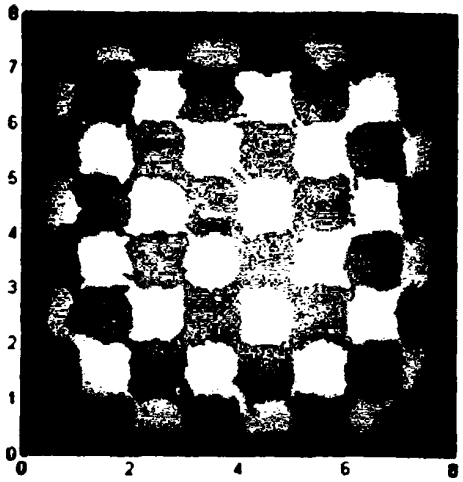
### 4.2.3 Δειγματοληψία

Θα επανέλθουμε τώρα στο πρόβλημα της σκακιέρας και θα δούμε πως μεταβάλεται η λύση κάνοντας πυκνότερη δειγματοληψία των σημείων. Θα παραθέσουμε μια σειρά εικόνων που αναπαριστούν λύσεις με αυξανόμενο αριθμό σημείων εκπαίδευσης.

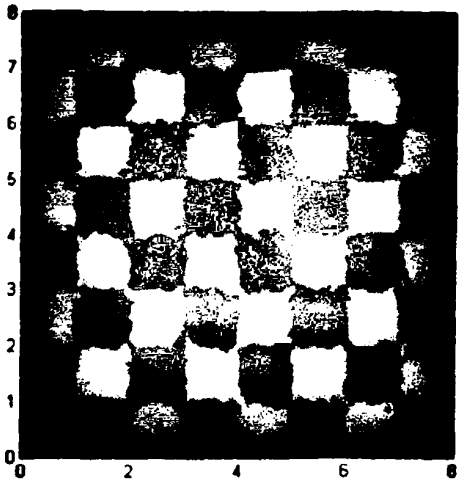



Σχήμα 4.22: 2560 σημεία

Παρατηρούμε ότι όσο αυξάνεται ο αριθμός των σημείων, και άρα όσο πιο πυκνή γίνεται η δειγματοληψία, τόσο πλησιάζει η λύση που δίνει ο αλγόριθμος στην τέλεια λύση.




Σχήμα 4.23: 6400 σημεία




Σχήμα 4.24: 12800 σημεία

## 4.3 Επιδόσεις σε γνωστά σύνολα δεδομένων

Τώρα θα παρουσιάσουμε τα πειραματικά αποτελέσματα με τα οποία θα αξιολογήσουμε τον αλγόριθμό μας. Κάθε φορά η εκπαίδευση χρησιμοποιούσε το 80% των δεδομένων για training set, το 10% για validation set και το 10% για test set. Χρησιμοποιήσαμε τη μέθοδο 10-fold cross-validation εκπαιδεύοντας 10 δίχτυα για κάθε κύκλο δεδομένων (συνολικά 100 δίχτυα).

### 4.3.1 bupa

$n = 345$ $d = 6$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 37 | 0 | 25/35 | 11/35 |
| 2. | 36 | 0 | 25/35 | 13/35 |
| 3. | 38 | 0 | 26/35 | 12/35 |
| 4. | 35 | 0 | 27/35 | 9/35 |
| 5. | 40 | 0 | 23/35 | 11/35 |
| 6. | 38 | 0 | 26/35 | 14/35 |
| 7. | 33 | 0 | 26/35 | 13/35 |
| 8. | 34 | 0 | 20/35 | 11/35 |
| 9. | 38 | 0 | 26/35 | 12/35 |
| 10. | 35 | 0 | 22/35 | 10/35 |

Μέσος αριθμός σφαλμάτων 11.6 (33.6%)

