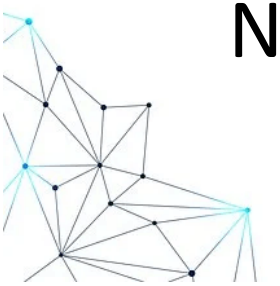
Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
ΔΠΜΣ «Επιστήμη Δεδομένων και Μηχανική Μάθηση»

Μηχανική Μάθηση

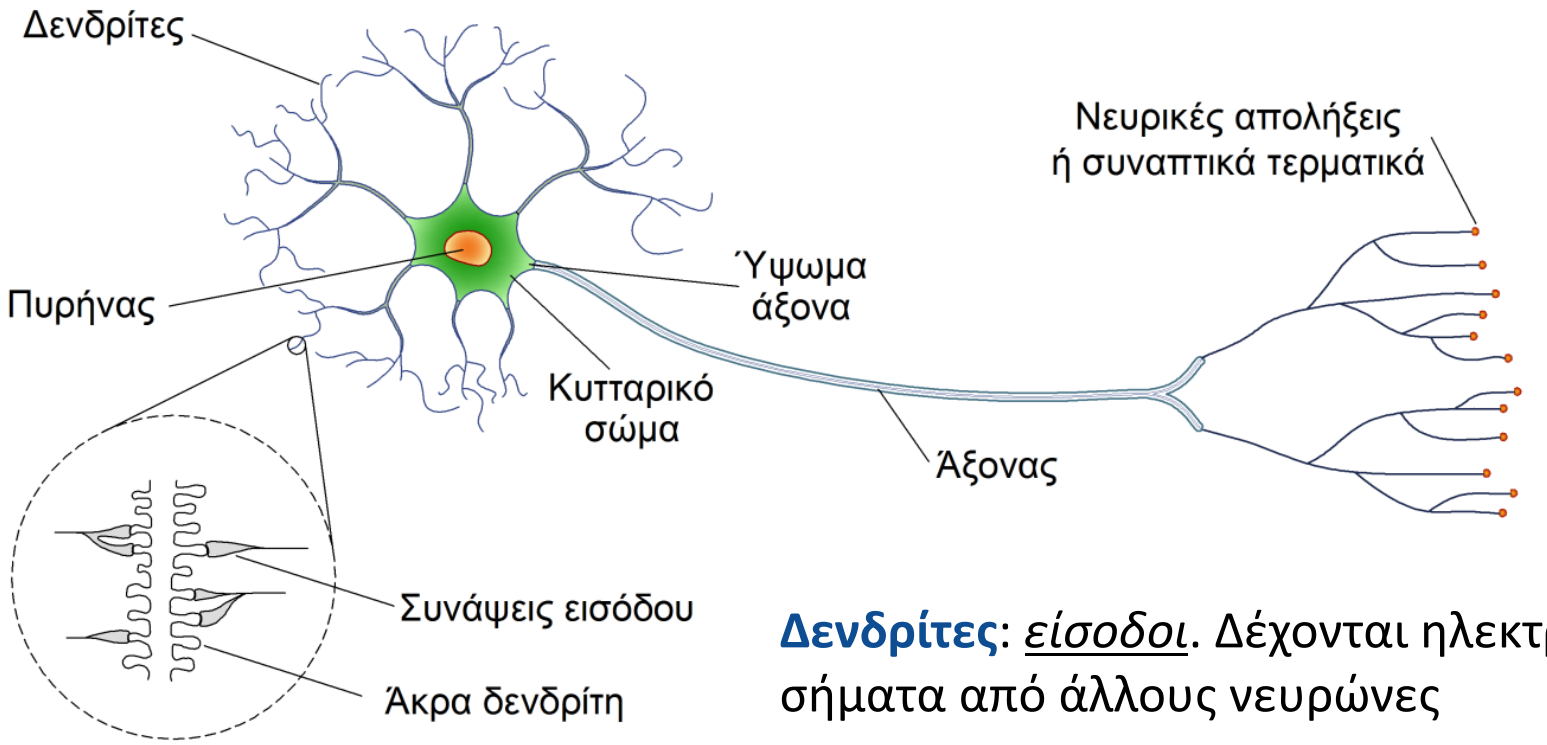
1ο εξάμηνο | ακαδημαϊκό έτος 2022-2023

Θάνος Βουλόδημος
Επ. Καθηγητής ΣΗΜΜΥ ΕΜΠ



Νευρωνικά Δίκτυα – Γραμμικά Μοντέλα – Perceptron – Multi Layer Perceptron

Πηγή έμπνευσης: Ο βιολογικός νευρώνας



Δενδρίτες: *είσοδοι*. Δέχονται ηλεκτρικά σήματα από άλλους νευρώνες

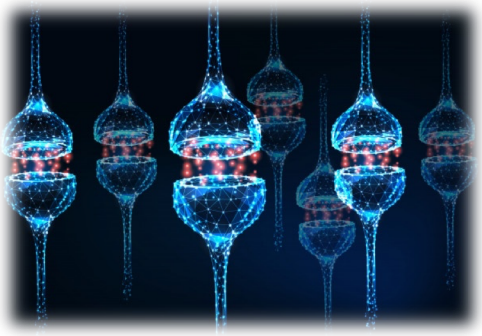
Άξονας: *έξοδος*. Μήκος από μερικά χιλιοστά έως >1m. Στέλνει ηλεκτρικούς παλμούς σταθερού πλάτους αλλά μεταβλητής συχνότητας.

Συνάψεις: σημεία ένωσης μεταξύ διακλαδώσεων του άξονα ενός νευρώνα και των δενδριτών από άλλους νευρώνες.

Συνάψεις

Είδη συνάψεων

- ενισχυτικές (exitatory)
- ανασταλτικές (inhibitory)



Φούσκες με ιόντα (Na^+, K^+). Το *πλάτος* της σύναψης, η *απόστασή* της από τον δενδρίτη και η *πυκνότητα* του ηλεκτροχημικού υλικού επηρεάζουν την ευκολία με την οποία η ηλεκτρική δραστηριότητα μεταδίδεται από τον άξονα στο δενδρίτη. Το ποσοστό της ηλεκτρικής δραστηριότητας που μεταδίδεται τελικά στο δενδρίτη λέγεται ***συναπτικό βάρος***.

Λειτουργία βιολογικού νευρώνα

- Συχνότητα παλμών στον άξονα (έξοδο) = ανάλογη της συνολικής διέγερσης
- Συνολική διέγερση = άθροισμα των διεγέρσεων σε όλους τους δενδρίτες

Όμως:

$$\textit{Συχνότητα παλμών} < 1 / (t_p + t_r)$$

- **Πλαστικότητα**: οι νευρώνες έχουν ρυθμιζόμενες (μεταβαλλόμενες) συνάψεις
- **Πολύ μεγάλο πλήθος** νευρώνων: 100 δισεκατομμύρια κατά μέσο όρο στον ανθρώπινο εγκέφαλο
  - **παραλληλισμός** της επεξεργασίας
  - **κατανομή** της πληροφορίας

Το μοντέλο McCulloch-Pitts

Threshold Logic Unit (Warren McCulloch, Walter Pitts, 1943)

- Μοντελοποίηση νευρώνα όπως περίπου τα τρανζίστορ (on/off):
  - $y = 0 \rightarrow$ ο νευρώνας είναι αδρανής
  - $y = 1 \rightarrow$ μέγιστη συχνότητα παλμών

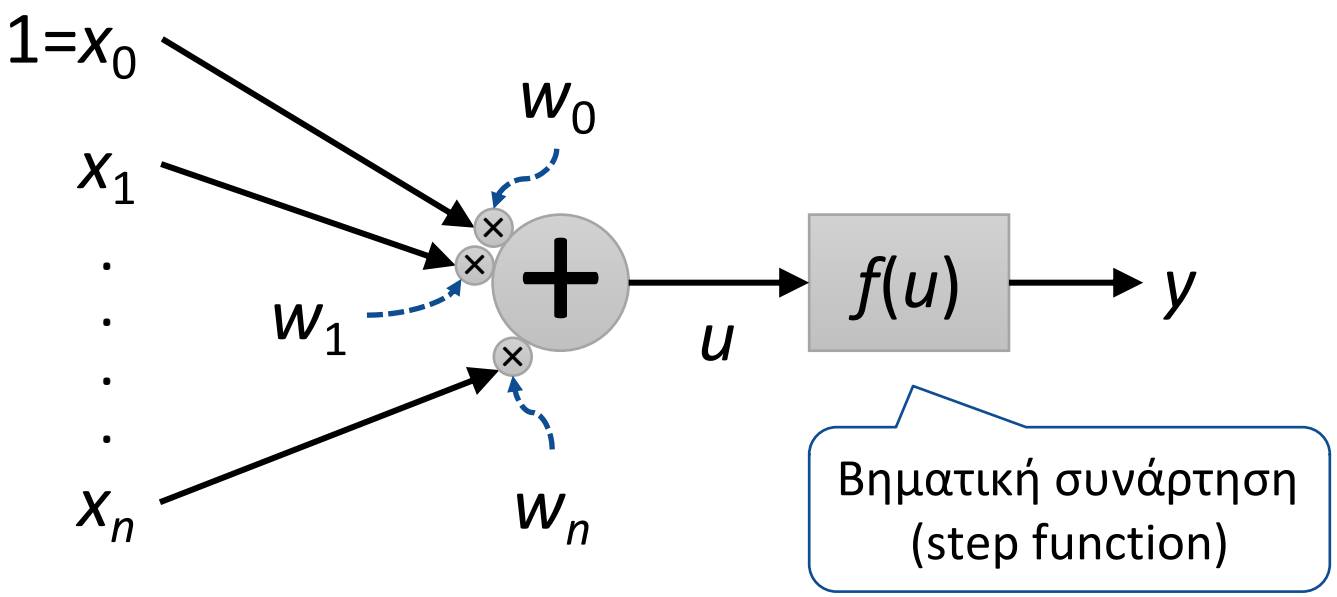
Δύο δυνατές καταστάσεις εξόδου

- Είσοδοι: $x_1, x_2, \cdots, x_n$
- Συναπτικά βάρη: $w_1, w_2, \cdots, w_n$
- Πόλωση: w_0
- Συνολική διέγερση: $u = w_1 x_1 + \cdots + w_n x_n + w_0$

Βηματική συνάρτηση (step function)

$$y = f(u) = \begin{cases} 0 & \text{αν } u < 0 \\ 1 & \text{αν } u > 0 \end{cases}$$

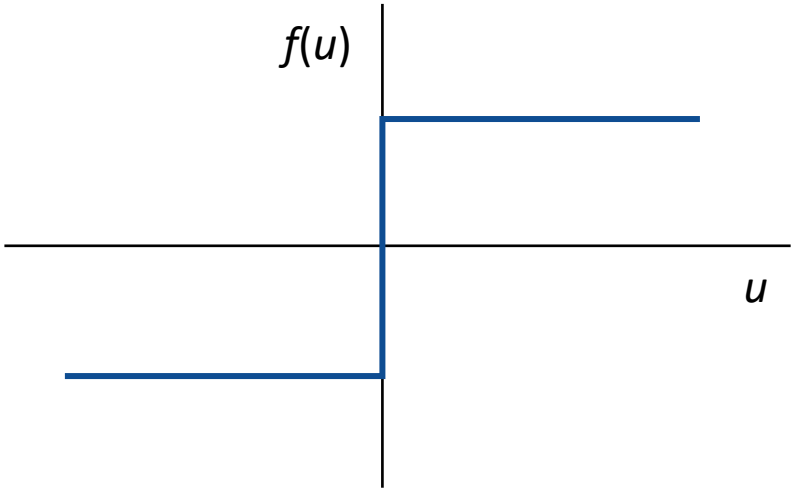
Το μοντέλο McCulloch-Pitts (2)

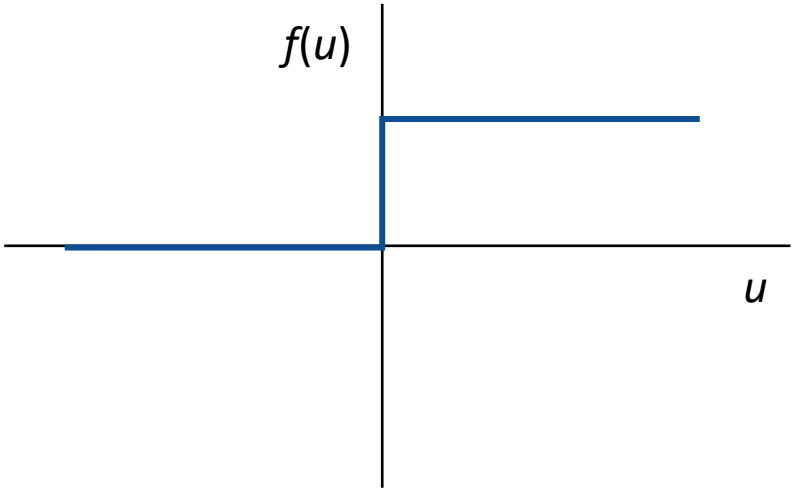


Το μοντέλο Perceptron είναι ουσιαστικά ένας νευρώνας McCulloch-Pitts

Συναρτήσεις ενεργοποίησης

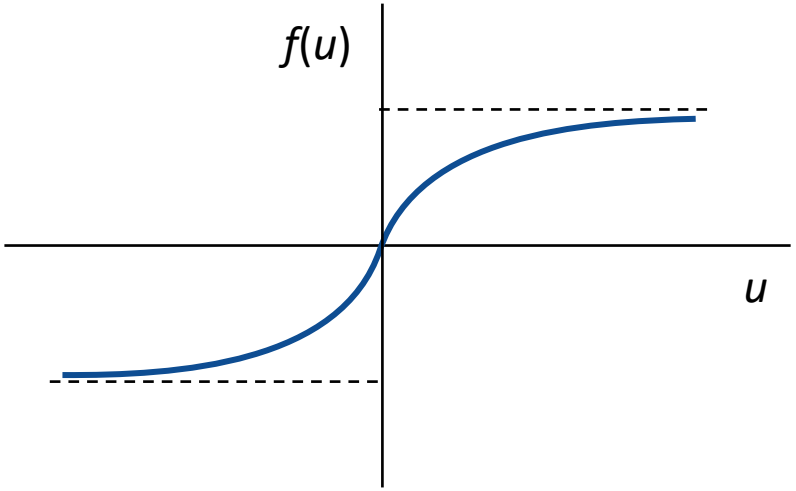
Βηματική (διπολική – sgn) -1/1	Βηματική (κλασική) 0/1
$f(u) = \begin{cases} -1, & \text{αν } u < 0 \\ 1, & \text{αν } u > 0 \end{cases}$	$f(u) = \begin{cases} 0, & \text{αν } u < 0 \\ 1, & \text{αν } u > 0 \end{cases}$

