Τεχνητή Νοημοσύνη

19η διάλεξη (2023-24)

Ίων Ανδρουτσόπουλος

http://www.aueb.gr/users/ion/

Οι διαφάνειες αυτές βασίζονται σε ύλη των βιβλίων:

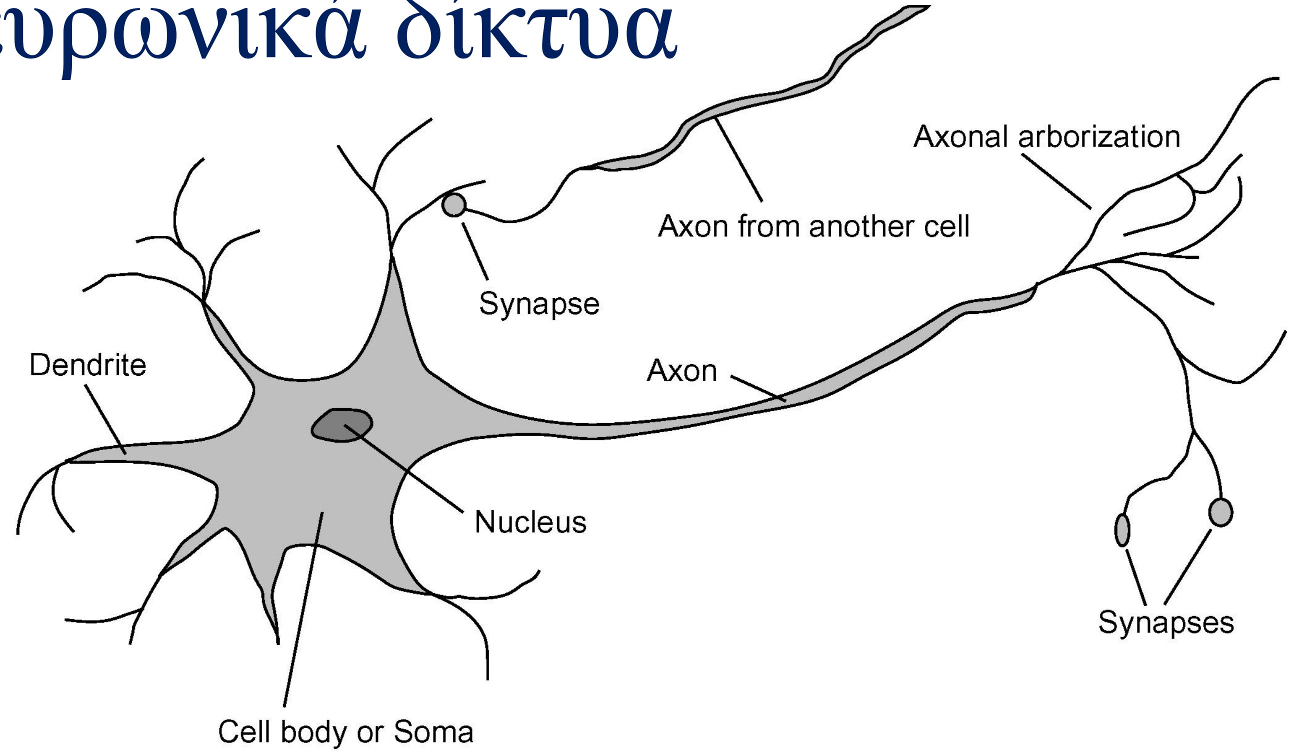
- *Artificial Intelligence – A Modern Approach* των S. Russel και P. Norvig, 2η και 4η έκδοση, Prentice Hall, 2003 και 2020,
- *Τεχνητή Νοημοσύνη* των Βλαχάβα κ.ά., 3η έκδοση, Β. Γκιούρδας Εκδοτική, 2006,
- *Machine Learning* του T. Mitchell, McGraw-Hill, 1997.

Τα περισσότερα σχήματα των διαφανειών προέρχονται από τα παραπάνω βιβλία και διαλέξεις.

Τι θα ακούσετε σήμερα

- Νευρωνικά δίκτυα.
  - Φυσικά και τεχνητά νευρωνικά δίκτυα.
  - Perceptron.
  - Γραμμική διαχωρισιμότητα.
  - Πολύ-επίπεδα Perceptron (MLPs).
  - Αλγόριθμος ανάστροφης μετάδοσης (back-propagation).

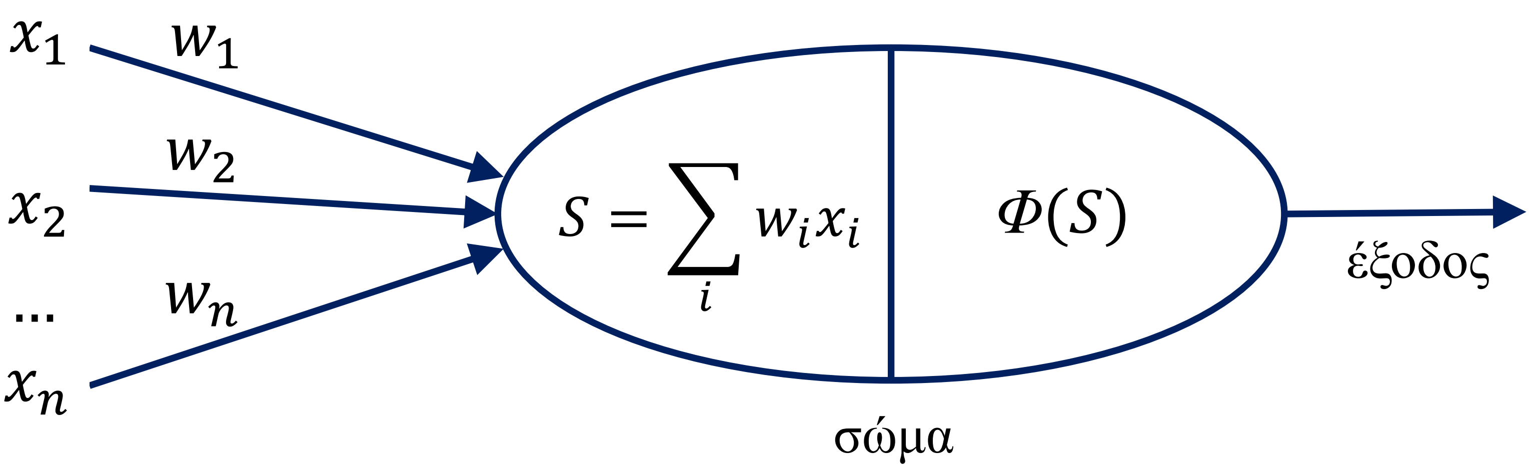
Φυσικά νευρωνικά δίκτυα



- **Νευρώνας**: κύτταρο του εγκεφάλου.
  - **Σώμα**: το κυρίως μέρος, που περιέχει τον **πυρήνα**.
  - **Δενδρίτες**: λαμβάνουν σήματα από άλλους νευρώνες.
  - **Άξονας**: παρέχει κοινή έξοδο προς πολλούς άλλους νευρώνες.
  - **Σύναψη**: μέσω ηλεκτροχημικών μηχανισμών μεταβάλλεται η αγωγιμότητά της.
- **Νευρωνικό δίκτυο**: δίκτυο συνδεδεμένων νευρώνων.

