ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Αλληλεπίδραση Ανθρώπου–Υπολογιστή

Β5. Επεξεργασία φυσικής γλώσσας και εικόνας με πολυεπίπεδα Perceptron

(2023–24)

Ίων Ανδρουτσόπουλος

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

Οι διαφάνειες αυτές βασίζονται εν μέρει σε ύλη των βιβλίων:

- *Artificial Intelligence – A Modern Approach* των S. Russel και P. Norvig, 2η έκδοση, Prentice Hall, 2003,
- *Τεχνητή Νοημοσύνη* των Βλαχάβα κ.ά., 3η έκδοση, Β. Γκιούρδας Εκδοτική, 2006,

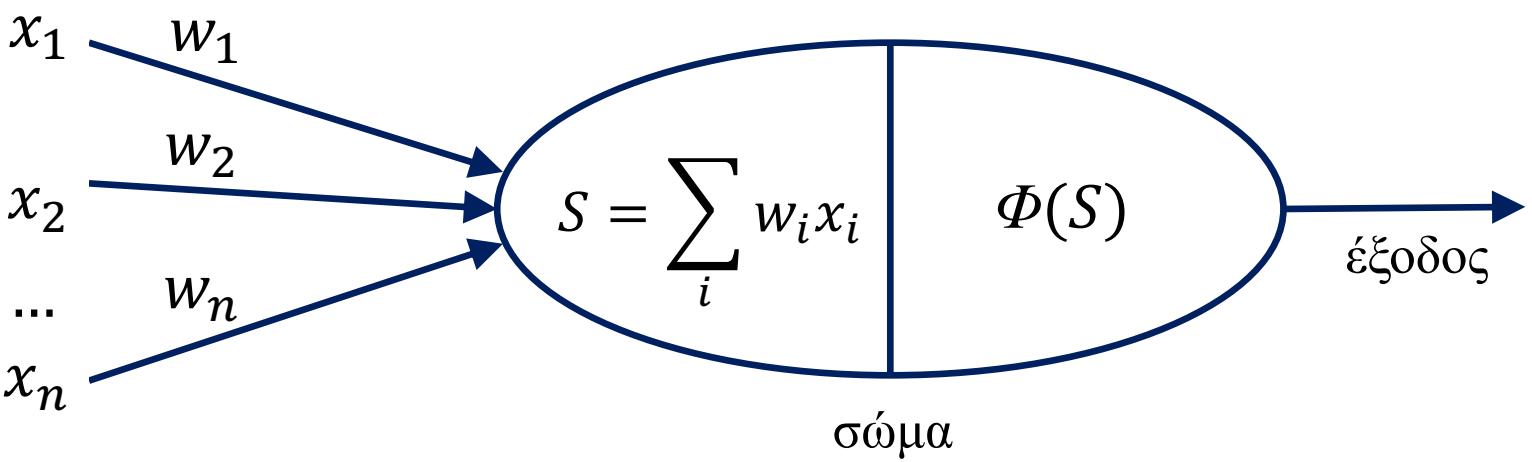
Κάποια από τα σχήματα των διαφανειών προέρχονται από τα παραπάνω βιβλία (βλ. σημειώσεις στις παρακάτω διαφάνειες).

Τι θα ακούσετε σήμερα

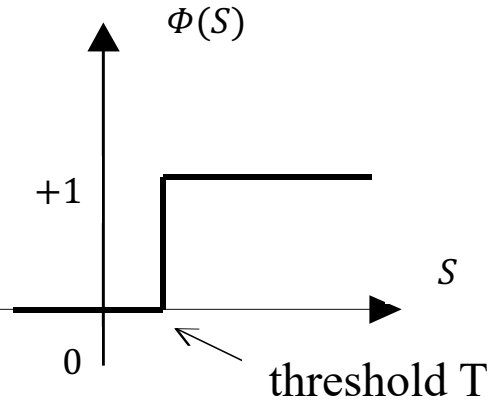
- Νευρωνικά δίκτυα απλής τροφοδότησης (feed-forward neural networks)
  - Πολυεπίπεδο Perceptron (Multi-layer Perceptron, MLP).
  - Αλγόριθμος ανάστροφης μετάδοσης (back-propagation), γράφος υπολογισμού, αυτόματος υπολογισμός παραγώγων.
  - Dropout, batch/layer normalization.
- Εφαρμογές σε προβλήματα κατηγοριοποίησης και παλινδρόμησης φυσικής γλώσσας και εικόνας.