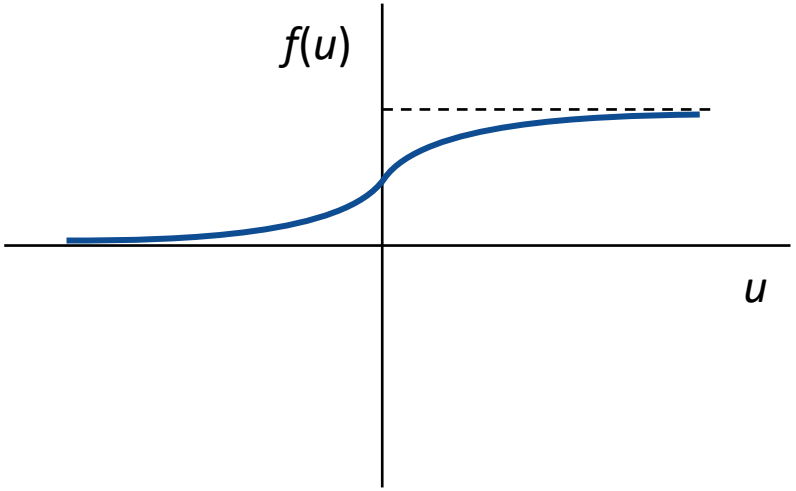




Συναρτήσεις ενεργοποίησης (2)

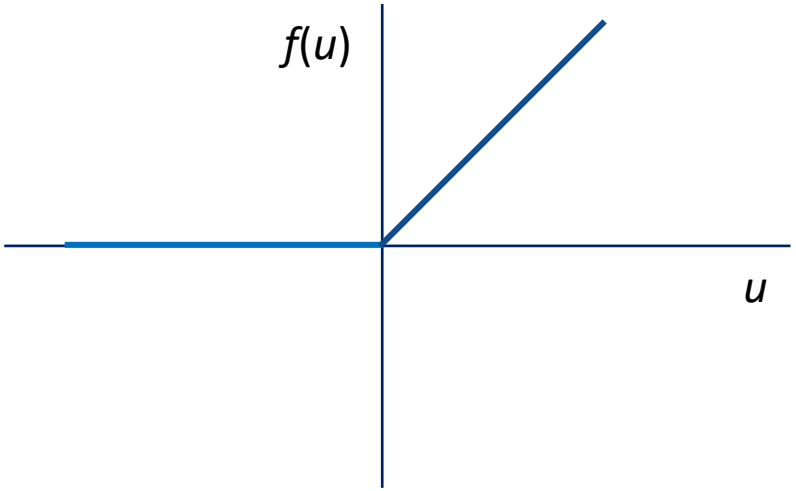
Υπερβολική εφαπτομένη	Σιγμοειδής
$f(u) = \tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}$	$f(u) = \frac{1}{1 + e^{-u}}$

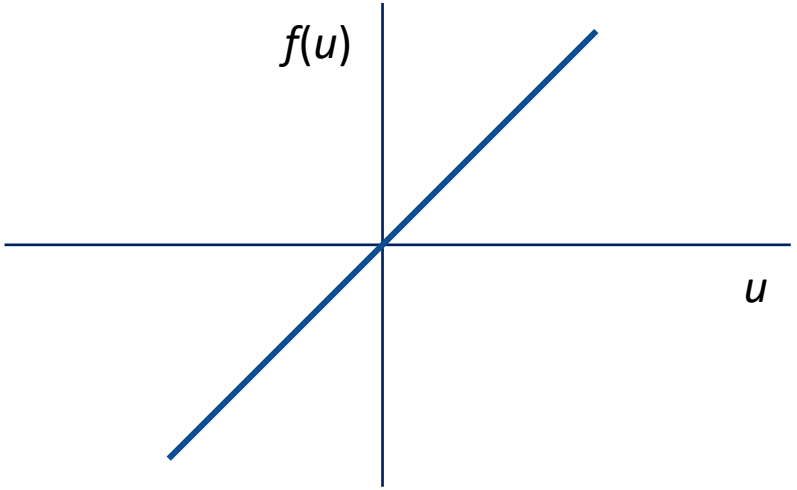




Συναρτήσεις ενεργοποίησης (3)

Ράμπα (ReLU = Rectifier Linear Unit)	**Γραμμική**
$f(u) = \begin{cases} 0, & \text{αν } u < 0 \\ u, & \text{αν } u > 0 \end{cases}$	$f(u) = u$



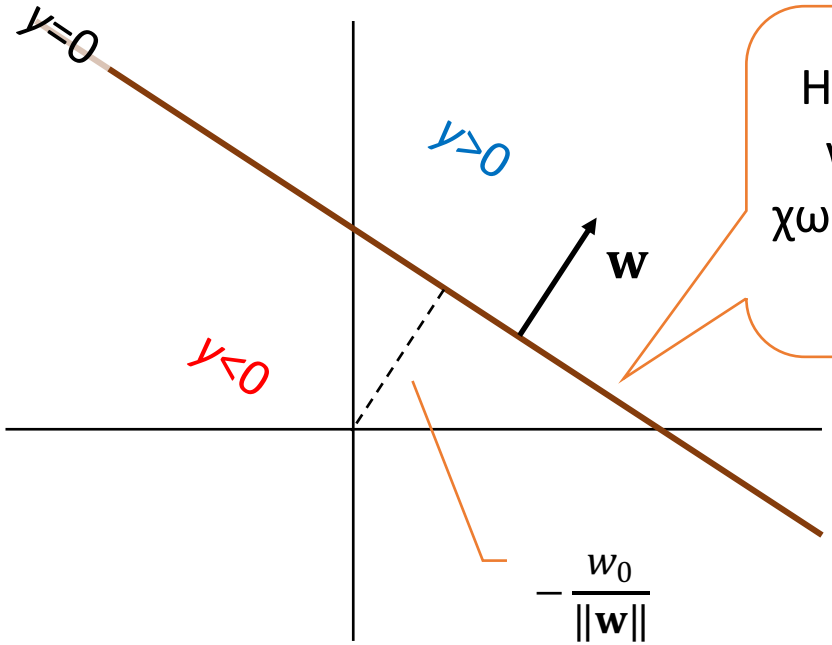


Γραμμική Συνάρτηση Διαχωρισμού

Ειδική περίπτωση όπου η $y=f(\mathbf{x};\mathbf{w})$ είναι της μορφής :

$$y=\mathbf{w}^T\mathbf{x}+w_0 \quad (=w_1x_1+w_2x_2+w_0)$$

w: διάνυσμα βαρών (weight vector), w_0: πόλωση (bias)



Η f δημιουργεί μια διαχωριστική γραμμή κάθετη στο **w** η οποία χωρίζει το επίπεδο σε θετικό μέρος και σε αρνητικό μέρος

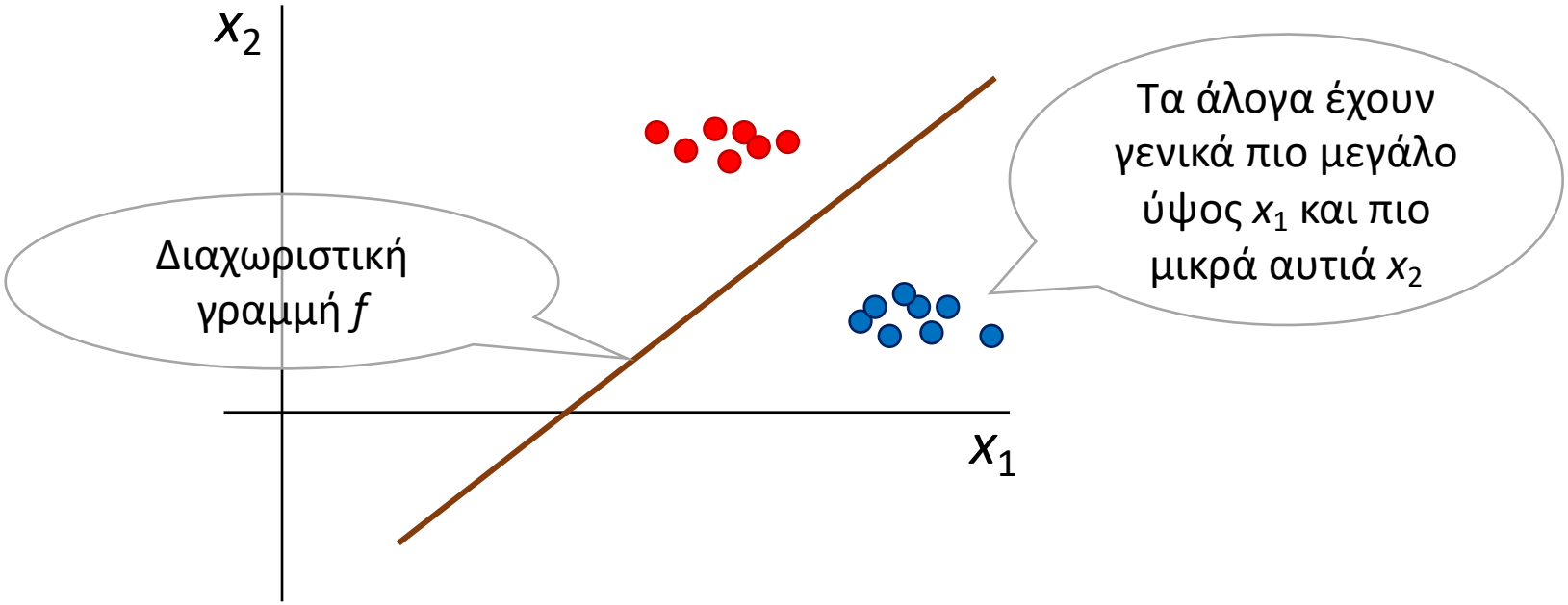
Αν $y>0$: Το πρότυπο ταξινομείται στην C_1

Αν $y<0$: Το πρότυπο ταξινομείται στην C_0

Χρήση Συνάρτησης Διαχωρισμού

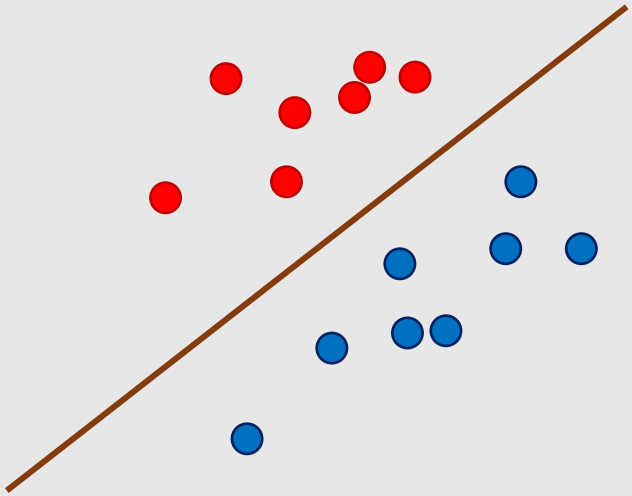
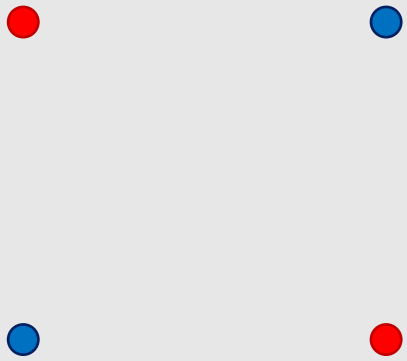
C_0: Γαϊδούρια ● C_1: Άλογα ●

x_1 = ύψος ζώου, x_2 = μήκος αυτιών



Γραμμική Διαχωρισιμότητα

- Linear Separability: Υπάρχει γραμμική επιφάνεια (πχ. ευθεία 2-d, επίπεδο 3-d) που χωρίζει τις δύο κλάσεις;

Γραμμικά Διαχωρίσιμο Πρόβλημα Linearly Separable	Μη Γραμμικά Διαχωρίσιμο Πρόβλημα Non-linearly Separable
	

Το Πρόβλημα της Ταξινόμησης (3)

- **Πολλαπλές Κλάσεις**: Δίνεται ένα πρότυπο εισόδου $\mathbf{x} \in \mathbb{R}^n$
  - Ζητείται να ταξινομηθεί σε μια από L κλάσεις $C_0, \cdots, C_{L-1}$
  - Ετικέτες: $\boldsymbol{t} = [1,0,\cdots,0]$ για C_0, $\boldsymbol{t} = [0,1,\cdots,0]$ για $C_1, \ldots, \boldsymbol{t} = [0,0,\cdots,1]$ για C_{L-1}

- **Παράδειγμα**. Αναγνώριση χειρόγραφων ψηφίων
  - $\boldsymbol{x}$=  $\boldsymbol{t}$=[0,0,**1**,0,0,0,0,0,0,0] (κλάση C_2)
  - $\boldsymbol{x}$=  $\boldsymbol{t}$=[0,0,0,0,0,0,0,0,**1**,0] (κλάση C_8)