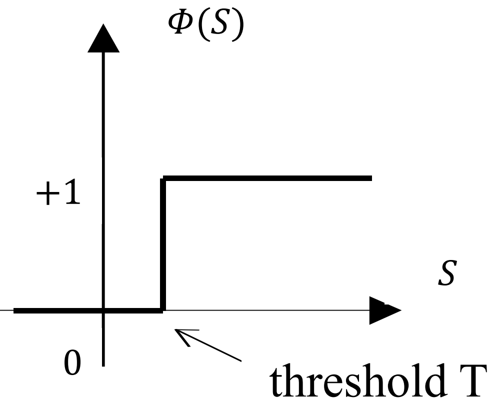
Τεχνητά νευρωνικά δίκτυα

- **Τεχνητός νευρώνας**:
  - **Είσοδοι**: πραγματικές τιμές.
  - **Βάρη εισόδων**: πραγματικές τιμές (χονδρικά συνάψεις).
  - **Σώμα**: υπολογίζει το **ζυγισμένο άθροισμα** των εισόδων, κατόπιν εφαρμόζει τη **συνάρτηση ενεργοποίησης** στο ζυγισμένο άθροισμα.

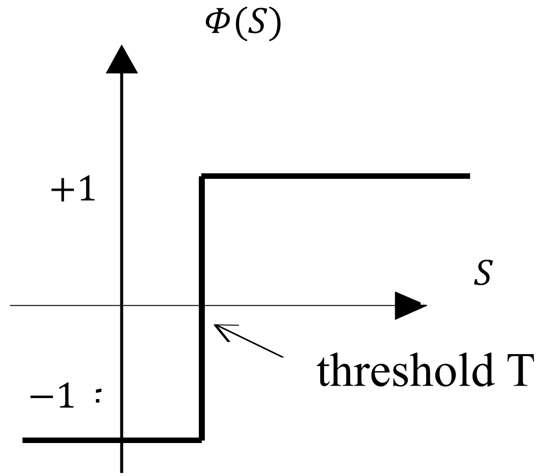


Τροποποιημένη εικόνα από το βιβλίο των Βλαχάβα κ.ά.

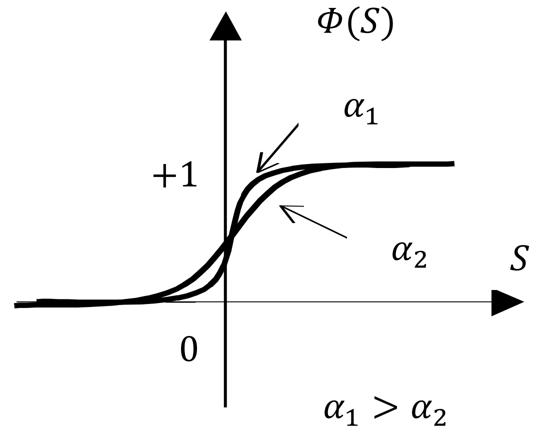
Συναρτήσεις ενεργοποίησης



step function



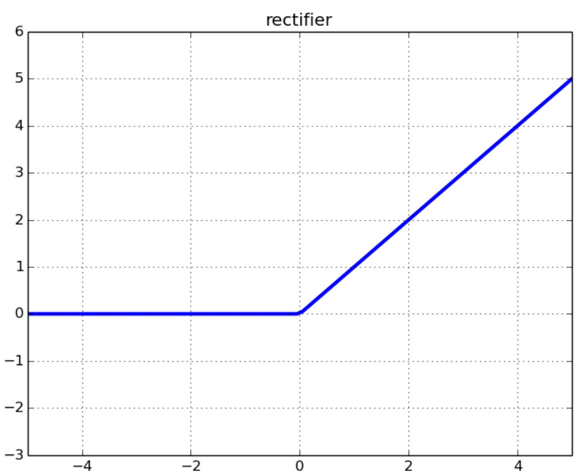
sign function



sigmoid
(logistic function)

$$\Phi(S) = \frac{1}{1 + e^{-a \cdot S}}$$

Η **σιγμοειδής** συνάρτηση είναι παντού **παραγωγίσιμη**. Η **υπερβολική εφαπτομένη** (**tanh**) είναι παρόμοια αλλά με τιμές στο $(-1, +1)$. Συνήθως προτιμότερη της σιγμοειδούς, εκτός αν θέλουμε τιμές στο (0, 1).

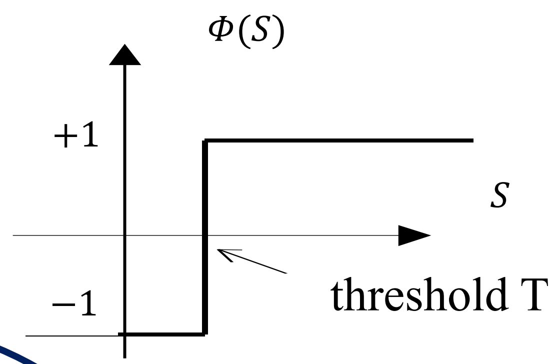


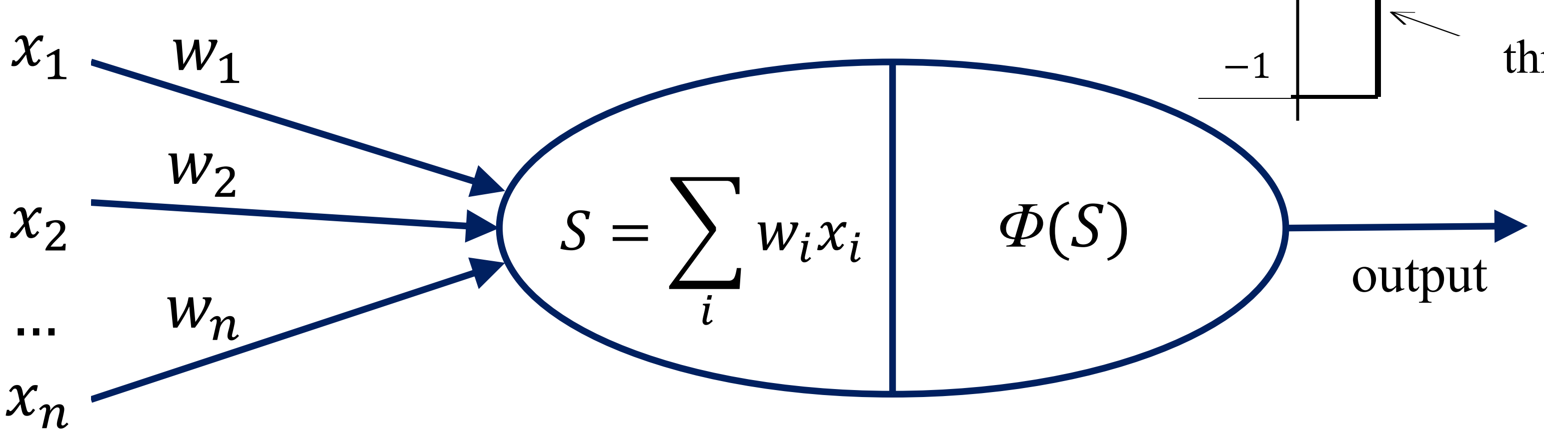
Rectified Linear Unit (ReLU)

Top images from the book of Vlahavas et al. Bottom image from http://datascience.stackexchange.com/questions/5706/

Perceptron

- Μόνο **ένας νευρώνας**, με συνάρτηση ενεργοποίησης **προσήμου**.
  - **Ισοδύναμα**, με **βηματική** (step) συνάρτηση ενεργοποίησης.
  - Το Perceptron μπορεί να **γενικευθεί** (έγινε αργότερα), ώστε να χρησιμοποιεί **σιγμοειδή ή άλλη** συνάρτηση ενεργοποίησης.
  - Ένα **μεμονωμένο Perceptron** είναι **γραμμικός διαχωριστής**.
  - **Για μη γραμμικά διαχωρίσιμα** προβλήματα, χρειαζόμαστε **πολυ-επίπεδο Perceptron (MLP)**.