Τεχνητά νευρωνικά δίκτυα

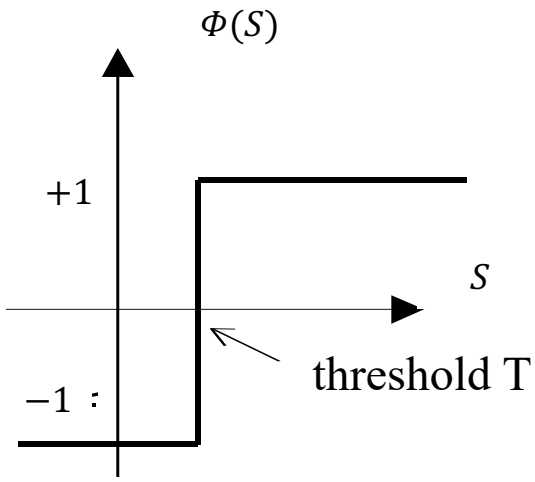
- **Τεχνητός νευρώνας**:
  - **Είσοδοι**: πραγματικές τιμές.
  - **Βάρη εισόδων**: πραγματικές τιμές (χονδρικά συνάψεις).
  - **Σώμα**: υπολογίζει το **ζυγισμένο άθροισμα** των εισόδων, κατόπιν εφαρμόζει τη **συνάρτηση ενεργοποίησης** στο ζυγισμένο άθροισμα.



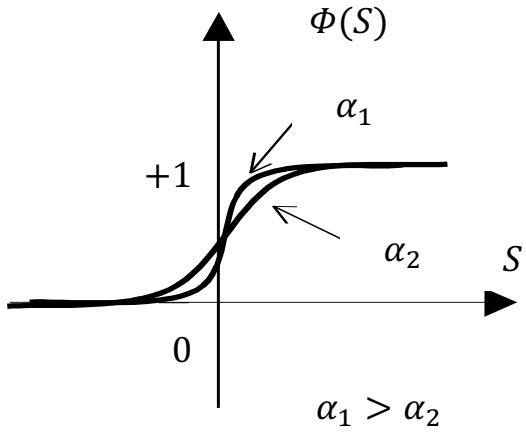
Συναρτήσεις ενεργοποίησης



step function

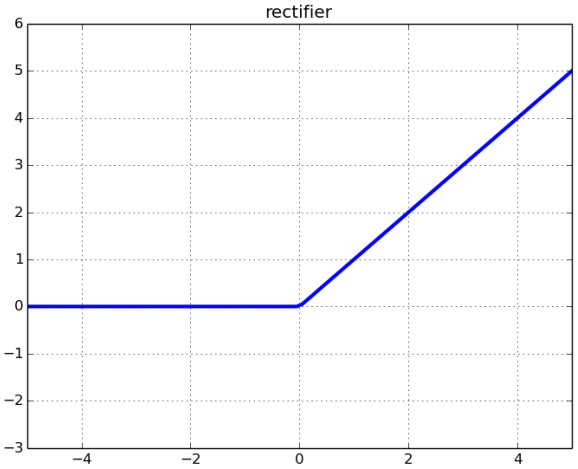


sign function



sigmoid
(logistic function)