### 4.3.2 cleveland

$n = 297$ $d = 13$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 28 | 0 | 26/30 | 10/30 |
| 2. | 25 | 0 | 24/30 | 4/30 |
| 3. | 23 | 0 | 24/30 | 9/30 |
| 4. | 30 | 0 | 23/30 | 7/30 |
| 5. | 26 | 0 | 24/30 | 6/30 |
| 6. | 21 | 0 | 24/30 | 7/30 |
| 7. | 22 | 0 | 23/30 | 6/30 |
| 8. | 29 | 0 | 24/30 | 7/30 |
| 9. | 24 | 0 | 22/30 | 5/30 |
| 10. | 25 | 0 | 25/30 | 11/30 |

Μέσος αριθμός σφαλμάτων 7.2 (24.2%)

### 4.3.3 clouds

$n = 5000$ $d = 2$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 41 | 0 | 446/500 | 56/500 |
| 2. | 30 | 0 | 450/500 | 61/500 |
| 3. | 41 | 0 | 438/500 | 66/500 |
| 4. | 37 | 0 | 438/500 | 63/500 |
| 5. | 33 | 0 | 438/500 | 64/500 |
| 6. | 29 | 0 | 446/500 | 60/500 |
| 7. | 39 | 0 | 443/500 | 60/500 |
| 8. | 29 | 0 | 449/500 | 57/500 |
| 9. | 42 | 0 | 448/500 | 52/500 |
| 10. | 29 | 0 | 449/500 | 55/500 |

Μέσος αριθμός σφαλμάτων 59.4 (11.8%)

### 4.3.4 iris

$n = 150$ $d = 4$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 9 | 0 | 14/15 | 1/15 |
| 2. | 8 | 0 | 15/15 | 1/15 |
| 3. | 7 | 0 | 15/15 | 1/15 |
| 4. | 7 | 0 | 15/15 | 0/15 |
| 5. | 8 | 0 | 15/15 | 3/15 |
| 6. | 8 | 0 | 15/15 | 2/15 |
| 7. | 8 | 0 | 14/15 | 0/15 |
| 8. | 8 | 0 | 15/15 | 2/15 |
| 9. | 10 | 0 | 15/15 | 1/15 |
| 10. | 8 | 0 | 15/15 | 1/15 |

Μέσος αριθμός σφαλμάτων 1.2 (8%)

### 4.3.5 phoneme

$n = 5404$ $d = 5$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 283 | 0 | 460/540 | 92/540 |
| 2. | 287 | 0 | 466/540 | 70/540 |
| 3. | 304 | 0 | 477/540 | 74/540 |
| 4. | 302 | 0 | 483/540 | 93/540 |
| 5. | 290 | 0 | 468/540 | 74/540 |
| 6. | 292 | 0 | 469/540 | 87/540 |
| 7. | 271 | 0 | 468/540 | 90/540 |
| 8. | 273 | 0 | 471/540 | 88/540 |
| 9. | 293 | 0 | 457/540 | 68/540 |
| 10. | 285 | 0 | 480/540 | 88/540 |

Μέσος αριθμός σφαλμάτων 82.4 (15.2%)

### 4.3.6 pima

$n = 500$ $d = 8$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 51 | 0 | 36/50 | 16/50 |
| 2. | 49 | 0 | 39/50 | 19/50 |
| 3. | 46 | 0 | 38/50 | 17/50 |
| 4. | 45 | 0 | 38/50 | 16/50 |
| 5. | 48 | 0 | 39/50 | 17/50 |
| 6. | 49 | 2 | 41/50 | 6/50 |
| 7. | 53 | 0 | 40/50 | 16/50 |
| 8. | 50 | 0 | 38/50 | 19/50 |
| 9. | 48 | 0 | 35/50 | 17/50 |
| 10. | 43 | 0 | 35/50 | 14/50 |

Μέσος αριθμός σφαλμάτων 15.7 (31.4%)

### 4.3.7 segment

$n = 2310$ $d = 18$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 54 | 0 | 220/231 | 13/231 |
| 2. | 55 | 0 | 224/231 | 13/231 |
| 3. | 59 | 0 | 223/231 | 15/231 |
| 4. | 61 | 0 | 218/231 | 11/231 |
| 5. | 59 | 0 | 218/231 | 10/231 |
| 6. | 52 | 0 | 222/231 | 12/231 |
| 7. | 54 | 0 | 225/231 | 21/231 |
| 8. | 59 | 0 | 218/231 | 9/231 |
| 9. | 63 | 0 | 222/231 | 14/231 |
| 10. | 58 | 0 | 220/231 | 19/231 |