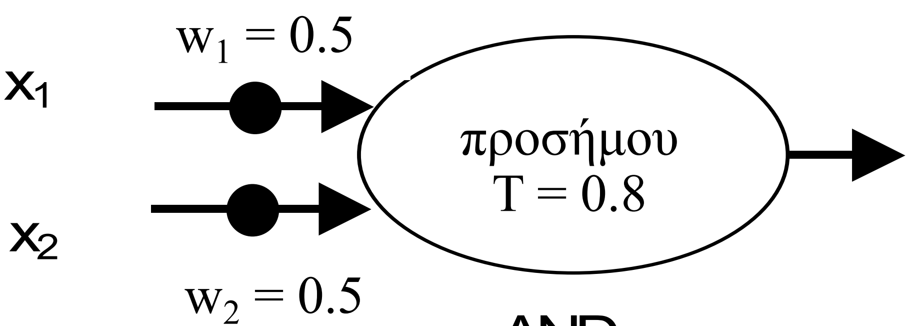




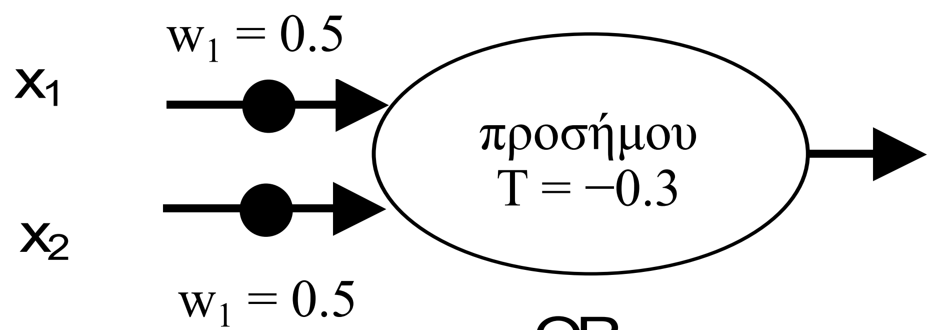
Original images from the book of Vlahavas et al.

Λογικές συναρτήσεις με Perceptron

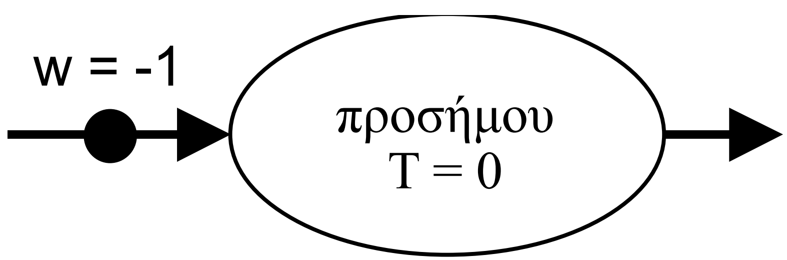
αληθές: 1, ψευδές: -1



AND



OR

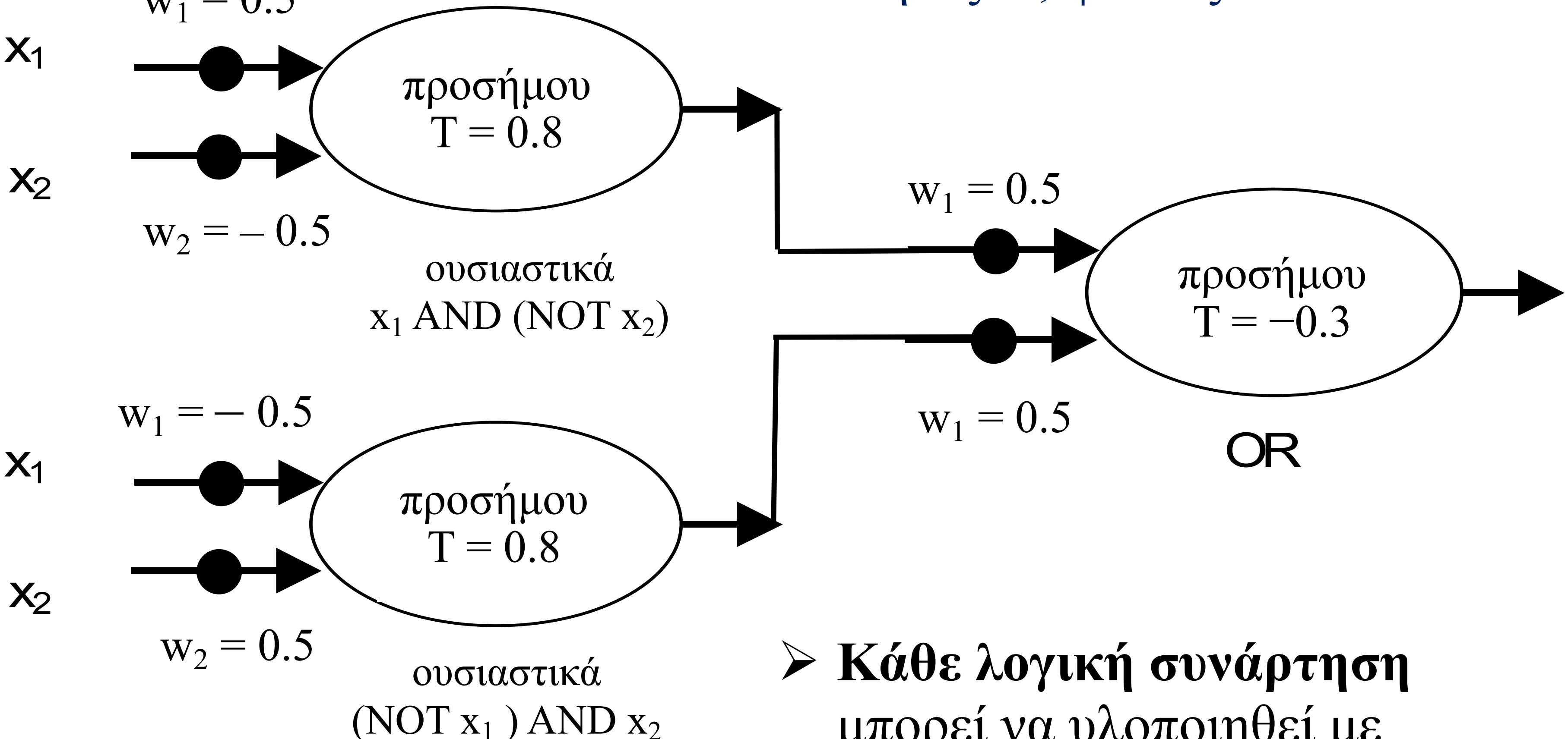


NOT

- **Δεν μπορούν** να υλοποιηθούν **όλες** οι λογικές συναρτήσεις με **μόνο ένα** Perceptron.
  - Π.χ. **XOR**.