Δίνονται εικόνες υπό μορφή διανυσμάτων από pixels

Πολλαπλές Κλάσεις (One vs. All)

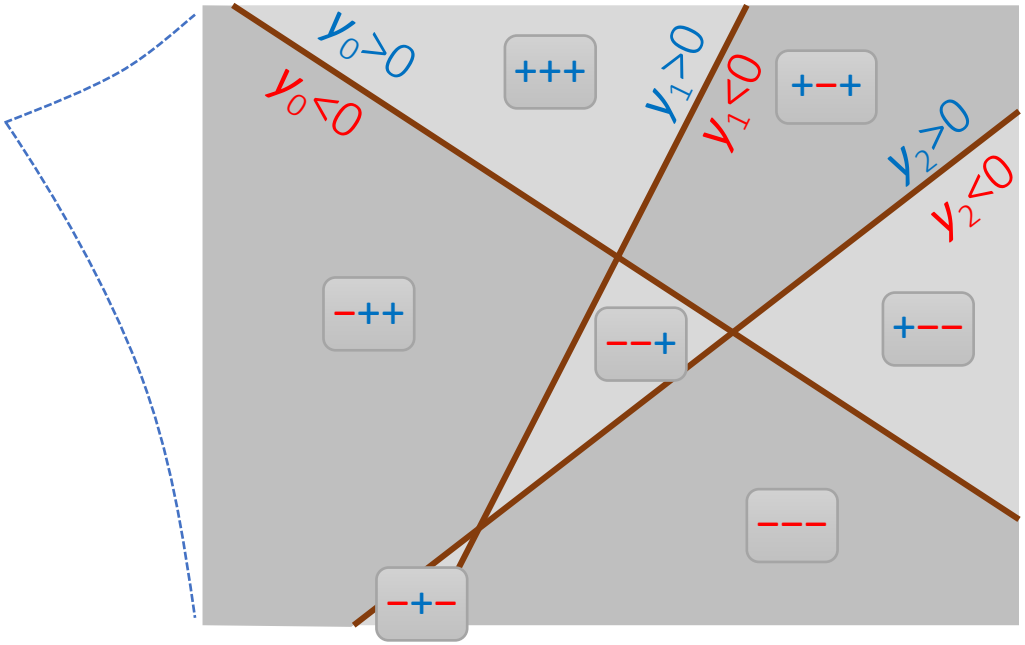
- Προσέγγιση Ένας ενάντια σε όλους (One vs All ή One vs Rest): Για κάθε κλάση C_i δημιουργούμε και μια διαφορετική συνάρτηση διαχωρισμού $y_i = \mathbf{w}_i^T \mathbf{x} + w_{i,0}$, ώστε να διακρίνει τα πρότυπα της κλάσης αυτής από όλες τις άλλες κλάσεις.

- **Παράδειγμα**.

  3 κλάσεις: C_0, C_1, C_2

- *Πρόβλημα*:

Δημιουργούνται τμήματα χωρίς σαφή «νικητή»



Γραμμική Ταξινόμηση Δύο Κλάσεων

Ελάχιστα Τετράγωνα, Perceptron

Λύση Ελαχίστων Τετραγώνων

Ας υποθέσουμε ότι οι στόχοι είναι $t_i = -1$, ή 1.

Μπορούμε να δούμε το πρόβλημα σαν ένα σύστημα εξισώσεων :

$$\mathbf{w}^T\mathbf{x}_1 + w_0 = t_1$$
$$\dots$$
$$\mathbf{w}^T\mathbf{x}_P + w_0 = t_P$$

ή πιο απλά

$$\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_1 = t_1$$
$$\dots$$
$$\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_P = t_P$$

όπου $\tilde{\mathbf{w}}^T = [\mathbf{w}^T,\ w_0]$ και $\tilde{\mathbf{x}}_i = \begin{bmatrix}\mathbf{x}_i \\ 1\end{bmatrix}$

Λύση Ελαχίστων Τετραγώνων (2)

Υπό μορφή Πίνακα :

$$\tilde{\mathbf{X}}\tilde{\mathbf{w}} = \mathbf{t}$$

όπου $\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{\mathbf{x}}_1^T \\ \vdots \\ \tilde{\mathbf{x}}_P^T \end{bmatrix}$, $\mathbf{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_P \end{bmatrix}$.

Ωστόσο, υπάρχει το εξής πρόβλημα: το πλήθος των **εξισώσεων** *P* (πλήθος γραμμών του πίνακα $\tilde{\mathbf{X}}$) είναι συνήθως **μεγαλύτερο** από τη διάσταση του $\tilde{\mathbf{w}}$ (πλήθος στηλών του πίνακα $\tilde{\mathbf{X}}$).

Αν η διάσταση του **w** είναι *n* (ισούται με τη διάσταση των προτύπων), τότε η διάσταση του $\tilde{\mathbf{w}}$ είναι (*n*+1).

Συνήθως $P > n + 1$.

Λύση Ελαχίστων Τετραγώνων (3)

Στην περίπτωση αυτή, το σύστημα $\tilde{\mathbf{x}}_i^T \tilde{\mathbf{w}} = t_i$ έχει περισσότερες εξισώσεις από αγνώστους (**υπερορισμένο**) και άρα δεν έχει λύση.

Προφανώς, όποιο διάνυσμα $\tilde{\mathbf{w}}$ και αν επιλέξουμε, θα έχουμε ένα σφάλμα ε_i για κάθε πρότυπο $\tilde{\mathbf{x}}_i$:

$$\tilde{\mathbf{x}}_i^T \tilde{\mathbf{w}} = t_i + \varepsilon_i$$

$$\varepsilon_i = \tilde{\mathbf{x}}_i^T \tilde{\mathbf{w}} - t_i$$

- Λύση ελαχίστων τετραγώνων: Θα προσπαθήσουμε να βρούμε εκείνο το $\tilde{\mathbf{w}}$ που θα **ελαχιστοποιεί** το άθροισμα των τετραγώνων των ε_i, δηλαδή θα προσπαθήσουμε να ελαχιστοποιήσουμε το κόστος :

$$J_{LS} = \sum_{i=1}^{P} \varepsilon_i^2 = \sum_{i=1}^{P} \left(\tilde{\mathbf{x}}_i^T \tilde{\mathbf{w}} - t_i\right)^2 = \|\tilde{\mathbf{X}}\tilde{\mathbf{w}} - \mathbf{t}\|^2$$

Λύση Ελαχίστων Τετραγώνων (4)

- Η λύση είναι γνωστή από τα μαθηματικά:

$$\tilde{\mathbf{w}} = \left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}^T\mathbf{t} \quad (\mathbf{X}\mathbf{w} = \mathbf{t}, \boldsymbol{X}^{-1}\mathbf{X}\mathbf{w} = \mathbf{I}\mathbf{w} = \boldsymbol{X}^{-1}\mathbf{t}, \mathbf{w} = \boldsymbol{X}^{-1}\mathbf{t})$$

- Ο πίνακας $\left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}^T$ λέγεται και ψευδο-αντίστροφος (pseudo-inverse) του $\tilde{\mathbf{X}}$ και συμβολίζεται με $\tilde{\mathbf{X}}^+$.

Ο $\tilde{\mathbf{X}}^+$ λέγεται ψευδο-αντίστροφος διότι :

$$\tilde{\mathbf{X}}^+\tilde{\mathbf{X}} = \left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} = \mathbf{I}$$

ακόμη και αν ο $\tilde{\mathbf{X}}$ δεν είναι τετραγωνικός πίνακας.

Βέβαια

$$\tilde{\mathbf{X}}\,\tilde{\mathbf{X}}^+ \neq \mathbf{I}$$

Συμπεράσματα

- Αν το πρόβλημα **είναι γραμμικά** διαχωρίσιμο:
  - Η μέθοδος των ελαχίστων τετραγώνων δίνει συχνά τη διαχωριστική γραμμή.
  - Κάποιες φορές όμως μπορεί να μην τη βρει.
- Αν το πρόβλημα **δεν είναι γραμμικά** διαχωρίσιμο:
  - Η μέθοδος θα δώσει κάποια ευθεία με καλά αποτελέσματα (συνήθως).

Το Μοντέλο Perceptron

- Το μοντέλο Perceptron είναι παρόμοιο με το γραμμικό μοντέλο που χρησιμοποιήθηκε στα ελάχιστα τετράγωνα:

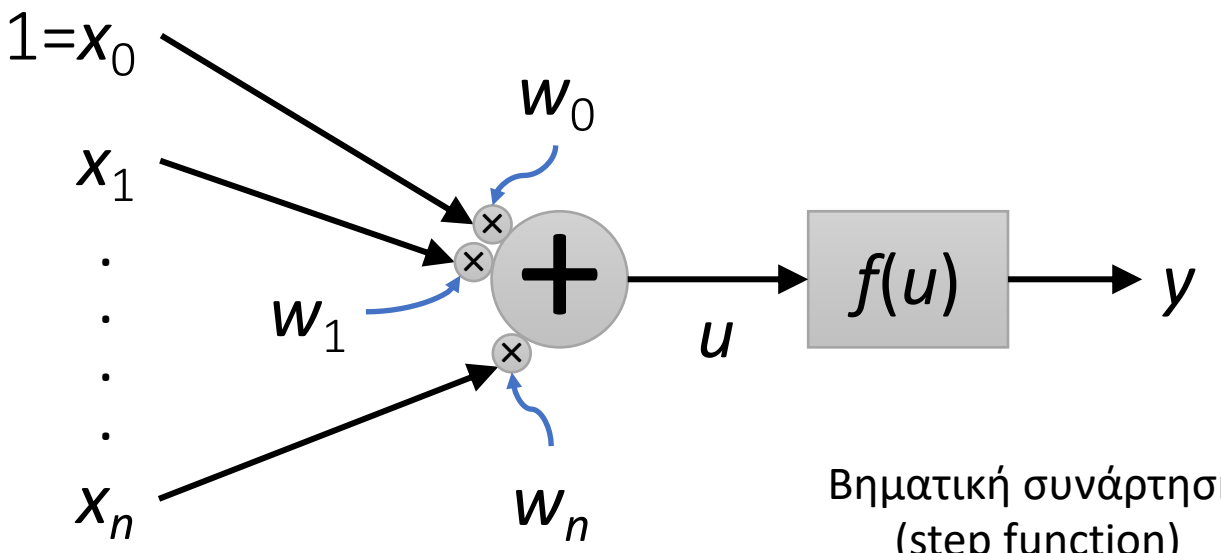
$$y_i = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i \qquad y_i = f(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i)$$

Γραμμικό μοντέλο ελαχίστων τετραγώνων

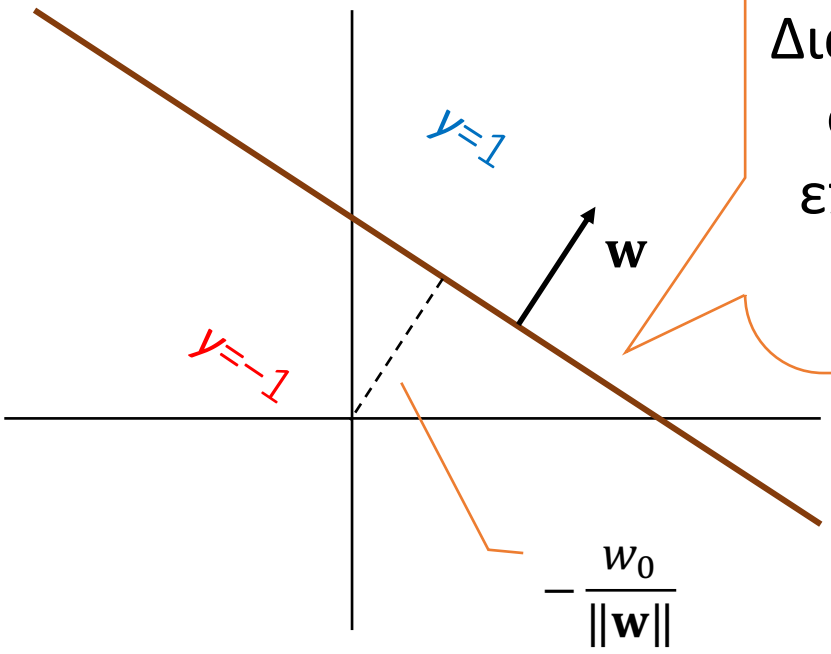
Μοντέλο Perceptron

- Η διαφορά είναι στη συνάρτηση f που παίρνει τιμές -1 ή 1:
- $f(u) = \begin{cases} -1, & \text{αν } u < 0 \\ 1, & \text{αν } u > 0 \end{cases}$
- Η έξοδος του μοντέλου Perceptron είναι -1/1, ενώ στο γραμμικό μοντέλο των ελαχίστων τετραγώνων η έξοδος ήταν πραγματικός αριθμός από $-\infty$ έως $+\infty$.