Μέσος αριθμός σφαλμάτων 13.7 (5%)

### 4.3.8 vehicles

$n = 846$ $d = 18$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 61 | 0 | 62/85 | 19/85 |
| 2. | 68 | 0 | 67/85 | 24/85 |
| 3. | 66 | 0 | 64/85 | 20/85 |
| 4. | 62 | 0 | 66/85 | 22/85 |
| 5. | 67 | 0 | 66/85 | 21/85 |
| 6. | 67 | 0 | 62/85 | 23/85 |
| 7. | 71 | 0 | 69/85 | 35/85 |
| 8. | 68 | 0 | 62/85 | 29/85 |
| 9. | 77 | 0 | 66/85 | 31/85 |
| 10. | 63 | 0 | 61/85 | 21/85 |

Μέσος αριθμός σφαλμάτων 24.5 (28.9%)

### 4.3.9 vowels

$n = 990$ $d = 10$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 88 | 0 | 79/99 | 23/99 |
| 2. | 89 | 0 | 82/99 | 25/99 |
| 3. | 89 | 0 | 84/99 | 16/99 |
| 4. | 85 | 0 | 84/99 | 17/99 |
| 5. | 90 | 0 | 80/99 | 23/99 |
| 6. | 97 | 0 | 78/99 | 26/99 |
| 7. | 94 | 0 | 85/99 | 27/99 |
| 8. | 83 | 0 | 85/99 | 22/99 |
| 9. | 94 | 0 | 85/99 | 28/99 |
| 10. | 82 | 0 | 83/99 | 25/99 |

Μέσος αριθμός σφαλμάτων 23.2 (23.4%)

### 4.3.10 waveform

$n = 5000$ $d = 21$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 199 | 0 | 406/500 | 106/500 |
| 2. | 190 | 0 | 405/500 | 106/500 |
| 3. | 205 | 0 | 408/500 | 107/500 |
| 4. | 195 | 0 | 413/500 | 101/500 |
| 5. | 199 | 0 | 408/500 | 106/500 |
| 6. | 196 | 0 | 413/500 | 94/500 |
| 7. | 198 | 0 | 410/500 | 103/500 |
| 8. | 197 | 0 | 407/500 | 109/500 |
| 9. | 198 | 0 | 401/500 | 125/500 |
| 10. | 202 | 0 | 402/500 | 116/500 |

Μέσος αριθμός σφαλμάτων 107.3 (21.4%)

### 4.3.11 wdbc

$n = 569$ $d = 30$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 19 | 0 | 56/57 | 1/57 |
| 2. | 15 | 0 | 55/57 | 1/57 |
| 3. | 14 | 0 | 56/57 | 7/57 |
| 4. | 14 | 0 | 56/57 | 4/57 |
| 5. | 11 | 0 | 55/57 | 2/57 |
| 6. | 17 | 0 | 56/57 | 4/57 |
| 7. | 16 | 0 | 55/57 | 6/57 |
| 8. | 18 | 0 | 55/57 | 2/57 |
| 9. | 15 | 0 | 56/57 | 3/57 |
| 10. | 16 | 0 | 56/57 | 2/57 |

Μέσος αριθμός σφαλμάτων 3.2 (5.6%)

### 4.3.12 wine

$n = 178$ $d = 13$

| | Αριθμός νευρώνων | Σφάλμα εκπαίδευσης | Επιτυχίες στο validation | Σφάλμα στο test |
|---|---|---|---|---|
| 1. | 10 | 0 | 17/18 | 0/18 |
| 2. | 8 | 0 | 15/18 | 2/18 |
| 3. | 11 | 0 | 18/18 | 1/18 |
| 4. | 12 | 0 | 18/18 | 4/18 |
| 5. | 6 | 0 | 18/18 | 0/18 |
| 6. | 7 | 0 | 17/18 | 1/18 |
| 7. | 11 | 0 | 18/18 | 1/18 |
| 8. | 12 | 0 | 18/18 | 1/18 |
| 9. | 9 | 0 | 18/18 | 0/18 |
| 10. | 8 | 0 | 18/18 | 2/18 |