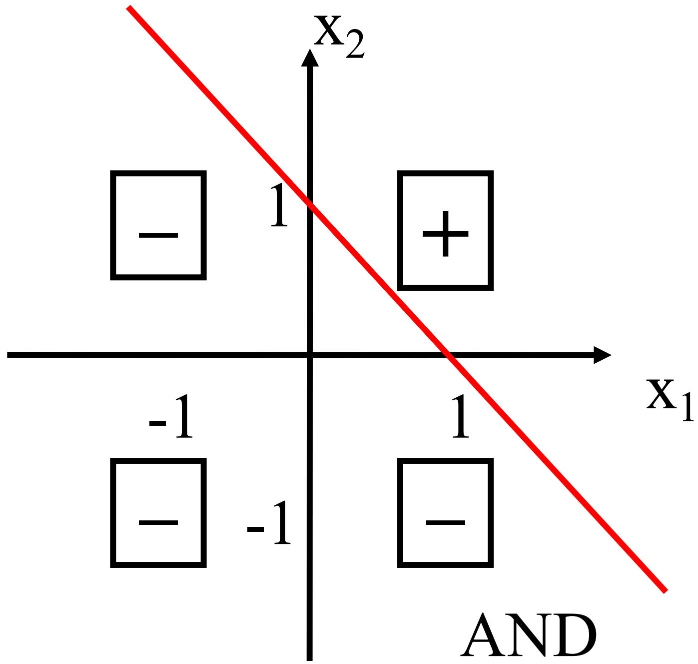
Υλοποίηση XOR με δύο επίπεδα

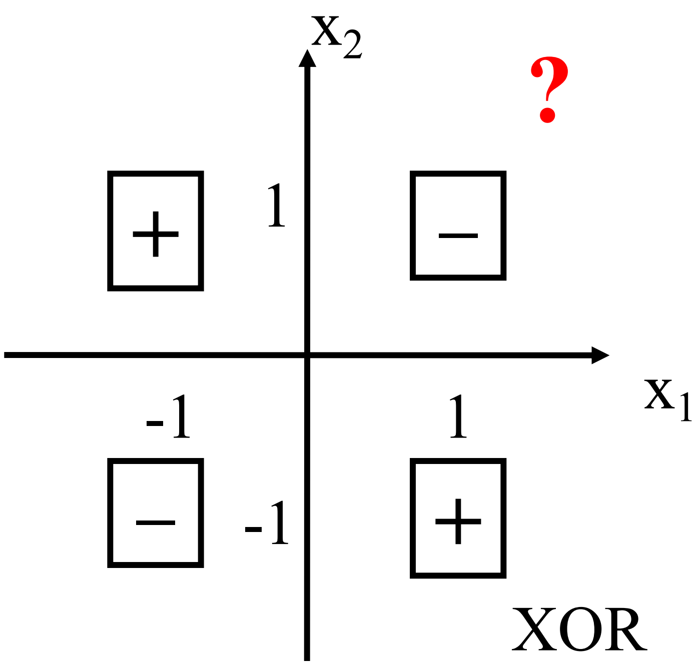
αληθές: 1, ψευδές: -1



- **Κάθε λογική συνάρτηση** μπορεί να υλοποιηθεί με δίκτυο Perceptrons **δύο επιπέδων**.

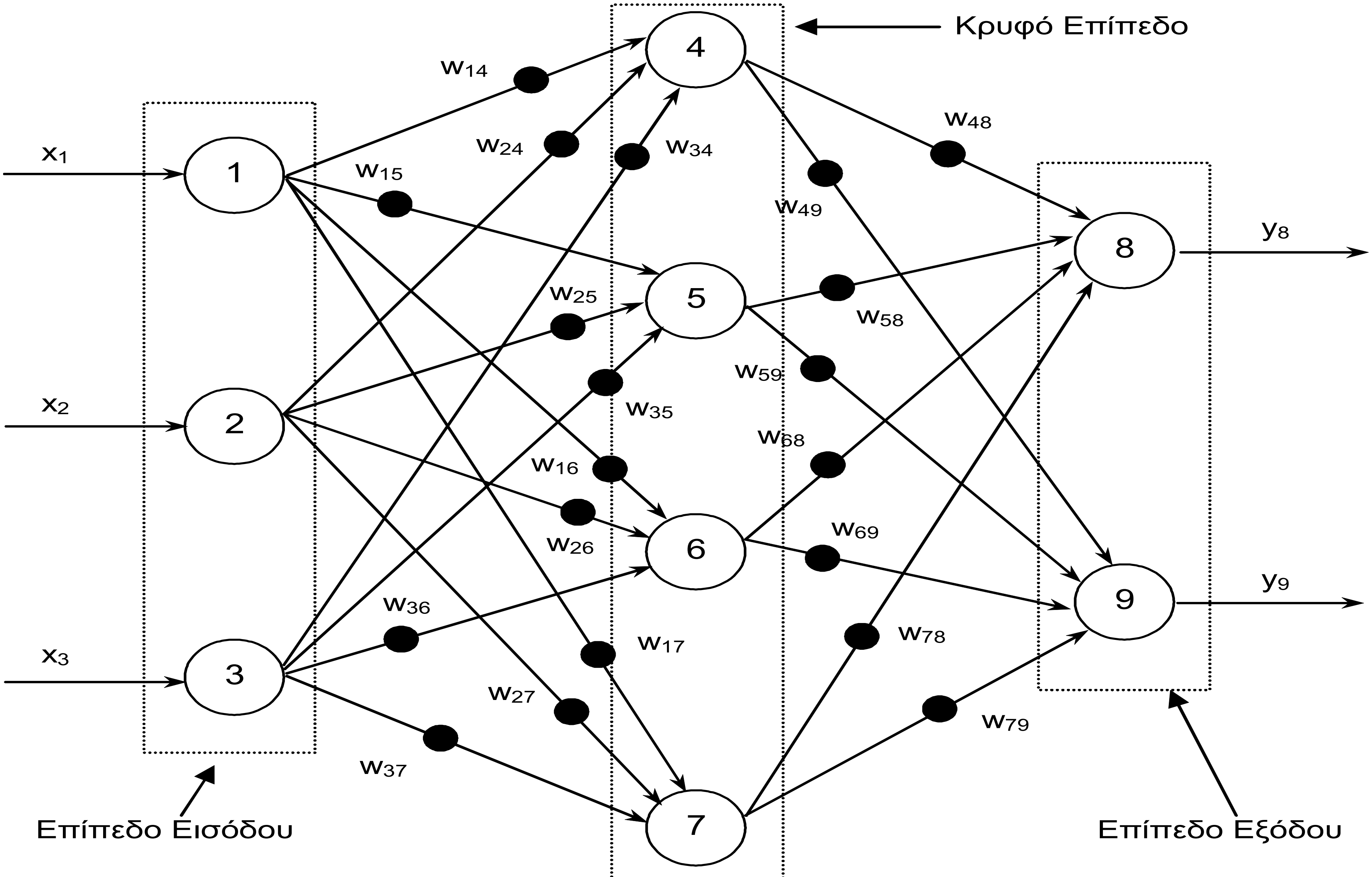
Γραμμική διαχωρισιμότητα





- Το **Perceptron** (όπως και ένας ταξινομητής λογιστικής παλινδρόμησης) μπορεί να μάθει **μόνο γραμμικά διαχωρίσιμες** συναρτήσεις (άσκηση μελέτης).
- Για **μη γραμμικά διαχωρίσιμες** συναρτήσεις, χρειαζόμαστε **πολυ-επίπεδο Perceptron (MLP)**.