Σχεδιάγραμμα Perceptron



Δυνατότητες Perceptron



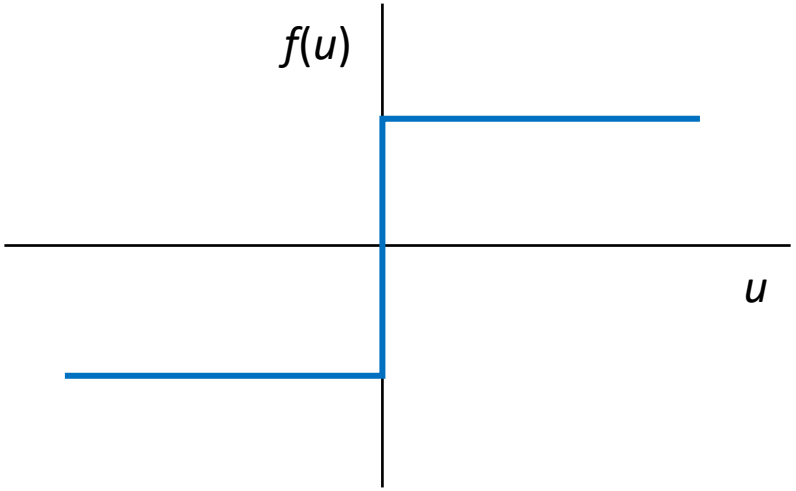
Διαχωριστική γραμμή κάθετη στο **w** η οποία χωρίζει το επίπεδο σε αρνητικό μέρος και σε θετικό μέρος

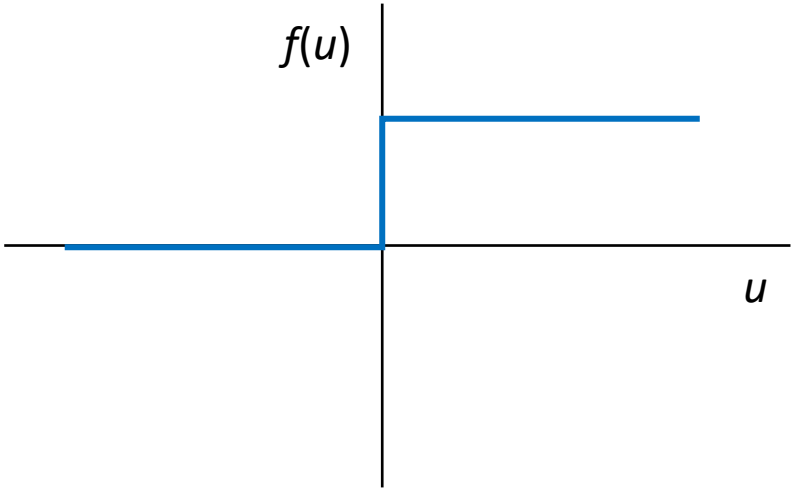
Αν y=1: Το πρότυπο ταξινομείται στην C_1

Αν y=−1: Το πρότυπο ταξινομείται στην C_0

Επιλογή Βηματικής Συνάρτησης

Βηματική -1/1 **Αν οι στόχοι είναι t=-1/1**	**Βηματική 0/1** **Αν οι στόχοι είναι t=0/1**
$f(u) = \begin{cases} -1, & \text{αν } u < 0 \\ 1, & \text{αν } u > 0 \end{cases}$	$f(u) = \begin{cases} 0, & \text{αν } u < 0 \\ 1, & \text{αν } u > 0 \end{cases}$





Εκπαίδευση Perceptron

Επαυξημένα Πρότυπα Εισόδου	$\tilde{\mathbf{x}}_1$	$\tilde{\mathbf{x}}_2$	…	$\tilde{\mathbf{x}}_P$
Στόχοι	t_1	t_2	…	t_P
Έξοδοι	y_1	y_2	…	y_P

Εποχή = μια κυκλική επανάληψη όλων των προτύπων.

Αρχικά **τυχαίο** $\tilde{\mathbf{w}}_0$, k=0

Για κάθε εποχή = 1:Maxepochs

  Για κάθε πρότυπο p = 1:P

    $k = k+1$

$$y_p = f\left(\tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}}_p\right)$$

$$\tilde{\mathbf{w}}_{k+1} = \tilde{\mathbf{w}}_k + \beta\left(t_p - y_p\right)\tilde{\mathbf{x}}_p$$

- *Διόρθωση μόνο αν στόχος $t_p \neq$ έξοδος y_p*

Τερματισμός Αλγορίθμου

- Ο αλγόριθμος τερματίζεται όταν δεν γίνεται πλέον καμία διόρθωση σε κανένα πρότυπο. Αυτό σημαίνει ότι **όλοι** οι στόχοι είναι **ίσοι** με **όλες** τις εξόδους:

$$\begin{aligned} & t_1 = y_1 \\ \text{ΚΑΙ} \quad & t_2 = y_2 \\ \text{ΚΑΙ} \quad & \dots \\ \text{ΚΑΙ} \quad & t_P = y_P \end{aligned}$$

- **Πρόβλημα**: Αν τα δεδομένα δεν είναι γραμμικά διαχωρίσιμα ο κανόνας Perceptron **δεν συγκλίνει** ποτέ. Αναγκαστικός ο πρόωρος τερματισμός.
- **Θεώρημα**: Αν τα δεδομένα είναι γραμμικά διαχωρίσιμα τότε ο κανόνας Perceptron **συγκλίνει** σε πεπερασμένο (αλλά άγνωστο) αριθμό επαναλήψεων.

Το Βήμα Εκπαίδευσης β

- Η παράμετρος β λέγεται **βήμα εκπαίδευσης** και είναι (συνήθως μικρή) θετική σταθερά.
- Η διόρθωση των βαρών είναι ανάλογη του β.
- Μεγάλο $\beta \rightarrow$ κίνδυνος ταλάντωσης.
- Μικρό $\beta \rightarrow$ αργή σύγκλιση.

Κανόνας ADALINE

Adaptive linear element

- Δεν χρησιμοποιείται η μη-γραμμική συνάρτηση ενεργοποίησης
  - Η έξοδος παίρνει συνεχείς τιμές
  - Οι στόχοι μπορούν να παίρνουν συνεχείς τιμές

$$\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(P)}]^T \qquad \mathbf{d} = [d^{(1)}, d^{(2)}, \ldots, d^{(P)}]^T$$

$$\mathbf{X}\mathbf{w} = \mathbf{d}$$

Επίλυση συστήματος P εξισώσεων με n+1 αγνώστους

Κανόνας ADALINE

Adaptive linear element

- Αν $P>n+1$ (συνήθης περίπτωση) το σύστημα μπορεί να μην έχει λύση
- Στην περίπτωση αυτή αναζητούμε προσεγγιστική λύση, χρησιμοποιώντας κριτήριο απόστασης από όλα τα πρότυπα
  - Ελάχιστο τετραγωνικό σφάλμα

$$J = \sum_{i=1}^{P} \left(d^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)} \right)^2$$

Ο αλγόριθμος τερματίζει αν το σφάλμα γίνει μικρότερο από ε

Αλγόριθμος εκπαίδευσης ADALINE

- ΒΗΜΑ 1: 1. Αρχικοποίησε το διάνυσμα βαρών τυχαίες τιμές

  2. Δώσε μία μικρή θετική τιμή στο βήμα εκπαίδευσης

  3. Όρισε ένα όριο *ε* για το σφάλμα εκπαίδευσης

  4. Όρισε το μέγιστο αριθμό εποχών

- ΒΗΜΑ 2: Επανάλαβε έως τον μέγιστο αριθμό εποχών

  Για κάθε ένα από τα *P* πρότυπα εκτέλεσε

  1. Υπολόγισε την έξοδο του perceptron

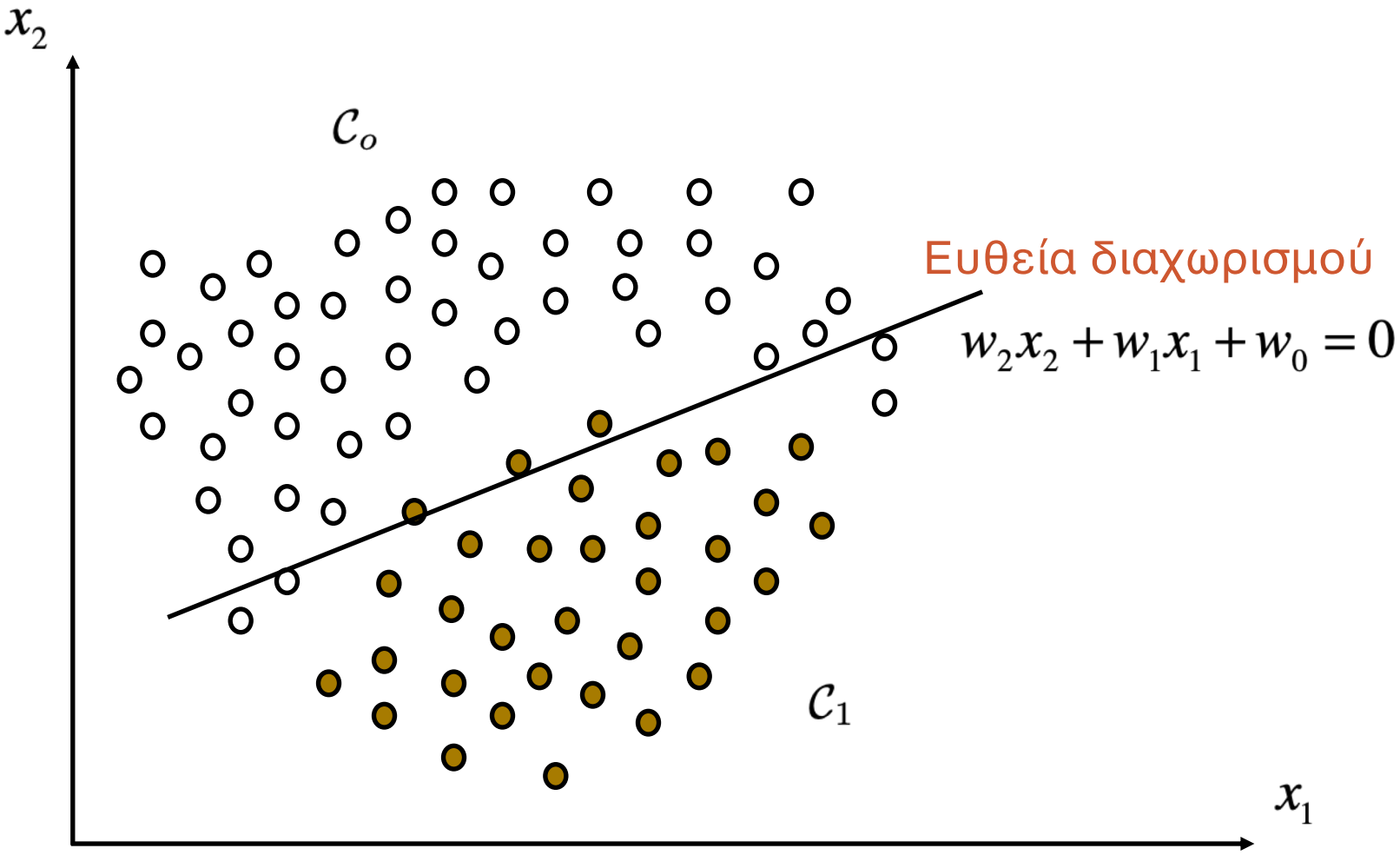
  2. Αν υπάρχει σφάλμα προσάρμοσε τα βάρη ως:

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \beta(d-y)\mathbf{x}$$

  Επίστρεψε τα βάρη αν ισχύει ότι: $\sum_{i=1}^{P}\left(d^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\right)^2 < \varepsilon$

- ΒΗΜΑ 3: Επίστρεψε ΣΦΑΛΜΑ

Ταξινόμηση με ADALINE



Παρατηρήσεις

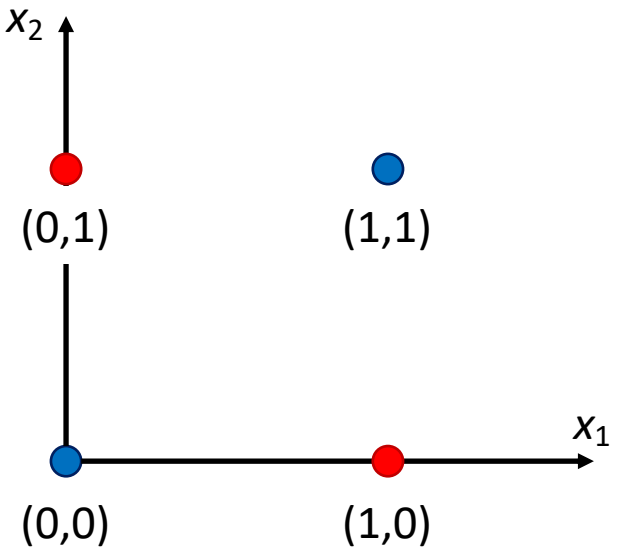
- Αν το πρόβλημα είναι γραμμικά διαχωρίσιμο, το perceptron βρίσκει πάντα λύση, σε αντίθεση με το adaline
- Αν το πρόβλημα δεν είναι γραμμικά διαχωρίσιμο, το perceptron δεν συγκλίνει, ενώ το adaline *μπορεί* να συγκλίνει επιτρέποντας λανθασμένες ταξινομήσεις
- Το adaline συγκλίνει σε περιπτώσεις μη-γραμμικά διαχωρίσιμων προβλημάτων *μόνο* όταν το πρόβλημα είναι *σχεδόν* γραμμικά διαχωρίσιμο (κάτω από το σφάλμα ε)