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



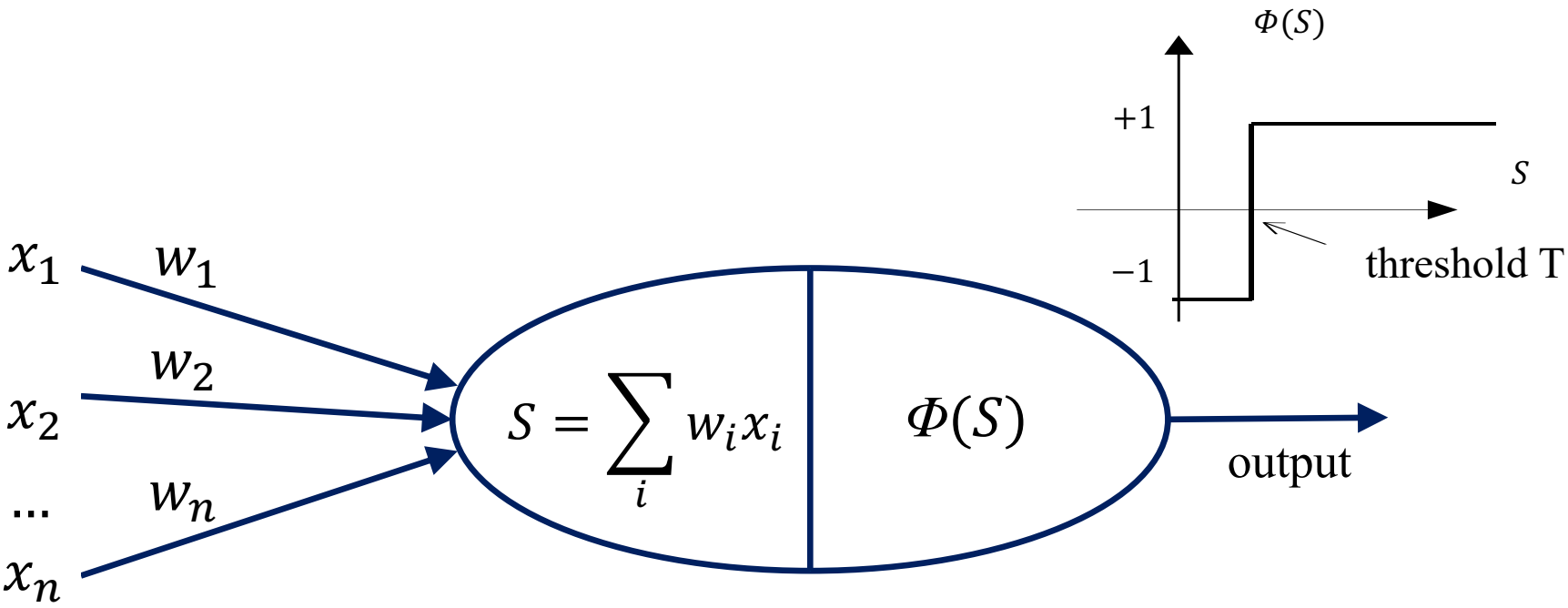
Rectified Linear Unit (ReLU)

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).