Πολυ-επίπεδο Perceptron (MLP)

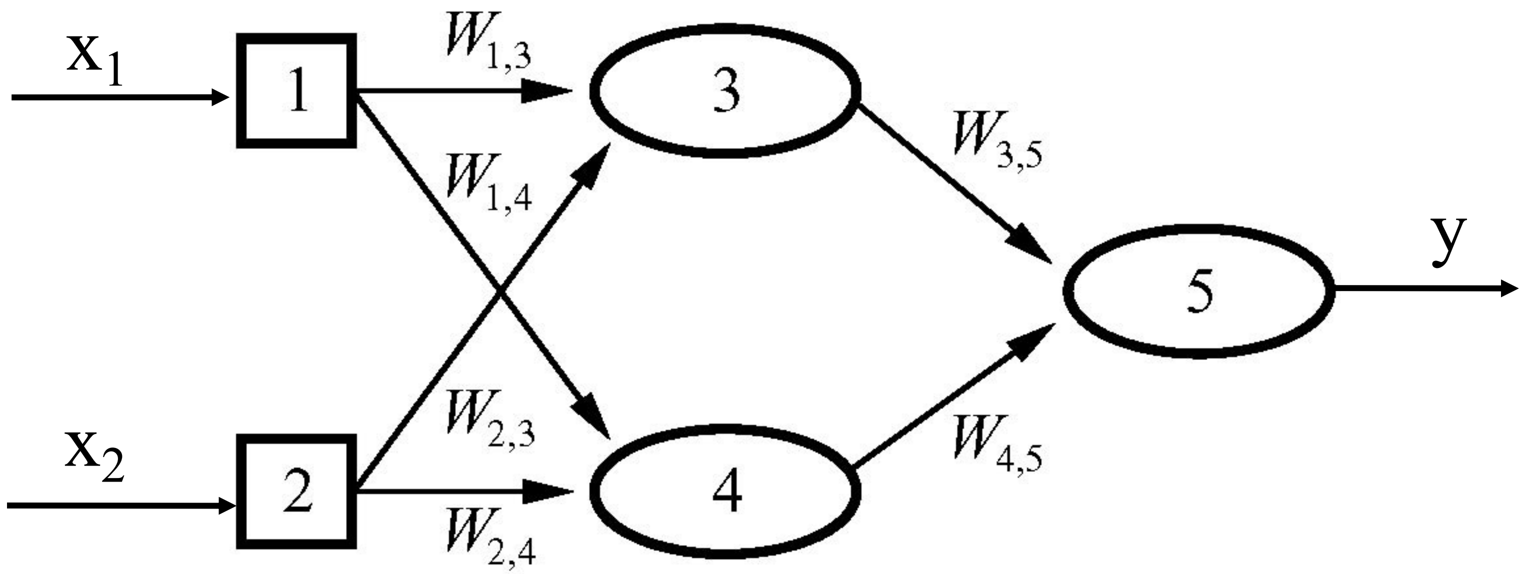


Εικόνα από το βιβλίο των Βλαχάβα κ.ά.

Πολυ-επίπεδο Perceptron (MLP)

- Νευρώνες **εισόδου**: συχνά χωρίς ουσιαστικούς υπολογισμούς.
  - Π.χ. μόνο μία είσοδος στον καθένα, μοναδιαίο βάρος, Φ(S) = S.
  - Χρήσιμοι αν θέλουμε να προσθέσουμε μια σταθερά στην κάθε είσοδο ή να πολλαπλασιάσουμε την είσοδο με κάποιο βάρος (ή πίνακα).
- Επίπεδο **εξόδου** και **ενδιάμεσα** («κρυφά») επίπεδα.
- ΤΝΔ με **απλή τροφοδότηση** (feed-forward networks):
  - Οι είσοδοι ενός νευρώνα **δεν μπορούν** να προέρχονται από νευρώνες του **ίδιου** ή **επόμενου** επιπέδου.
- ΤΝΔ με **ανατροφοδότηση** (recurrent networks, RNNs):
  - Είσοδοι και από νευρώνες του **ίδιου** ή **επόμενου** επιπέδου.
  - Η έξοδος εξαρτάται και από την **προηγούμενη κατάσταση** του δικτύου.
  - Χρησιμοποιούνται σε ΤΝΔ που επεξεργάζονται **ακολουθίες** (π.χ. ακολουθίες λέξεων κειμένων).

Μάθηση με ΤΝΔ



- $y = \Phi(w_{3,5} \cdot x_{3,5} + w_{4,5} \cdot x_{4,5}) =$
  $\Phi(w_{3,5} \cdot \Phi(w_{1,3} \cdot x_1 + w_{2,3} \cdot x_2) + w_{4,5} \cdot \Phi(w_{1,4} \cdot x_1 + w_{2,4} \cdot x_2))$
- Μεταβάλλοντας τα **βάρη** μαθαίνουμε **διαφορετική συνάρτηση**.
- **Ποιες συναρτήσεις** μπορούμε να μάθουμε; Εξαρτάται από τη Φ, το **πλήθος των επιπέδων**, το πλήθος **νευρώνων ανά επίπεδο** κ.λπ.

Μάθηση με ΤΝΔ

- **Επιβλεπόμενη** μάθηση:
  - **Εκπαίδευση:** παρέχονται είσοδοι και οι επιθυμητές αποκρίσεις. Τα **βάρη προσαρμόζονται**, ώστε να παράγονται οι επιθυμητές αποκρίσεις.
  - **Χρήση**: παρέχονται είσοδοι. Χρησιμοποιούμε τα **βάρη που προέκυψαν από την εκπαίδευση** και εμπιστευόμαστε τις αποκρίσεις.
  - Κατάταξη (κατηγοριοποίηση) σε **δύο κατηγορίες**: συνήθως **ένας νευρώνας εξόδου**. Αν η έξοδός του ξεπερνά ένα **κατώφλι**, κατατάσσουμε στη μία κατηγορία, διαφορετικά στην άλλη.
  - Με περισσότερες **κατηγορίες**: συνήθως **ένας νευρώνας εξόδου για κάθε κατηγορία**. (Μία περίπτωση μπορεί να ανήκει σε πολλές κατηγορίες.)
  - Μάθηση συνεχούς συνάρτησης (προβλήματα **παλινδρόμησης**): συνήθως **ένας ή περισσότεροι νευρώνες εξόδου** (π.χ. στροφή, επιτάχυνση αυτοκινήτου).
- **Μη επιβλεπόμενη** μάθηση:
  - Π.χ. ΤΝΔ μπορούν να χρησιμοποιηθούν και για **ομαδοποίηση** (clustering).

Κανόνας ενημέρωσης βαρών Perceptron

- Σφάλμα στο **<u>τρέχον</u>** παράδειγμα εκπαίδευσης:

$$E_i(\vec{w}) = \frac{1}{2}(t^{(i)} - o^{(i)})^2 = \frac{1}{2}\left(t^{(i)} - \Phi\left(\sum_l w_l x_l^{(i)}\right)\right)^2$$

Ξεκινώ με τυχαία βάρη. Μετράω σφάλμα $E_i(\vec{w})$ στο <u>τρέχον</u> παράδειγμα εκπαίδευσης με τα τρέχοντα βάρη $\vec{w}$. Προς τα πού να μεταβάλω τα βάρη;

Πηγή εικόνας: http://en.wikipedia.org/wiki/Gradient_descent

Η κλίση $\nabla E_i(\vec{w})$ είναι ένα διάνυσμα που δείχνει προς την κατεύθυνση μεταβολής των βαρών που οδηγεί στη μεγαλύτερη **αύξηση** του E_i. Το $-\nabla E_i(\vec{w})$ δείχνει προς την μεγαλύτερη **μείωση** του E_i.