Multi Layer Perceptron

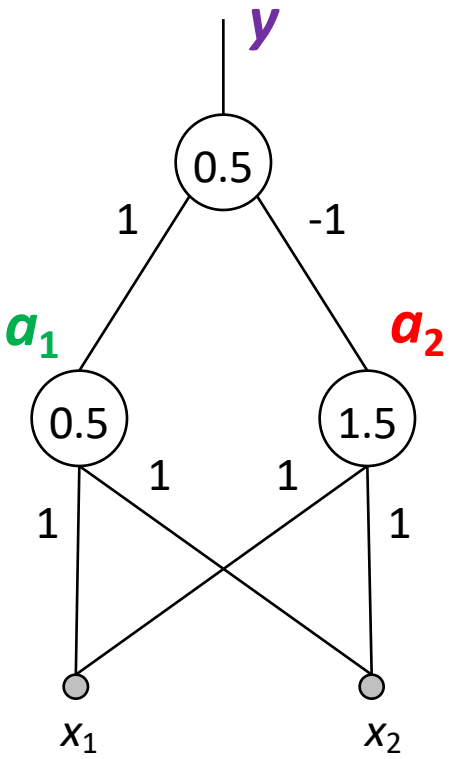
Το Πρόβλημα XOR

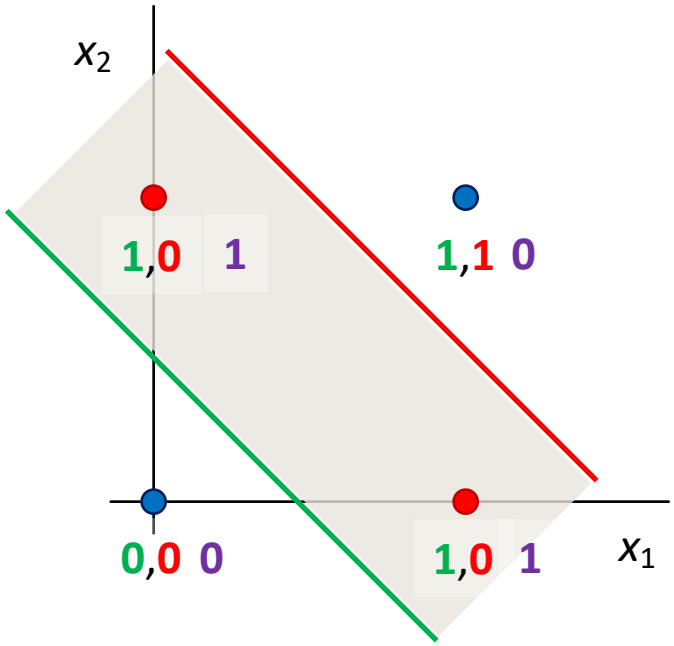
- Το μοντέλο Perceptron είναι ουσιαστικά ένας νευρώνας
- Έχει περιορισμένες δυνατότητες: Δημιουργεί μόνο γραμμικές διαχωριστικές επιφάνειες (πχ. ευθείες σε 2-D)
- Αδυναμία επίλυσης μη-γραμμικά διαχωρίσιμων προβλημάτων. Παράδειγμα, το πρόβλημα XOR:

x_1	x_2	t
0	0	0
0	1	1
1	0	1
1	1	0



Λύση του XOR με 2 στρώματα νευρώνων



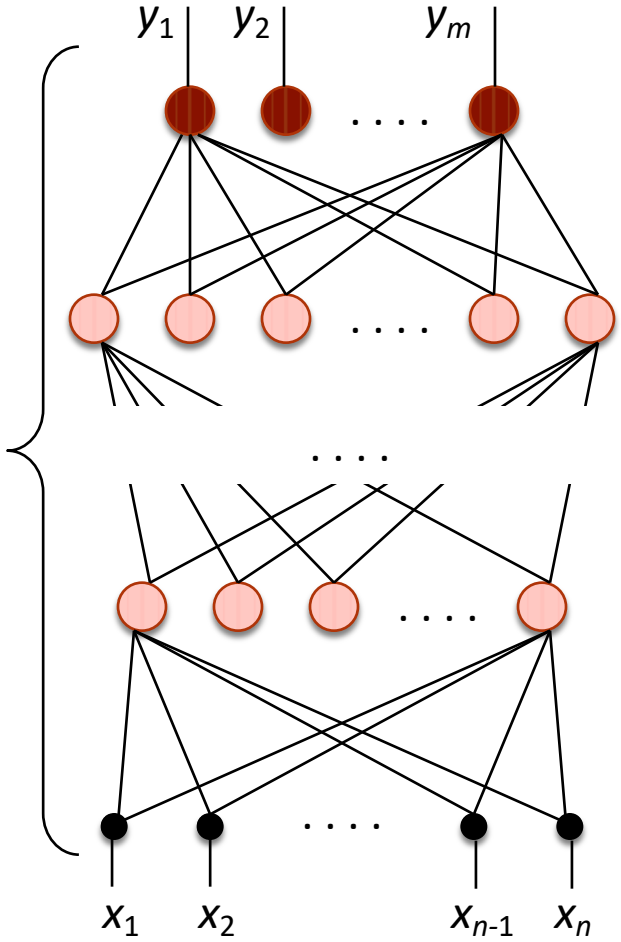


* Έστω για απλούστευση ότι η συνάρτηση ενεργοποίησης των κρυφών νευρώνων είναι η βηματική συνάρτηση 0/1

Multi-Layer Perceptron (MLP):

Πολλαπλοί νευρώνες σε στρώματα

Δίκτυο Perceptron Πολλαπλών Στρωμάτων
Multi-Layer Perceptron (MLP)



Στρώμα Εξόδου

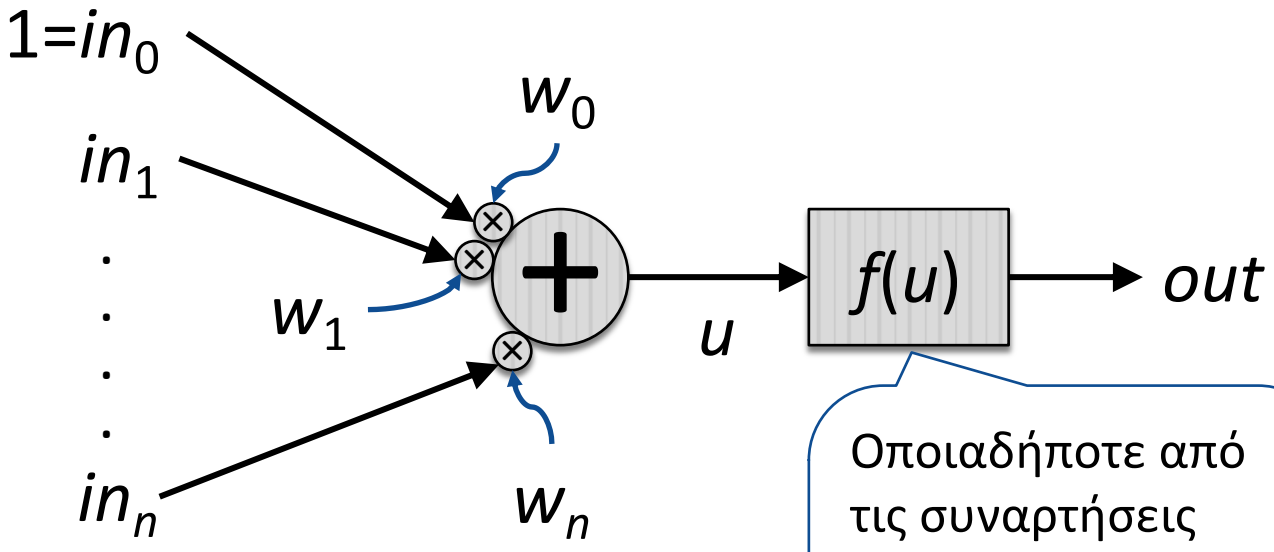
Κρυφό Στρώμα N

. . . .

Κρυφό Στρώμα 1

Στρώμα Εισόδου

Νευρώνες στο MLP



Οποιαδήποτε από τις συναρτήσεις

- Γραμμική
- Σιγμοειδής
- Tanh
- ReLU

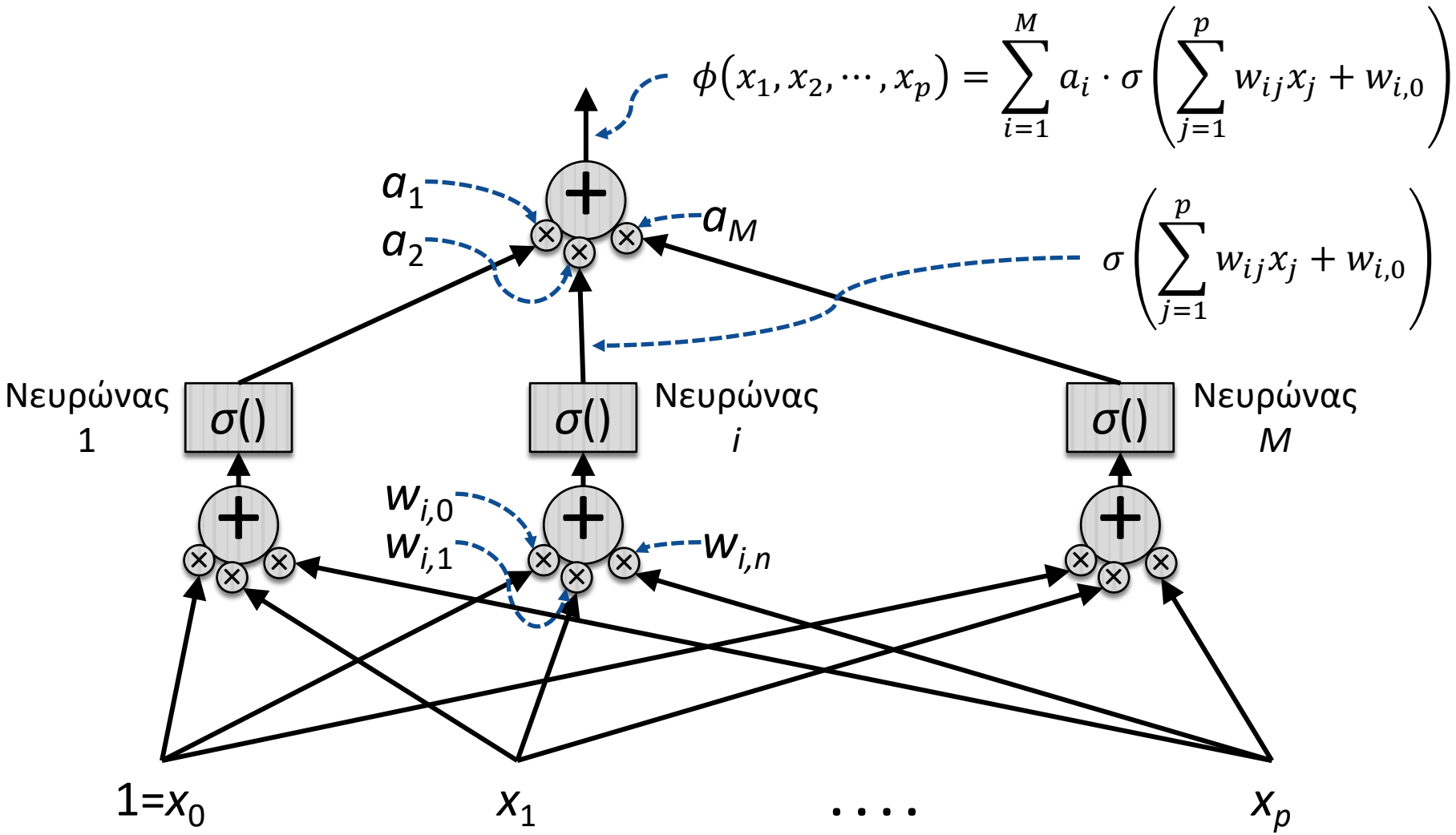
Δυνατότητες δικτύων MLP

Θεώρημα: Έστω $\sigma(\cdot)$= η σιγμοειδής συνάρτηση και $g(x_1, x_2, \cdots, x_p)$ οποιαδήποτε συνεχής συνάρτηση με *p* μεταβλητές $0 \leq x_i \leq 1$.
Τότε υπάρχει ακέραιος *M* και κάποιες τιμές των παραμέτρων a_i, w_{ij}, έτσι ώστε η συνάρτηση

$$\phi(x_1, x_2, \cdots, x_p) = \sum_{i=1}^{M} a_i \cdot \sigma\left(\sum_{j=1}^{p} w_{ij} x_j + w_{i,0}\right)$$

προσεγγίζει την $g(x_1, x_2, \cdots, x_p)$ με σφάλμα μικρότερο του ε για όλες τις τιμές $0 \leq x_1, x_2, \cdots, x_p \leq 1$ και για οποιοδήποτε $\varepsilon > 0$.

Ερμηνεία Θεωρήματος



Ερμηνεία Θεωρήματος (2)

- Ιδιότητα ***Καθολικού Προσεγγιστή*** *(Universal Approximator)*
- Με απλά λόγια: Ένα δίκτυο δύο στρωμάτων μπορεί να προσεγγίσει όσο καλά επιθυμούμε οποιαδήποτε συνεχή συνάρτηση, αρκεί
  - Να έχουμε αρκετούς κρυφούς νευρώνες *M*
  - Οι νευρώνες του κρυφού στρώματος να έχουν τη σιγμοειδή συνάρτηση ενεργοποίησης
  - Ο νευρώνας εξόδου να έχει τη γραμμική συνάρτηση ενεργοποίησης

Δυνατότητες Δικτύων MLP (2)

- Όσο πιο πολύπλοκη είναι η συνάρτηση που επιθυμούμε να προσεγγίσουμε τόσο περισσότερους κρυφούς νευρώνες θέλουμε
- Δύο στρώματα αρκούν (θεωρητικά)
- Universal Approximator: ισχύει και για συναρτήσεις πολλών εξόδων
- Γενικού σκοπού: Κατάλληλα και για ταξινόμηση και για παλινδρόμηση.