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

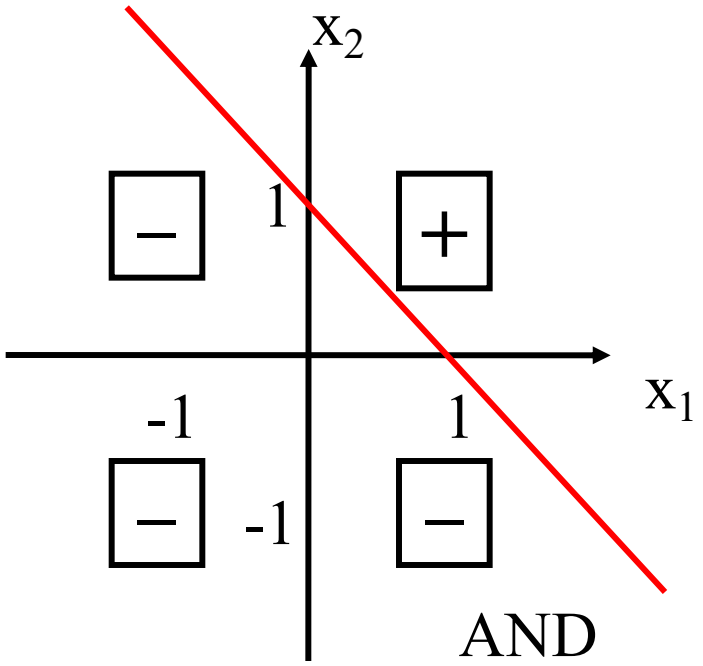
Perceptron

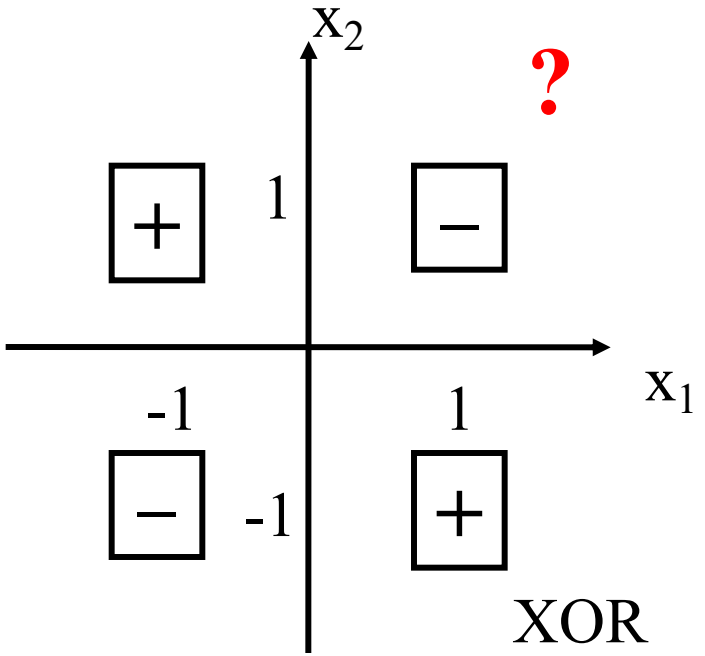
- Μόνο **ένας νευρώνας**, με συνάρτηση ενεργοποίησης **προσήμου**.
  - **Ισοδύναμα**, με **βηματική** (step) συνάρτηση ενεργοποίησης.
  - Το Perceptron μπορεί να **γενικευθεί** (έγινε αργότερα), ώστε να χρησιμοποιεί **σιγμοειδή ή άλλη** συνάρτηση ενεργοποίησης.
  - Ένα **μεμονωμένο Perceptron** είναι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



Original images from the book of Vlahavas et al.