Μέσος αριθμός σφαλμάτων 1.2 (6.7%)

Για λόγους σύγκρισης θα παραθέσουμε τις επιδόσεις του αλγρίθμου μας δίπλα σε επιδόσεις που επιτεύθησαν με SVM, όπως αυτές παρουσιάζονται στο [8].

| Σύνολο δεδομένων | Επίδοση του αλγορίθμου μας | Επίδοση του SVM |
|---|---|---|
| bupa | 33.6% | 28.4% |
| iris | 8% | 4.7% |
| pima | 31.4% | 22.8% |
| segment | 5% | 3.1% |
| vehicles | 28.9% | 15.5 % |
| waveform | 21.4 % | 13.2 % |
| wdbc | 5.6% | 3.0 % |
| wine | 6.7% | 2.3% |

# Κεφάλαιο 5

# Συμπεράσματα και Επεκτάσεις

Κλείνοντας θα ανακεφαλαιώσουμε τα περιεχόμενα της διατριβής που παρουσιάσαμε και θα αναφέρουμε ορισμένα ανοιχτά θέματα καθώς και κάποιες νέες ιδέες που προέκυψαν από τη μελέτη του αλγορίθμου.

## 5.1 Συμπεράσματα

Σε αυτήν την εργασία παρουσιάσαμε έναν αυξητικό αλγόριθμο εκπαίδευσης νευρωνικών δικτύων ταξινόμησης. Είδαμε τις ιδιότητες που έχει η προβολή των δεδομένων σε κυρτές καμπύλες μιας παραπάνω διάστασης και εκμεταλλευτήκαμε την τοπικότητα των νευρώνων σε αυτή την περίπτωση για να διατυπώσουμε έναν αλγόριθμο που χρησιμοποιεί σημεία από το σύνολο εκπαίδευσης για να προσθέσει νευρώνες στο δίκτυο.

Αντίθετα με τις προσεγγίσεις που βασίζονται στην ελαχιστοποίηση του σφάλματος εκπαίδευσης, ο αλγόριθμός μας δεν απαιτεί τον καθορισμό τιμών παραμέτρων από την πλευρά του χρήστη και καθορίζει μόνος του τον αριθμό των νευρώνων που απαιτούνται. Κάτι τέτοιο αυτόματα δίνει ένα άνω φράγμα στον αριθμό των νευρώνων και μπορεί να χρησιμοποιηθεί ως ενδεικτικό της δυσκολίας του προβλήματος ή να βοηθήσει το χρήστη να ορίσει το πλήθος των νευρώνων και να εκπαιδεύσει το δίκτυο με συμβατικές μεθόδους. Ένα επίσης πλεονέκτημα του αυτόματου καθορισμού της αρχιτεκτονικής ενός δικτύου σε σχέση με τα παραδοσιακά MLP είναι ότι, για συγκεκριμένη αρχιτεκτονική ενός MLP, απαιτούνται αρκετές δοκιμές, λόγω σημαντικής εξάρτησης από την αρχικοποίηση της μεθόδου βελτιστοποίησης. Ο αλγόριθμός μας μπορεί ακόμα να χρησιμοποιηθεί, όπως είδαμε, για να ανιχνεύσουμε την παρουσία θορύβου (επικάλυψη) στα δεδομένα.

Οι επιδόσεις του αλγορίθμου μας, όπως τις παρουσιάσαμε στην ενότητα 4.3, είναι σε γενικές γραμμές ικανοποιητικές, ενώ σε αρκετές περιπτώσεις τα αποτελέσματα είναι συγκρίσιμα με τις επιδόσεις των SVMs, που θεωρούνται κορυφαίες μέθοδοι ταξινόμησης.