Ανάκληση στο MLP

Αλγόριθμος

Είσοδοι: $a_1(0) = x_1, \cdots, a_{N(0)}(0) = x_{N(0)}$

0 = στρώμα εισόδου, L = στρώμα εξόδου

Έξοδοι: $y_1 = a_1(L) =, \cdots, y_{N(L)} = a_{N(L)}(L)$

Για κάθε στρώμα $l = 1, \ldots, L$ {

Για κάθε νευρώνα $i = 1, \ldots, N(l)$ {

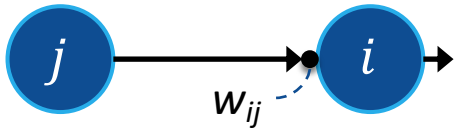
$$a_i(l) = \sigma\left(\sum_{j=1}^{N(l-1)} w_{ij}(l) a_j(l-1) + w_{i0}(l)\right)$$

}

}

Συμβολισμοί

- $N(l)$: πλήθος νευρώνων στο στρώμα l
- $a_i(l)$: έξοδος του νευρώνα i στο στρώμα l
- x_i : είσοδοι του δικτύου (στρώμα 0)
- y_i : έξοδοι του δικτύου (στρώμα L)
- w_{ij} : συναπτικό βάρος από τον νευρώνα j στον i



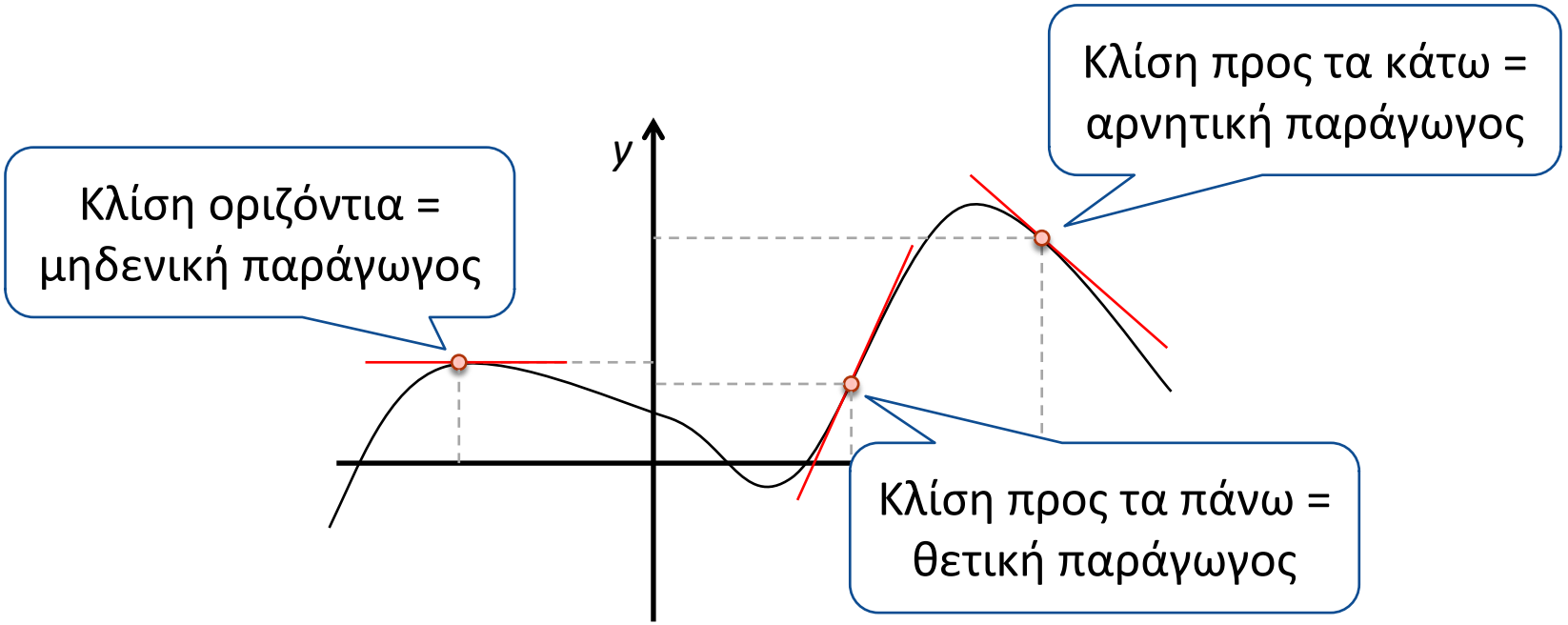
Κατάβαση δυναμικού (gradient descent) – αλγόριθμος back-propagation

Δύο βασικά προβλήματα Βελτιστοποίησης

- Ελαχιστοποίηση
  - Δίνεται: μια συνάρτηση $y = f(\mathbf{x})$ όπου $\mathbf{x} \in \mathbb{R}^n$, $y \in \mathbb{R}$
  - Ζητείται: να βρεθεί το σημείο $\mathbf{x}_*$ ώστε το $y_* = f(\mathbf{x}_*)$ να είναι το ελάχιστο δυνατό, δηλαδή, $y_* = f(\mathbf{x}_*) \leq f(\mathbf{x}), \forall \mathbf{x}$.
- Μεγιστοποίηση
  - Δίνεται: μια συνάρτηση $y = f(\mathbf{x})$ όπου $\mathbf{x} \in \mathbb{R}^n$, $y \in \mathbb{R}$
  - Ζητείται: να βρεθεί το σημείο $\mathbf{x}_*$ ώστε το $y_* = f(\mathbf{x}_*)$ να είναι το μέγιστο δυνατό, δηλαδή, $y_* = f(\mathbf{x}_*) \geq f(\mathbf{x}), \forall \mathbf{x}$.
- Δεν υπάρχει πάντα λύση στα προβλήματα αυτά.

Παράγωγοι

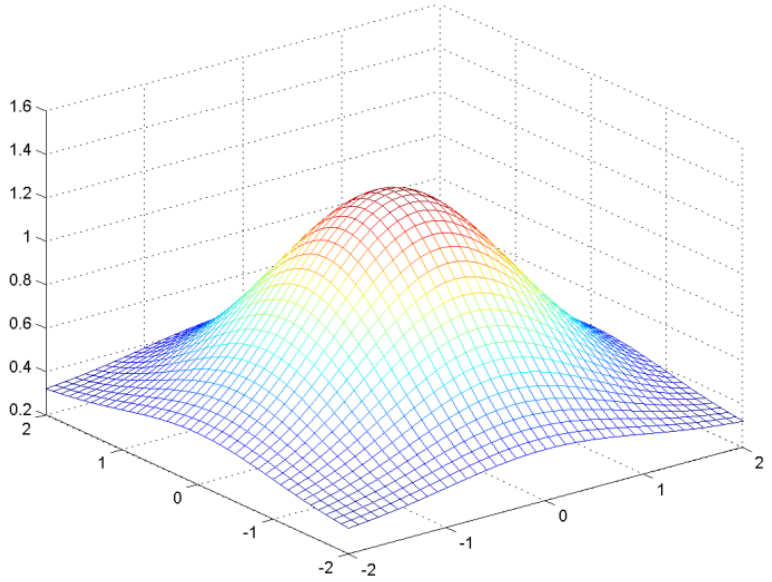
- Παράγωγος βαθμωτής συνάρτησης f με βαθμωτή μεταβλητή x, δηλ. $y = f(x), x, y \in \mathbb{R}$:
  - $\frac{df}{dx} = \lim_{\delta \to 0} \frac{f(x+\delta)-f(x)}{\delta}$

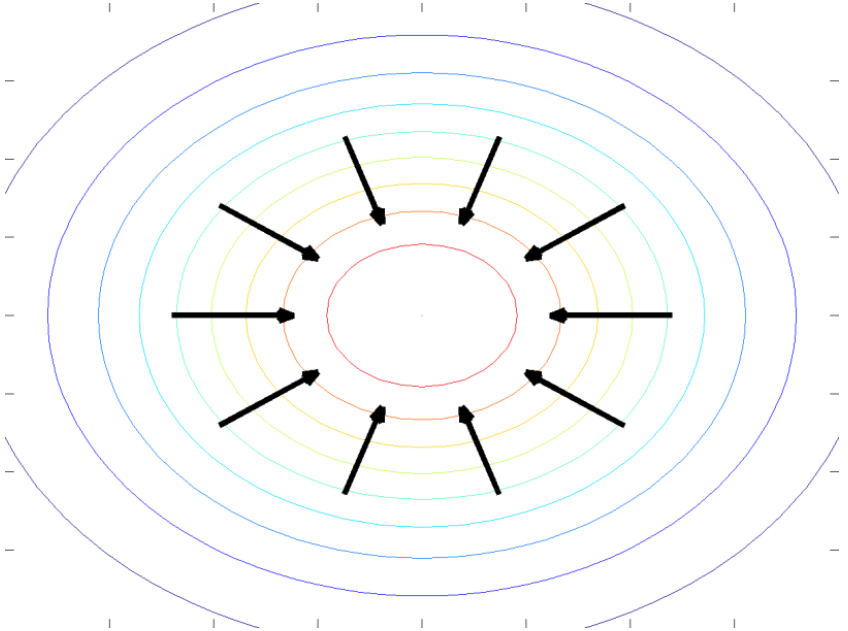


Παράγωγοι (2)

- Παράγωγος βαθμωτής συνάρτησης με διανυσματική μεταβλητή, δηλ. $y = f(\mathbf{x}), y \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n$:
  - $\nabla_{\mathbf{x}} f = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n}\right]^T$
  - Συχνά γράφεται και $\frac{\partial f}{\partial \mathbf{x}}$.

Καλείται **κλίση (gradient)**





Ελαχιστοποίηση

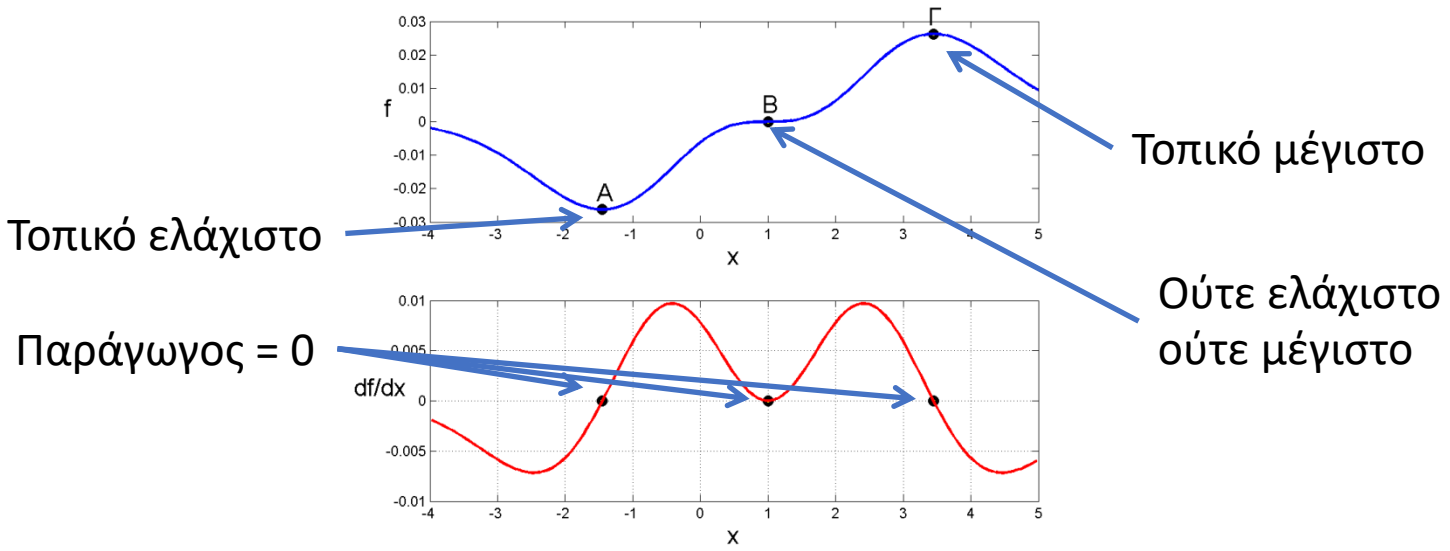
- Εντελώς παρόμοια με την μεγιστοποίηση
- Εμφανίζεται σε προβλήματα όπου ψάχνουμε το μικρότερο κόστος (πχ. στη μηχανική μάθηση, στα οικονομικά, κα.) ή την ελάχιστη ενέργεια (πχ. στη φυσική, κα).
- Όσο πιο πολύπλοκη είναι η συνάρτηση $f(\cdot)$ τόσο πιο δύσκολο να βρεθεί η λύση με αναλυτικό τρόπο.

Συνθήκες ελαχίστου

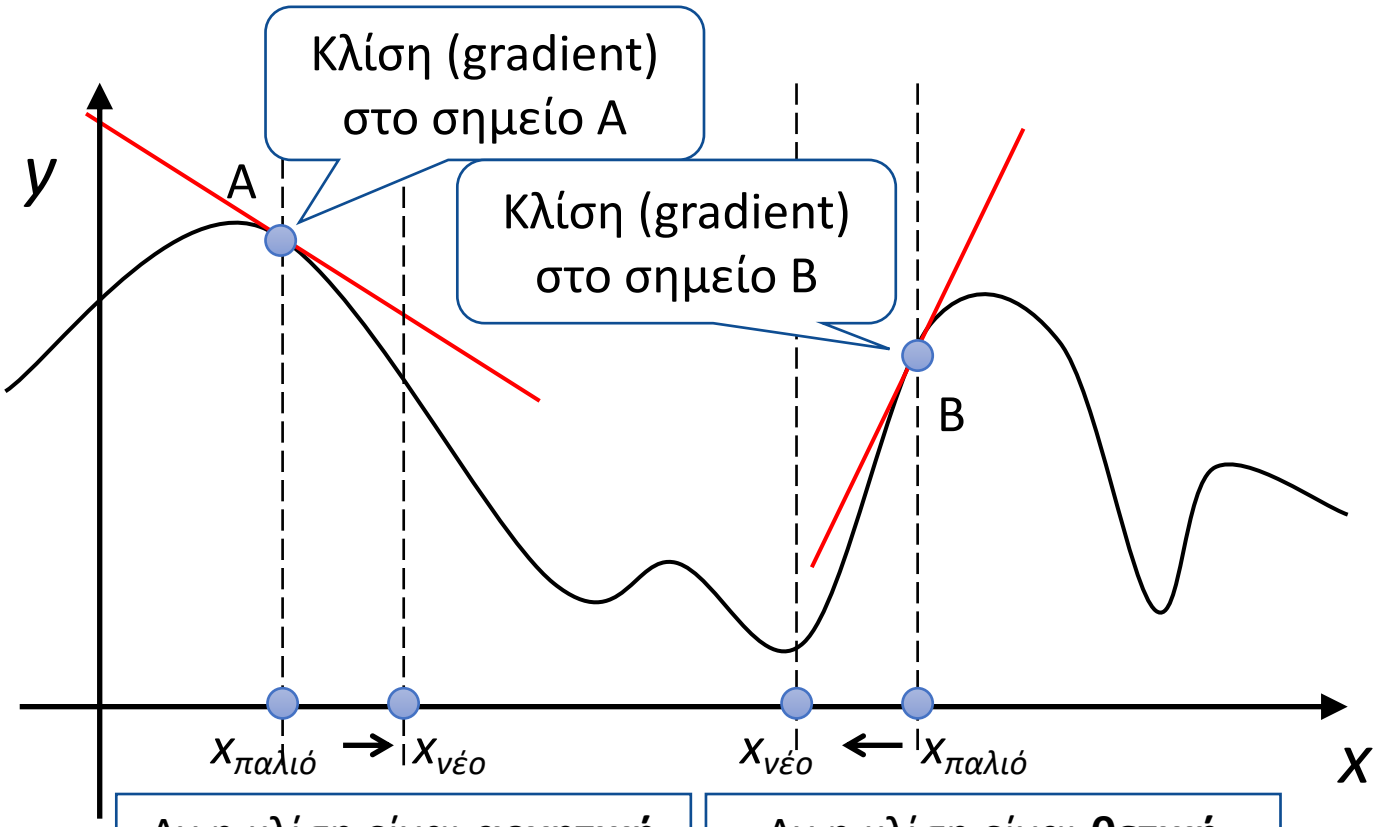
- Συνθήκη Karush-Kuhn-Tucker (KKT): Αν το σημείο $\mathbf{x}_*$ δίνει το ελάχιστο $f(\mathbf{x}_*)$ τότε

$$\nabla_{\mathbf{x}} f(\mathbf{x}_*) = \mathbf{0}$$

- Το αντίστροφο δεν ισχύει. Δηλαδή αν ικανοποιείται η συνθήκη δεν σημαίνει ότι το σημείο είναι οπωσδήποτε ελάχιστο.
- Παράδειγμα: Βαθμωτή συνάρτηση βαθμωτής μεταβλητής $y = f(x)$