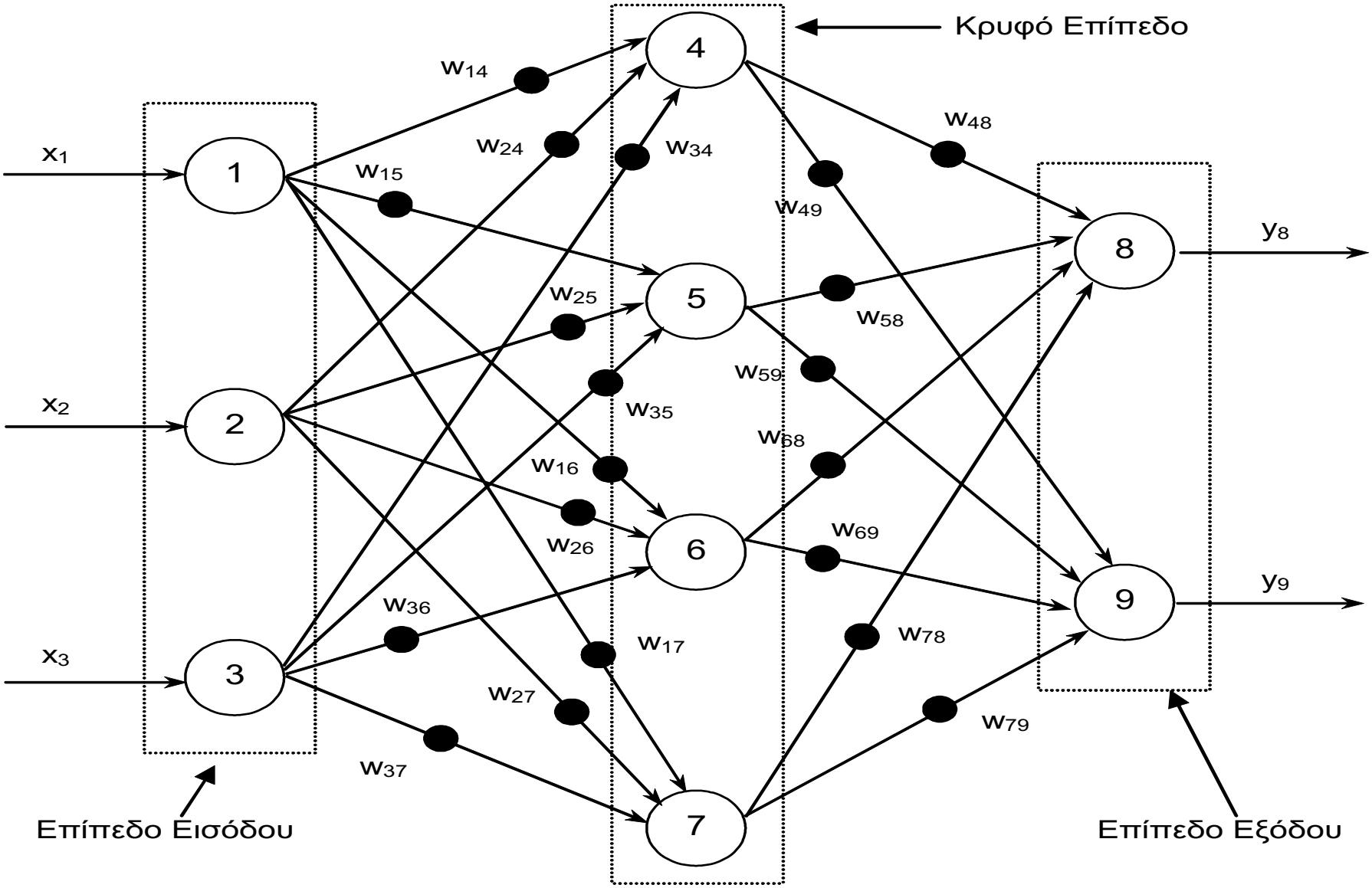
Γραμμική διαχωρισιμότητα





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

Πολυ-επίπεδο Perceptron (MLP)

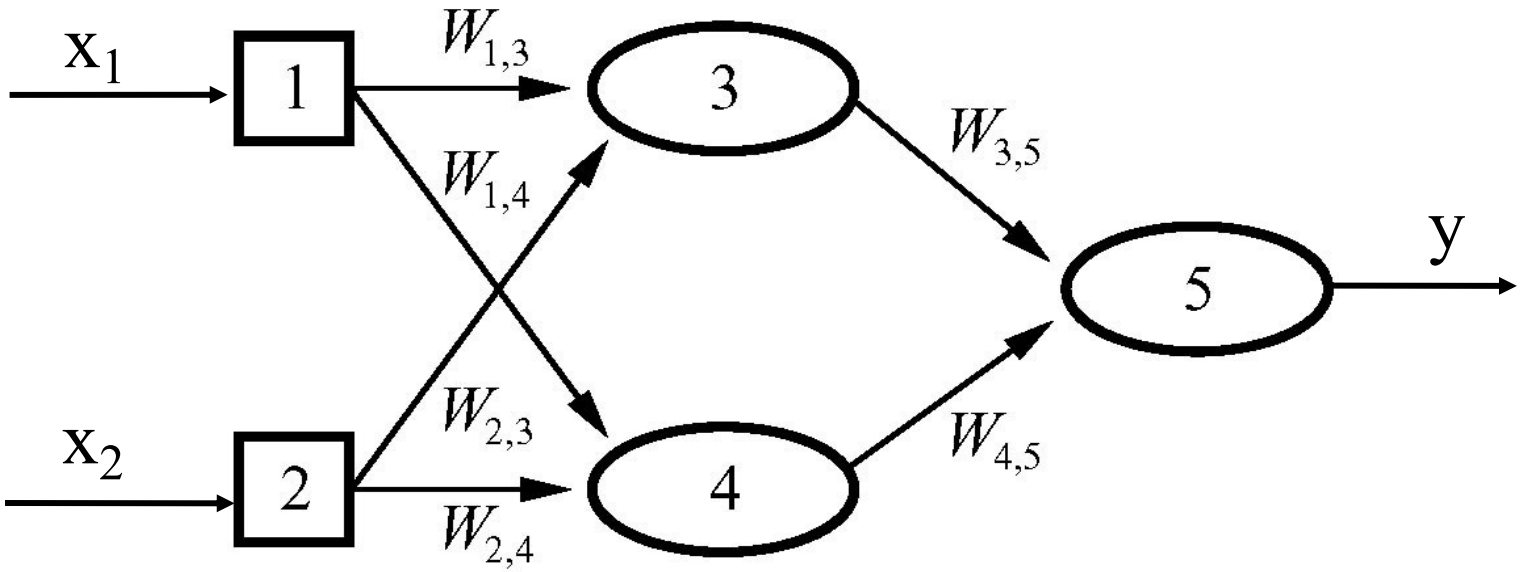


Εικόνα από το βιβλίο των Βλαχάβα κ.ά.

Πολυ-επίπεδο Perceptron (MLP)

- Νευρώνες **εισόδου**: συχνά χωρίς ουσιαστικούς υπολογισμούς.
  - Π.χ. μόνο μία είσοδος στον καθένα, μοναδιαίο βάρος, $\Phi(S) = S$.
  - 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



Εικόνα από το βιβλίο των Russel & Norvig.