Τροποποιούμε τα $\boldsymbol{w_i}$ κατά η προς την κατεύθυνση που προκαλεί τη μεγαλύτερη μείωση του E_i:

$$\vec{w} \leftarrow \vec{w} - \eta \cdot \nabla E_i(\vec{w})$$

Στοχαστική κατάβαση κλίσης.

Εκπαίδευση Perceptron με SGD

Χρησιμοποιούμε εδώ συνεχή συνάρτηση ενεργοποίησης Φ(s).

1. Ξεκίνα με τυχαία βάρη $\vec{w}$.
2. Θέσε $i \leftarrow 1$ και $s \leftarrow 0$.
3. Έστω t^i η επιθυμητή απόκριση για το i-στό παράδειγμα και o^i η τρέχουσα απόκριση για το παράδειγμα αυτό.
4. Θέσε $s \leftarrow s + E_i(\vec{w})$, με: $E_i = \frac{1}{2}\left[t^{(i)} - o^{(i)}\right]^2$
5. Ενημέρωσε τα βάρη: $\vec{w} \leftarrow \vec{w} - \eta \, \nabla E_i(\vec{w})$
6. Αν υπάρχει (i+1)-στό παράδειγμα, θέσε $i \leftarrow 1 + 1$ και πήγαινε στο βήμα 3.
7. Αν $s > \delta$ και δεν υπερβήκαμε τον μέγιστο αριθμό επαναλήψεων («εποχών»), πήγαινε στο βήμα 2.

- Το η είναι θετική μικρή σταθερά (π.χ. 0,01).

Κανόνας ενημέρωσης βαρών Perceptron

$$E_i(\vec{w}) = \frac{1}{2}(t^{(i)} - o^{(i)})^2 = \frac{1}{2}(t^{(i)} - \Phi\left(\sum_{l=1}^{n} w_l x_l^{(i)}\right))^2$$

$$\nabla E_i(\vec{w}) = \left\langle \frac{\partial E_i(\vec{w})}{\partial w_1}, \dots, \frac{\partial E_i(\vec{w})}{\partial w_l}, \dots, \frac{\partial E_i(\vec{w})}{\partial w_n} \right\rangle$$

$$\frac{\partial E_i}{\partial w_l} = (t^{(i)} - \Phi\left(\sum_l w_l x_l^{(i)}\right)) \cdot \frac{\partial (t^{(i)} - \Phi\left(\sum_l w_l x_l^{(i)}\right))}{\partial w_l} = -(t^{(i)} - o^{(i)}) \cdot \Phi'\left(\sum_l w_l x_l^{(i)}\right) \cdot x_l^{(i)}$$

Άρα: $\nabla E_i(\vec{w}) = -(t^{(i)} - o^{(i)}) \cdot \Phi'\left(\sum_l w_l x_l^{(i)}\right) \cdot \left\langle x_1^{(i)}, \dots, x_n^{(i)} \right\rangle$

$$= -(t^{(i)} - o^{(i)}) \cdot \Phi'(\vec{w} \cdot \vec{x}^{(i)}) \cdot \vec{x}^{(i)}$$

Κανόνας ενημέρωσης βαρών:

$$\vec{w} \leftarrow \vec{w} - \eta \cdot \nabla E_i(\vec{w}) = \vec{w} + \eta \cdot (t^{(i)} - o^{(i)}) \cdot \Phi'(\vec{w} \cdot \vec{x}^{(i)}) \cdot \vec{x}^{(i)}$$

Οπότε: $w_l \leftarrow w_l + \eta \cdot (t^{(i)} - o^{(i)}) \cdot \Phi'(\vec{w} \cdot \vec{x}^{(i)}) \cdot x_l^{(i)}$

Perceptron με σιγμοειδή ή άλλη Φ

- Οπότε αν χρησιμοποιούμε συνάρτηση ενεργοποίησης $\Phi(S)$:

$$w_l \leftarrow w_l + \eta \cdot \Phi'(\sum_l w_l x_l^{(i)}) \cdot (t^{(i)} - \Phi\left(\sum_l w_l x_l^{(i)}\right)) \cdot x_l^{(i)}$$

- Για $\Phi(S) = \dfrac{1}{1+e^{-S}}$, $\Phi'(S) = \Phi(S) \cdot (1 - \Phi(S))$.
- Και αφού $o^{(i)} = \Phi\left(\sum_l w_l x_l^{(i)}\right)$, ο κανόνας ενημέρωσης των βαρών γίνεται:

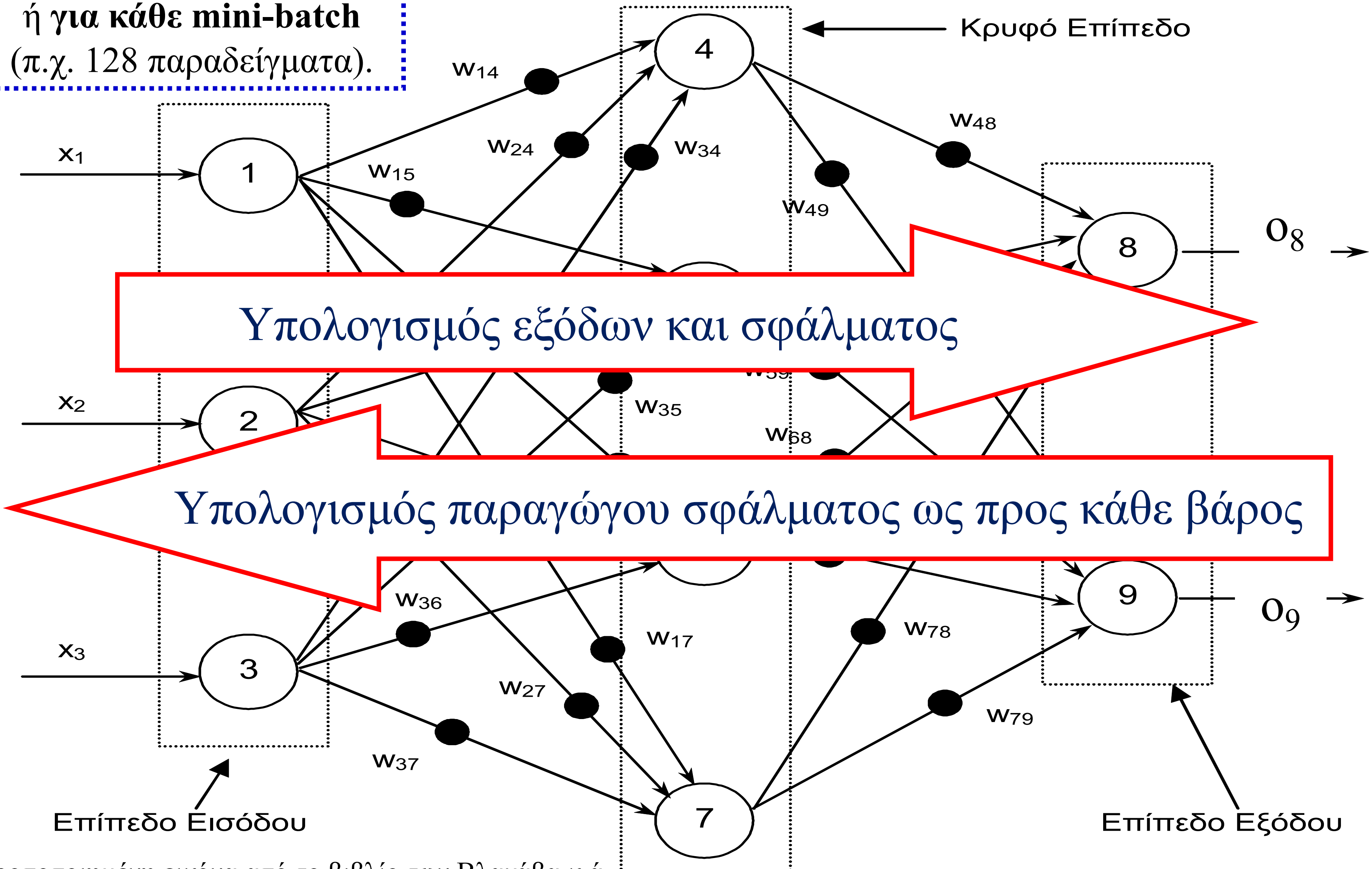
$$w_l \leftarrow w_l + \eta \cdot o^{(i)} \cdot (1 - o^{(i)}) \cdot (t^{(i)} - o^{(i)}) \cdot x_l^{(i)}$$