Κατάβαση Δυναμικού/Κλίσης (Gradient Descent)



Αν η κλίση είναι **αρνητική** τότε πάω μπροστά (→)

Αν η κλίση είναι **θετική** τότε πάω πίσω (←)

Κατάβαση Δυναμικού (GD)

- Κινούμαστε αντίθετα απ' ό,τι λέει η παράγωγος: αν η παράγωγος είναι θετική τότε μειώνω το *x*, αν η παράγωγος είναι αρνητική τότε αυξάνω το *x*.
- Κάνω μικρά βήματα χρησιμοποιώντας το β = βήμα εκπαίδευσης (= μικρή θετική τιμή)

$$x(t+1) = x(t) - \beta \frac{df}{dx}$$

Αν το *x* είναι βαθμωτό

ή

$$\mathbf{x}(t+1) = \mathbf{x}(t) - \beta \nabla_{\mathbf{x}} f$$

Αν το **x** είναι διάνυσμα

Αλγόριθμος Back-Propagation (BP)

- Εκπαίδευση MLP με *L* στρώματα (Paul Werbos, 1974)
- Μάθηση με επίβλεψη
- $p = 1, \ldots, P$ ζεύγη $\{\mathbf{x}^{(p)}, \mathbf{t}^{(p)}\}$

Διάνυσμα εισόδου

Διάνυσμα στόχων

- Διάνυσμα εισόδου: $\mathbf{x}^{(p)} = \begin{bmatrix} x_1^{(p)} & \cdots & x_n^{(p)} \end{bmatrix}^T$
- Διάνυσμα στόχων: $\mathbf{t}^{(p)} = \begin{bmatrix} t_1^{(p)} & \cdots & t_m^{(p)} \end{bmatrix}^T$
- Διάνυσμα εξόδου: $\mathbf{y}^{(p)} = \begin{bmatrix} y_1^{(p)} & \cdots & y_m^{(p)} \end{bmatrix}^T$

Αλγόριθμος BP (2)

- Κριτήριο μάθησης: ελάχιστα τετράγωνα.
- Ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος για όλα τα πρότυπα (*P* το πλήθος)

$$J_{MLP} = \frac{1}{P}\sum_{p=1}^{P} \|\mathbf{t}^{(p)} - \mathbf{y}^{(p)}\|^2 = \frac{1}{P}\sum_{p=1}^{P}\sum_{i=1}^{m}\left[t_i^{(p)} - y_i^{(p)}\right]^2$$

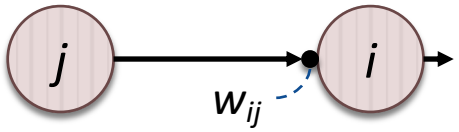
- Μέθοδος βελτιστοποίησης: Κατάβαση Δυναμικού

$$w_{ij}(k+1) = w_{ij}(k) - \beta\frac{\partial J_{MLP}}{\partial w_{ij}}$$

Αλγόριθμος BP (3)

- Έχει υπολογιστεί η παράγωγος:

$$\frac{\partial J_{MLP}}{\partial w_{ij}} = -\delta_i a_j$$



Αν ο νευρώνας i είναι στο τελευταίο στρώμα L	Αν ο νευρώνας i είναι σε οποιοδήποτε εσωτερικό στρώμα l < L
$\delta_i(L) = (t_i - y_i) f'(u_i)$	$\delta_i(l) = f'(u_i) \sum_{j=1}^{N(l+1)} w_{ji}\delta_j(l+1)$

$f'(u_i)$ = παράγωγος της συνάρτησης ενεργοποίησης f

- f = σιγμοειδής: $f'(u_i) = a_i[1 - a_i]$
- f = tanh(): $f'(u_i) = 1 - a_i^2$
- f = γραμμική: $f'(u_i) = 1$

Αλγόριθμος BP (4)

- Ένας **κανόνας εκπαίδευσης** άσχετα από το στρώμα

$$w_{ij}(k+1) = w_{ij}(k) + \beta \delta_i a_j$$

- Μόνη διαφορά ο τρόπος υπολογισμού του σφάλματος δ_i ανάλογα με το στρώμα.

Α. Στο εξωτερικό στρώμα:

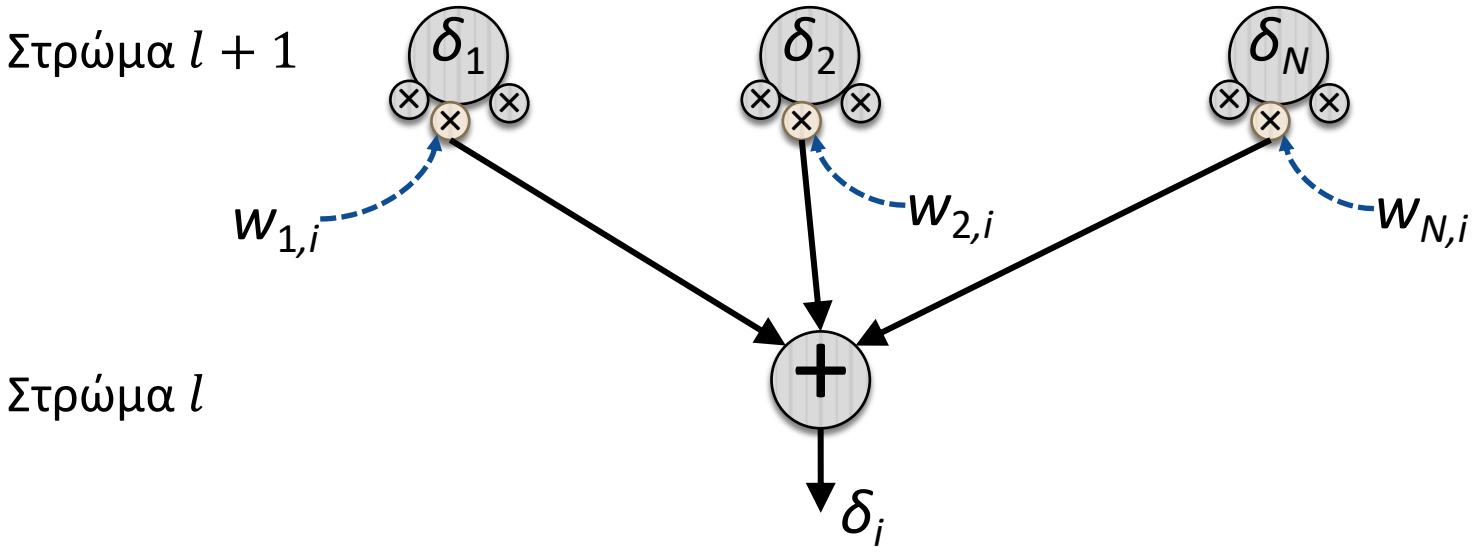
Το σφάλμα δ_i αποτελείται από το γινόμενο δύο όρων

1. Το σφάλμα $(t_i - y_i)$ και
2. Την παράγωγο $f'(u_i)$ της συνάρτησης ενεργοποίησης f των νευρώνων του στρώματος L.

Αλγόριθμος BP (5)

Β. Σε οποιοδήποτε εσωτερικό στρώμα $l < L$:

- *Υπολογισμός* $\delta_i(l)$ → προώθηση προς τα πίσω (backward propagation). Υπολογίζω το $\delta_i(l)$ του στρώματος l χρησιμοποιώντας τα $\delta_j(l+1)$ του πιο πάνω στρώματος.



Αλγόριθμος BP (6)

Αρχικοποίησε τα βάρη $w_{ij}(l)$ σε μικρές τυχαίες τιμές

Επανάλαβε {

Για κάθε *πρότυπο* p = 1, …, P { /* Εκπαίδευση */

/* Φάση ανάκλησης = Forward phase */

Υπολόγισε τις εξόδους

/* Φάση υπολογισμού δ = Backward phase */

Υπολόγισε τα σφάλματα δ

/* Φάση ενημέρωσης βαρών = Update phase */

Ανανέωσε τα βάρη

}

} *Μέχρι* J_{MLP} *μικρότερο από κάποιο κατώφλι MINJ ή να έχουμε φτάσει στο μέγιστο αριθμό εποχών*

Στοχαστική κατάβαση δυναμικού

Stochastic gradient descent (SGD)

- Υποθέστε ότι η συνάρτηση που ελαχιστοποιούμε είναι **άθροισμα Ν όρων**, για παράδειγμα:

$$f(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w})$$

- Η παράγωγος είναι

$$\nabla_{\mathbf{w}} f = \sum_{n=1}^{N} \nabla_{\mathbf{w}} E_n$$

- Αν εφαρμόζουμε κλασική κατάβαση δυναμικού πρέπει να αθροίσουμε όλους τους όρους $\sum_{n=1}^{N} \nabla_{\mathbf{w}} E_n$, για $n = 1, \dots, N$ και μετά να ενημερώσουμε τα βάρη $\mathbf{w}$.
- Μια άλλη προσέγγιση: Επιλέγουμε δειγματοληπτικά (τυχαία) και δημιουργούμε υποσύνολα του συνόλου δεδομένων, που ονομάζονται batches. Στη συνέχεια ανανεώνουμε τα βάρη με βάση μόνο τα δεδομένα που ανήκουν στο batch.
- Η προσέγγιση αυτή συγκλίνει πιο γρήγορα και λέγεται **Στοχαστική Κατάβαση Δυναμικού** (**Stochastic Gradient Descent - SGD**).

Στοχαστική κατάβαση δυναμικού

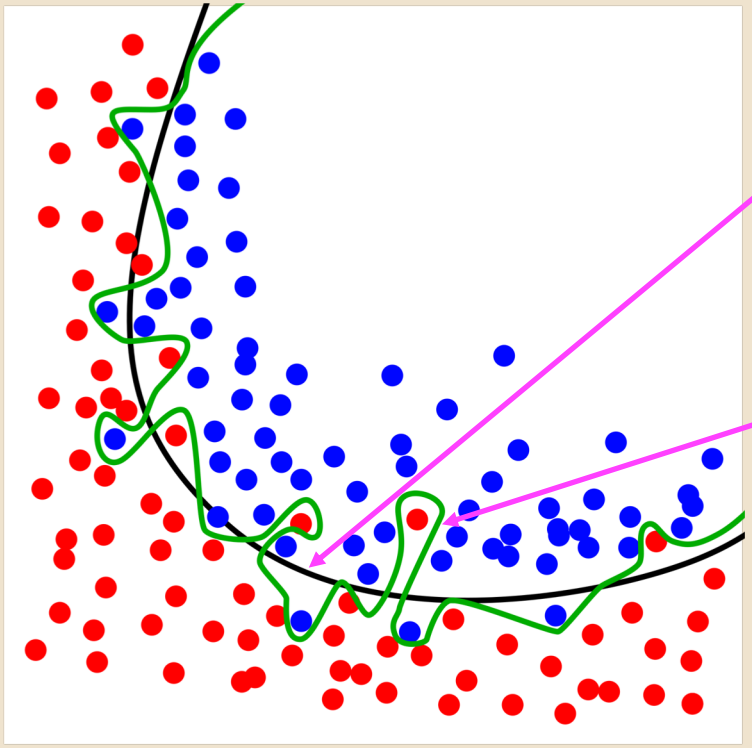
Stochastic gradient descent (SGD)

- Στην ακραία περίπτωση που το μέγεθος του batch B=1, έχουμε ανανέωση των βαρών μετά από την παρουσίαση ενός μόνο προτύπου.
- Η ακραία περίπτωση B=N αντιστοιχεί πρακτικά στην κλασική κατάβαση δυναμικού
- Για ποιο λόγο μπορεί να είναι προτιμότερη η SGD έναντι της GD;
  - Σε μεγάλα datasets, είναι υπολογιστικά ακριβό να γίνεται ανανέωση των βαρών μετά από την παρουσίαση του συνόλου των δεδομένων
  - Επιπλέον, η δειγματοληψία δεδομένων εισάγει κατά έναν τρόπο «θόρυβο», ο οποίος αποτρέπει την εύρεση «ρηχών» τοπικών ελαχίστων, γεγονός που είναι καλό για την βελτιστοποίηση μη-κυρτών συναρτήσεων