## 5.2 Επεκτάσεις

Στην περιγραφή του αλγορίθμου είδαμε πως η επιλογή της διάστασης στην οποία θα κινηθούμε γίνεται τυχαία, ενώ η αλλαγή των σημείων ελέγχου εξαντλητικά. Ίσως να υπάρχουν τρόποι με τους οποίους μπορούμε να κάνουμε πιο έξυπνα αυτές τις επιλογές.

Είδαμε επίσης κατά τα πειράματα για να αξιολογήσουμε τις επιδόσεις του αλγορίθμου ότι, ενώ βρίσκει πολύ καλές λύσεις, μερικές φορές επιλέγει με βάση το validation αρκετά χειρότερες. Αυτό έρχεται σε αντίφαση με τη διάσθηση ότι αν κάτι μια λύση έχει καλή επίδοση στο validation θα έχει καλή επίδοση και στο test, όπου βασίζεται και ο τρόπος επιλογής λύσης χρησιμοποιώντας άλλες μεθόδους. Υπάρχει ένα διαφορετικό κριτήριο επιλογής με το οποίο θα μπορούμε να επιλέγουμε τις καλές λύσεις;

Στην προσπάθειά μας να εκπαιδεύσουμε νευρωνικά δίκτυα για δεδομένα που έχουν προβληθεί στο παραβολοειδές χρησιμοποιώντας αριθμητικές μεθόδους διαπιστώσαμε ότι αυτές οι μέθοδοι εξαρτώνται κατά πολύ από την τυχαία αρχικοποίηση. Θα μπορούσαμε να χρησιμοποιήσουμε τον αλγόριθμό μας σε ενδεχομένως μικρότερο αριθμό δεδομένων για να αρχικοποιήσουμε το δίκτυο και στη συνέχεια να εκπαιδεύσουμε το δίκτυο χρησιμοποιώντας όλα τα δεδομένα.

Μέχρι τώρα μελετήσαμε τη συμπεριφορά που έχουν γραμμικοί νευρώνες σε δεδομένα που έχουν προβληθεί σε μια κυρτή επιφάνεια μιας παραπάνω διάστασεις. Η προβολή που κάνουμε μας περιορίζει ως προς τη μορφή που θα έχουν οι καμπύλες στον αρχικό χώρο. Μια εναλλακτική προσέγγιση που αξίζει να μελετηθεί είναι να λύνουμε το πρόβλημα σε μια παραπάνω διάσταση χωρίς προβάλουμε τα δεδομένα σε μια κυρτή επιφάνεια ($x_{d+1} = 0$) και να χρησιμοποιούμε παραβολοειδείς νευρώνες οι οποίοι με τις κατάλληλες παραμέτρους έχουν μεγαλύτερη ευελιξία ως προς το σχήμα της διαχωριστικής επιφάνειας που υλοποιούν.

# Βιβλιογραφία


[1] H. Edelsbrunner and R. Seidel, Voronoi diagrams and arrangements, *Disc. and Comp. Geometry* **1**, pp. 25–44, 1986.

[2] S. M. Omohundro, Geometric learning algorithms, *Physica D* **42**, pp. 307–321, 1990.

[3] S. Ridella, S. Rovetta and R. Zunino, Circular bàckpropagation networks for classification, *IEEE Transaction on Neural Networks* **8**(1), pp. 84–97, 1997.

[4] T. Cover, Geometrical and statistical properties of inequalities with application in pattern recognition *IEEE Trans. Electron. Comput.* **14**, pp. 326–334, 1965.

[5] V. N. Vapnik and A. J. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, *Theor. Prob. Applicat.* **16**, pp. 264–280, 1971.

[6] Joseph O'Rourke, Computational Geometry in C, *Cambridge university press*, 1st Edition, 1993.

[7] Christopher M. Bishop, Neural Networks for Pattern Recognition, *Clarendon press - Oxford*, 1st Edition, 1995.

[8] Constantinos Constantinopoulos and Aristidis Likas, An incremental approach to hierarchical training of classification mixture models, *Technical Report no 13 - 5 / 2005* Department of Computer Science, University of Ioannina, 2005.

[9] `http://vv.carleton.ca/~neil/neural/neuron-a.html`