Γραμμική διαχωρισιμότητα

- Η εκπαίδευση του Perceptron **συγκλίνει**, αν τα παραδείγματα εκπαίδευσης είναι **γραμμικά διαχωρίσιμα** (και το η είναι αρκετά μικρό).
  - Υπάρχουν βελτιώσεις που εξασφαλίζουν ότι συγκλίνει πάντα, αλλά σε **προσεγγιστική** λύση.
- **Γραμμικά διαχωρίσιμα** παραδείγματα σημαίνει:
  - Στην περίπτωση δύο ιδιοτήτων (x_1, x_2), να υπάρχει **ευθεία** που να **διαχωρίζει** τα αρνητικά από τα θετικά παραδείγματα.
  - Γενικότερα στην περίπτωση m ιδιοτήτων, να υπάρχει ένα **υπερ-επίπεδο** που να διαχωρίζει τα αρνητικά από τα θετικά παραδείγματα.

Αλγόριθμος ανάστροφης μετάδοσης

Για κάθε ένα παράδειγμα
ή για κάθε mini-batch
(π.χ. 128 παραδείγματα).



Τροποποιημένη εικόνα από το βιβλίο των Βλαχάβα κ.ά.

Αλγόριθμος ανάστροφης μετάδοσης (back-propagation)

- Χρησιμοποιείται για την εκπαίδευση ΤΝΔ **απλής τροφοδότησης** με **πολλαπλά επίπεδα**.
  - Μπορεί να χρησιμοποιηθεί και για την εκπαίδευση **ΤΝΔ με ανατροφοδότηση (RNNs)** (επόμενες διαλέξεις).
  - Η προηγούμενη διαφάνεια δείχνει μόνο ένα κρυφό επίπεδο αλλά ο αλγόριθμος **δουλεύει και για πολλαπλά κρυφά επίπεδα**.
  - Σε προβλήματα **παλινδρόμησης**, το **τοπικό σφάλμα** (σφάλμα στο τρέχον παράδειγμα) είναι συνήθως: $E = \frac{1}{n}\sum_{i=1}^{n}(o_i - y_i)^2$
- Συμβολισμός:
  - **Απόκριση** του ΤΝΔ σε ένα παράδειγμα: $\langle o_1, \dots, o_n \rangle$
  - Σωστή απόκριση: $\langle y_1, \dots, y_n \rangle$
  - **Βάρος** σύνδεσης από νευρώνα *i* σε νευρώνα *j*: $\boldsymbol{w_{ij}}$
  - **Είσοδος** από το νευρώνα *i* στο νευρώνα *j*: $\boldsymbol{x_{ij}}$

Αλγόριθμος ανάστροφης μετάδοσης

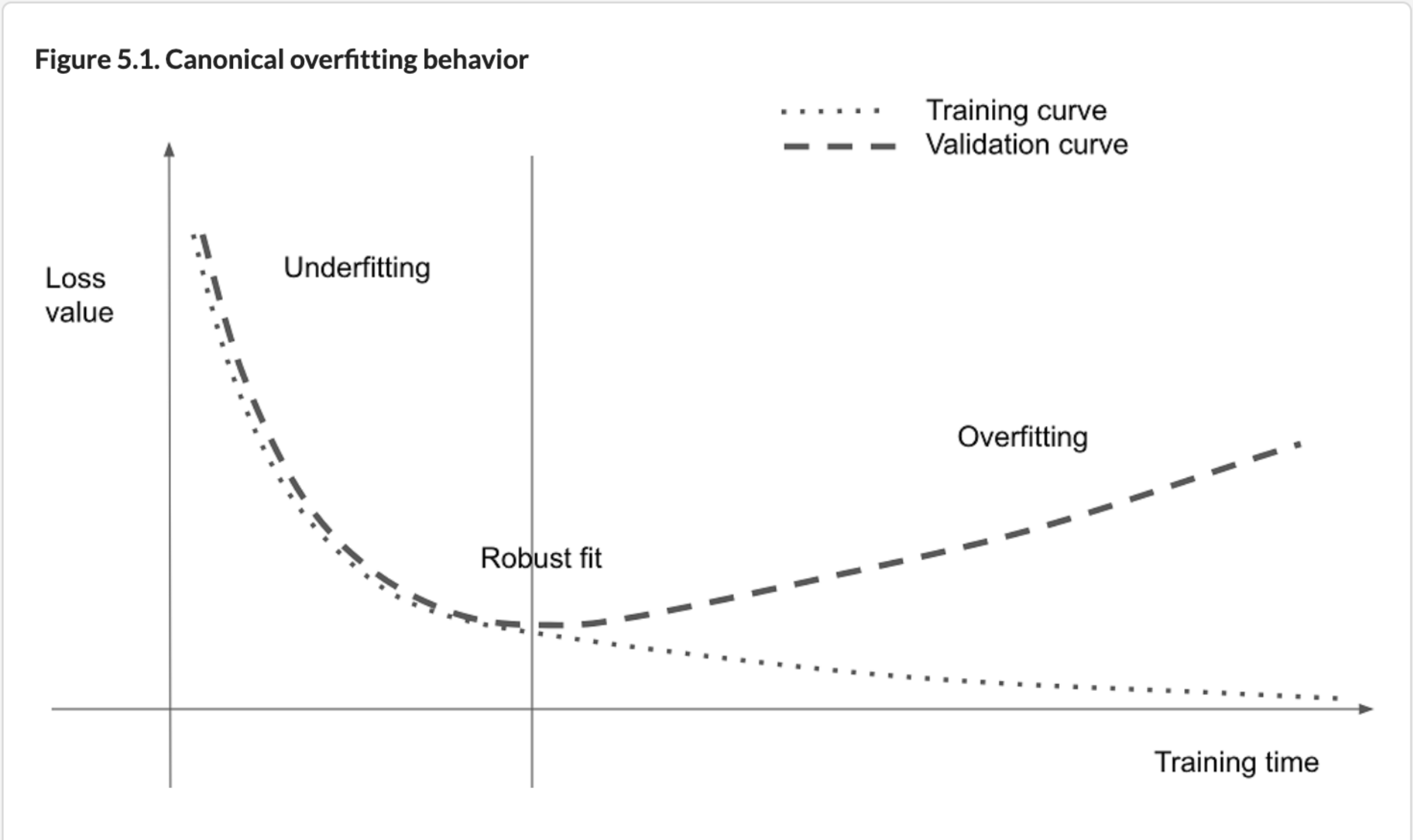
- **Αρχικοποίησε** όλα τα βάρη σε μικρές **τυχαίες** τιμές.
- Όσο **δεν ικανοποιείται** η **συνθήκη τερματισμού**:
  - **Νέα εποχή**: Για **κάθε παράδειγμα** εκπαίδευσης:
    - Υπολόγισε (από αριστερά προς τα δεξιά) τις **αποκρίσεις στο επίπεδο εξόδου**.
    - Υπολόγισε το **σφάλμα** E στο επίπεδο εξόδου.
    - Για **κάθε βάρος** w_{ij}, υπολόγισε (από δεξιά προς τα αριστερά) την **παράγωγο** $\frac{\partial E}{\partial w_{ij}}$ (δηλ. πώς επηρεάζει το E).
    - **Ενημέρωσε κάθε βάρος** w_{ij}:

$$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}}$$

Θα δούμε πώς στην επόμενη διάλεξη.

Για όλα τα βάρη μαζί:
$W \leftarrow W - \eta \cdot \nabla_W E$

Σφάλμα συναρτήσει των <u>εποχών</u>



Σχήμα από το **προτεινόμενο βιβλίο «Deep Learning with Python»** του F. Chollet, Manning Publications, 2η έκδοση. Η 1η έκδοση παρέχεται δωρεάν.
https://www.manning.com/books/deep-learning-with-python
https://www.manning.com/books/deep-learning-with-python-second-edition

Ανάστροφη μετάδοση: περισσότερα

- Συνθήκη **τερματισμού**:
  - υπερβήκαμε έναν **μέγιστο αριθμό εποχών**
  - ή το **συνολικό σφάλμα εποχής** υπολογισμένο σε (διαφορετικά) παραδείγματα **επικύρωσης** είναι εντός επιθυμητών ορίων ή έχει αρχίσει πλέον να χειροτερεύει (λόγω υπερ-εφαρμογής).
- **Δεν εγγυάται** ότι θα βρει τη **βέλτιστη λύση**:
  - Ουσιαστικά αλγόριθμος **αναρρίχησης λόφων**.
  - Κίνδυνος εγκλωβισμού σε **τοπικό μέγιστο**.
  - Για να αποφύγουμε τα τοπικά μέγιστα, εκπαιδεύουμε π.χ. **πολλές φορές** με **διαφορετικά αρχικά βάρη**. Επιλέγουμε τα **καλύτερα τελικά βάρη**, αξιολογώντας σε **δεδομένα ανάπτυξης/επικύρωσης** (όχι στα δεδομένα αξιολόγησης!).

Το σκεφτόμαστε πια συνήθως ως αλγόριθμο, αλλά ήταν μηχάνημα!

The perceptron is a machine

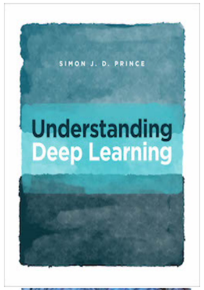
Από την παρουσίαση «Multilayer Neural Networks» του L. Bottou στο Deep Learning Summer School 2015 (βλ. http://videolectures.net/deeplearning2015_montreal/).

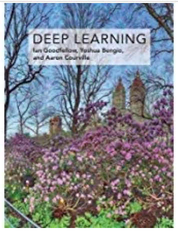
Βιβλιογραφία

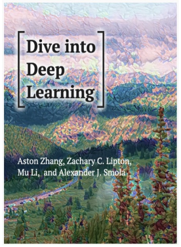
- Russel & Norvig (4η έκδοση, ελληνική μετάφραση): κεφάλαιο 21 ως και ενότητα 21.2.3.
  - Θα καλύψουμε αρκετές άλλες ενότητες αυτού του κεφαλαίου σε επόμενες διαλέξεις.
- Βλαχάβας κ.ά: ενότητες 18.10.1, 19.1–19.5.
  - Όσοι ενδιαφέρονται μπορούν να διαβάσουν προαιρετικά και τις υπόλοιπες ενότητες του κεφαλαίου 19.
  - Η ενότητα 19.9.2 (RNNs) θα καλυφθεί σε επόμενη διάλεξη.
- Αν σας λείπει μαθηματικό υπόβαθρο, ιδιαίτερα στις ασκήσεις μελέτης, συμβουλευτείτε το:
  - T. Parr και J. Howard, «The Matrix Calculus You Need for Deep Learning». https://explained.ai/matrix-calculus/

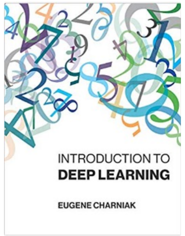
Πρόσθετη προτεινόμενη βιβλιογραφία

- Τα τελευταία χρόνια η βαθιά μάθηση (deep learning) έχει επιδείξει εξαιρετικά αποτελέσματα σε πολλές εφαρμογές.
  - Το βιβλίο «Understanding Deep Learning» του S.J.D. Prince, MIT Press, διατίθεται δωρεάν: https://udlbook.github.io/udlbook/
  - Το βιβλίο «Deep Learning» των I. Goodfellow κ.ά., MIT Press διατίθεται δωρεάν: http://www.deeplearningbook.org/
  - Το διαδραστικό βιβλίο «Dive into Deep Learning» των Zhang κ.ά. διατίθεται δωρεάν: https://d2l.ai/
  - Το βιβλίο «Introduction to Deep Learning» του E. Charniak, MIT Press υπάρχει στη βιβλιοθήκη του ΟΠΑ.

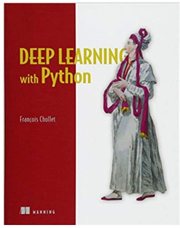








Πρόσθετη προτεινόμενη βιβλιογραφία



- Ως πιο πρακτική εισαγωγή, προτείνεται ιδιαίτερα το βιβλίο του F. Chollet «Deep Learning in Python», Manning Publications.
  - Καλύπτει σε μεγάλο βαθμό την ύλη αυτών των διαφανειών (βλ. κεφ. 1–4). Αποτελεί και εισαγωγή στο Keras (πλέον μέρος της βιβλιοθήκης Tensorflow), που θα χρησιμοποιηθεί στα φροντιστήρια του μαθήματος.
  - Η 1η έκδοση (2017), που μας αρκεί, παρέχεται δωρεάν: https://www.manning.com/books/deep-learning-with-python
  - Η 2η έκδοση (2022) απαιτεί πληρωμή αλλά την αξίζει!
- Η PyTorch είναι επίσης εξαιρετικά δημοφιλής βιβλιοθήκη ανάπτυξης νευρωνικών δικτύων.
  - Εισαγωγικά μαθήματα PyTorch: https://pytorch.org/tutorials/