Επίδοση αλγορίθμου back-propagation

- Ο αλγόριθμος back-propagation δεν βρίσκει *πάντα* λύση (παρότι με βάση το θεώρημα προσέγγισης πάντα υπάρχει)
  - Δεν προσεγγίζει *πάντα* τη συνάρτηση εισόδου-εξόδου με σφάλμα μικρότερο του ε
  - Όσο πιο σύνθετη η δομή του δικτύου, τόσο πιο πολλές οι πιθανότητες να βρεθεί λύση (σε κάποιο αριθμό εποχών)
    - Δεν είναι όμως απαραίτητο ότι η λύση αυτή θα είναι πρακτικά καλή (*πρόβλημα overfitting*)
- Ο αλγόριθμος back-propagation δεν βρίσκει *πάντα* τη λύση που δίνει το μικρότερο τετραγωνικό σφάλμα
  - Εξαρτάται από πολλούς παράγοντες όπως η αρχικοποίηση (*πρόβλημα τοπικών ελαχίστων*)

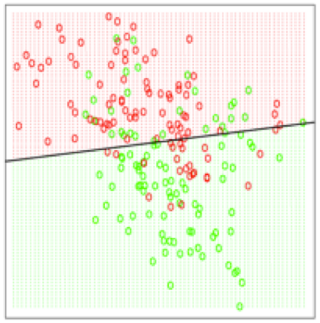
Πρόβλημα υπερπροσαρμογής (overfitting)

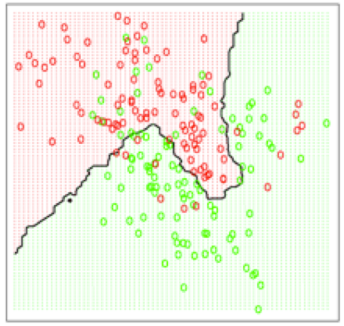


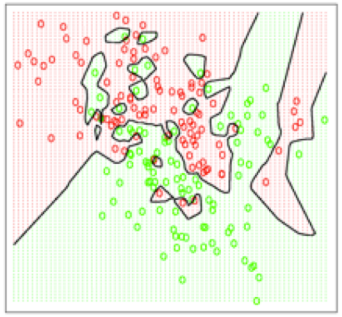
Λιγότεροι νευρώνες στο κρυμμένο στρώμα

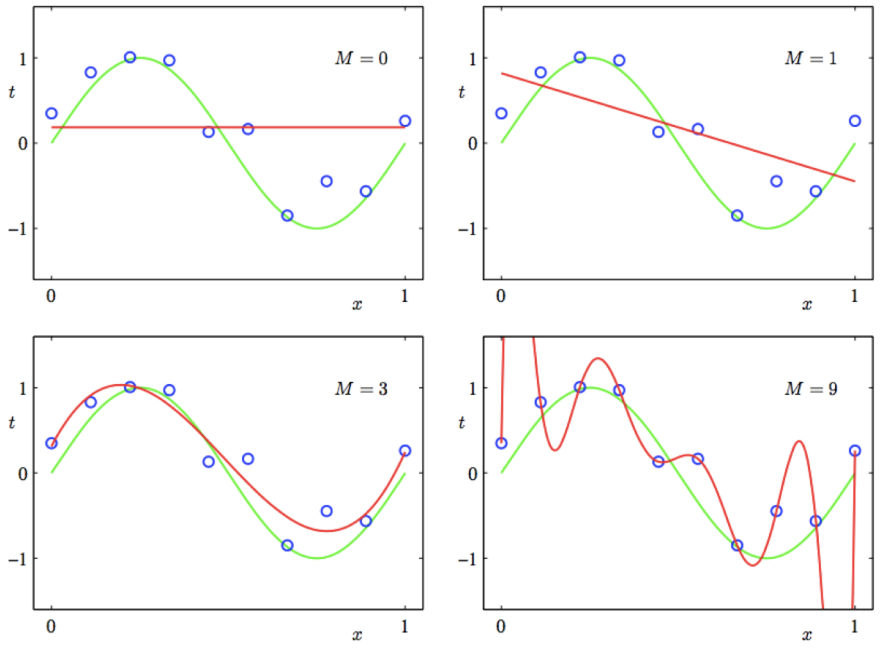
Περισσότεροι νευρώνες στο κρυμμένο στρώμα

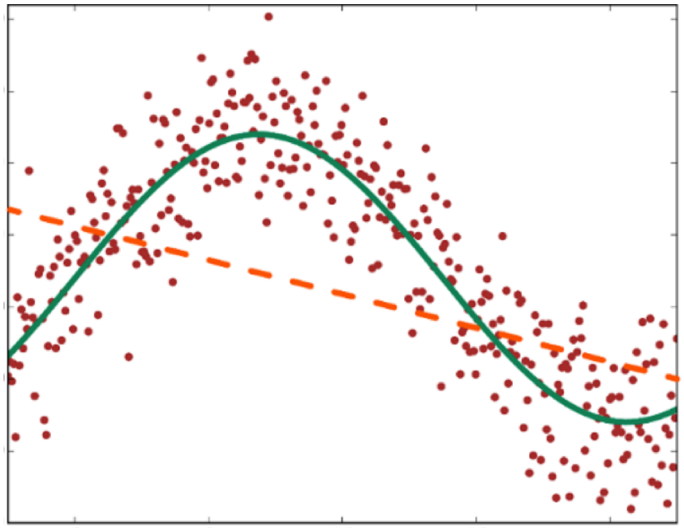
Πρόβλημα υπερπροσαρμογής (overfitting)





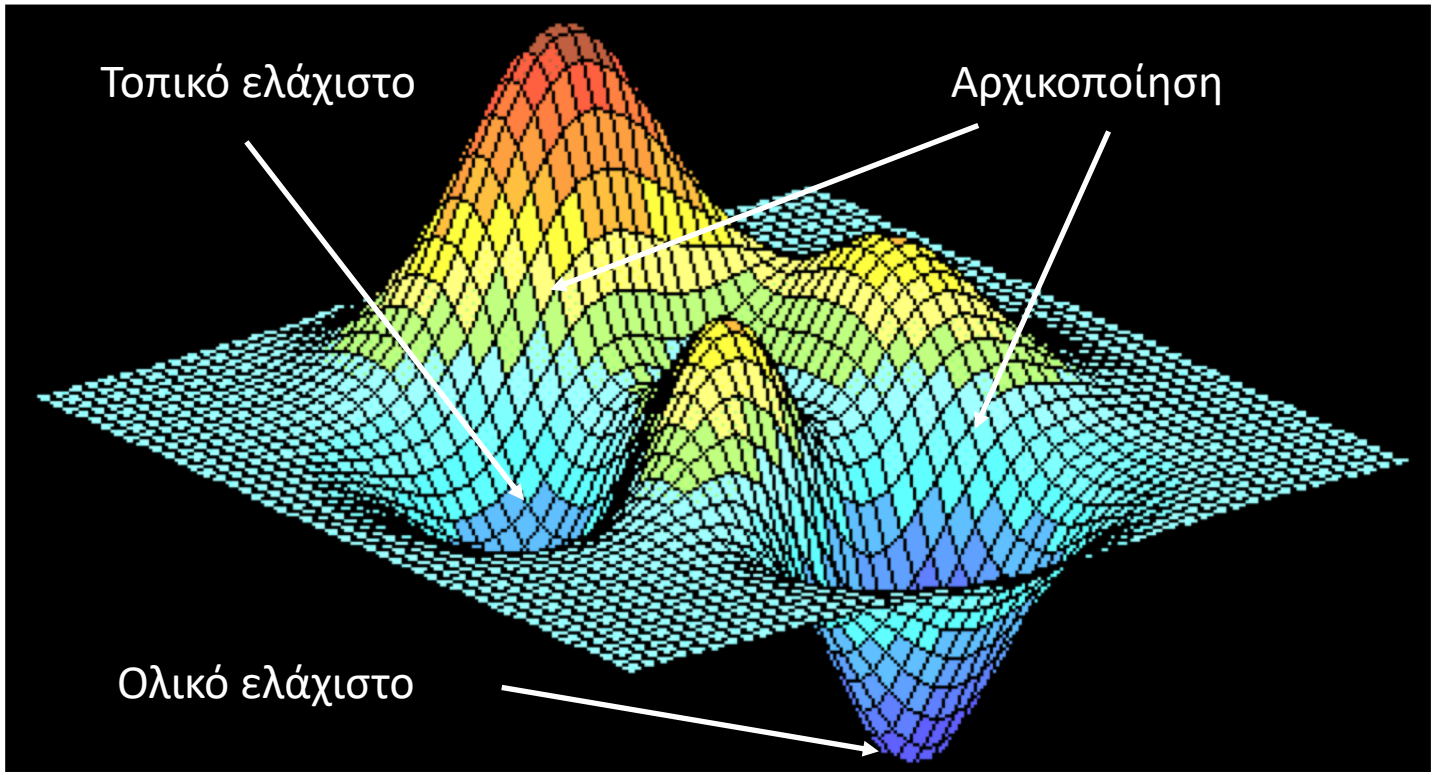






Πρόβλημα τοπικών ελαχίστων

Καμπύλη σφάλματος ως προς τα βάρη του δικτύου



Επιτάχυνση σύγκλισης με χρήση Ορμής (momentum)

- Χρήση ορμής (momentum): Κρατάμε την προηγούμενη διόρθωση $\Delta w_{ij}(k) = w_{ij}(k) - w_{ij}(k-1)$ και την προσθέτουμε στην τωρινή:

$$w_{ij}(k+1) = w_{ij}(k) + \beta \delta_i \alpha_j + \mu \cdot \Delta w_{ij}(k)$$

- Ο συντελεστής μ πρέπει να είναι μικρότερος από το 1 αλλά κοντά στο 1 (π.χ. 0.95). Όσο πιο κοντά, τόσο πιο γρήγορα τρέχει, αλλά αν είναι πολύ κοντά μπορεί να οδηγήσει σε απόκλιση.
- Αποτέλεσμα: Σε περιοχές όπου το *J* μειώνεται αργά, η ορμή επιταχύνει τον αλγόριθμο. Σε περιοχές όπου το *w* ταλαντώνεται επιβάλλει εξομάλυνση (είτε τείνει να κινείται προς μια κατεύθυνση είτε μειώνει τις ταλαντώσεις).

Πρακτικά θέματα εκπαίδευσης

- Βήμα εκπαίδευσης β:
  - Η βέλτιστη τιμή εξαρτάται από το πρόβλημα και το δίκτυο
  - Μικρό, αλλά πόσο μικρό; (0.1, 0.01, 0.001, …);
  - Ιδέα-1: διαφορετικά β_ℓ για διαφορετικά στρώματα ℓ
  - Ιδέα-2: διαφορετικά β_{ij} για διαφορετικά βάρη w_{ij}
  - Ιδέα-3: το β μεταβάλλεται από εποχή σε εποχή. Πχ.

$$\beta_{ij}(k) = \beta_{ij}(k-1) + \eta \left.\frac{\partial J}{\partial w_{ij}}\right|_k \cdot \left.\frac{\partial J}{\partial w_{ij}}\right|_{k-1}$$

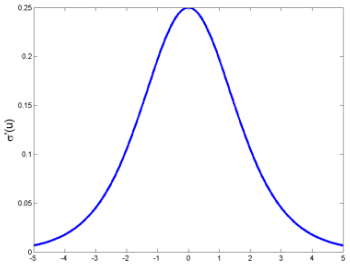
Πρακτικά θέματα εκπαίδευσης (2)

- Κριτήρια τερματισμού:
  - Το σφάλμα $J(k)$ σε κάποια εποχή k είναι κάτω από κάποιο όριο ε
  - Το σφάλμα σε δύο διαδοχικές εποχές $k, k-1$, δεν βελτιώνεται σημαντικά: $|J(k) - J(k-1)| < \varepsilon$
  - Τα βάρη σε μια εποχή k δέχονται αμελητέα διόρθωση: $\sum_{i,j}\left(\Delta w_{ij}\right)^2 < \varepsilon$
  - Φτάνουμε στο μέγιστο πλήθος εποχών

Πρακτικά θέματα εκπαίδευσης (3)

- Αρχικοποίηση Βαρών:
  - Πρέπει να αποφεύγεται οι αρχικές τιμές των βαρών να είναι ίδιες μεταξύ τους.
  - Τα βάρη δεν πρέπει να είναι πολύ μεγάλα για να μην επέρχεται κορεσμός της παραγώγου. Πχ. για τη σιγμοειδή συνάρτηση $\sigma(u)$ η παράγωγος είναι

$$\sigma'(u) = \sigma(u) \cdot \big(1 - \sigma(u)\big)$$



  - Μεγάλο $\mathbf{w} \rightarrow$ Μεγάλο $u = \mathbf{w}^T\mathbf{x} \rightarrow$ Μικρή παράγωγος $\rightarrow$ Μικρή διόρθωση βαρών
  - Καλό είναι να αρχικοποιούνται με μικρές τυχαίες τιμές
  - Καλή ιδέα οι πολλαπλές εκπαιδεύσεις από διαφορετικές αρχικές τιμές, ώστε να επιβεβαιωθεί ότι η επίδοση του ΝΔ είναι ανεξάρτητη των αρχικών τιμών. Επίσης, έτσι αυξάνεται η πιθανότητα αποφυγής παγίδευσης σε τοπικά ελάχιστα.

Βιβλιογραφία

- [1] Κ. Διαμαντάρας, Δ. Μπότσης, Μηχανική Μάθηση, Εκδόσεις Κλειδάριθμος, 2019.
- [2] S. Theodoridis, Machine Learning: A Bayesian and Optimization Perspective, 2nd Edition, Academic Press, 2020.
- [3] Shai Ben-David and Shai Shalev-Shwartz, Understanding Machine Learning, Cambridge University Press
- Διαφάνειες των συγγραφέων για το σύγγραμμα [1].