- $y = \Phi(w_{3,5} \cdot x_{3,5} + w_{4,5} \cdot x_{4,5}) =$

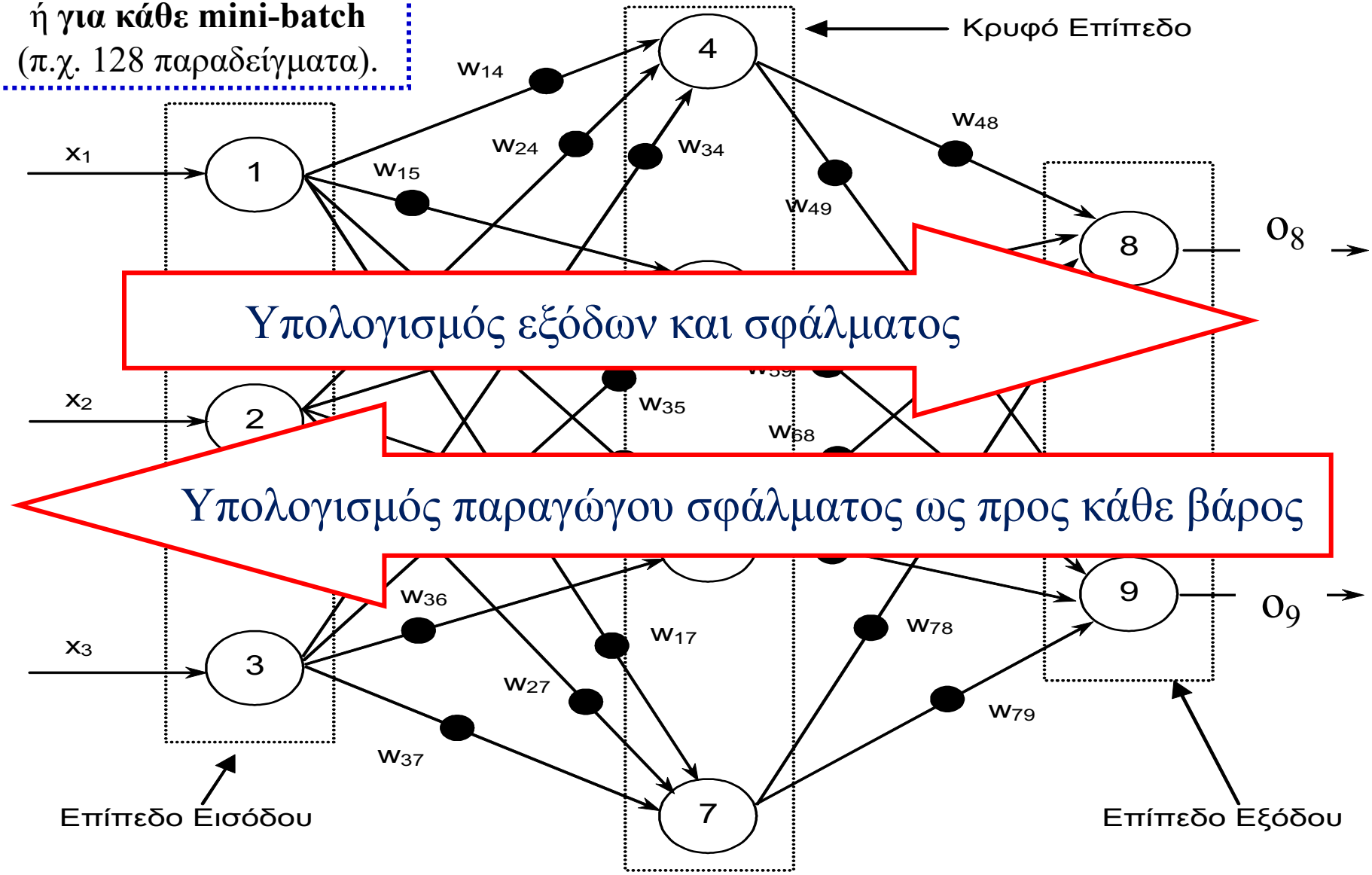
  $\Phi(w_{3,5} \cdot \Phi(w_{1,3} \cdot x_1 + w_{2,3} \cdot x_2) + w_{4,5} \cdot \Phi(w_{1,4} \cdot x_1 + w_{2,4} \cdot x_2))$
- Μεταβάλλοντας τα **βάρη** μαθαίνουμε **διαφορετική συνάρτηση**.
- **Ποιες συναρτήσεις** μπορούμε να μάθουμε; Εξαρτάται από τη Φ, το **πλήθος των επιπέδων**, το πλήθος **νευρώνων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

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: $\langle o_1, \ldots, o_n \rangle$
  - Σωστή απόκριση: $\langle y_1, \ldots, y_n \rangle$
  - **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}:

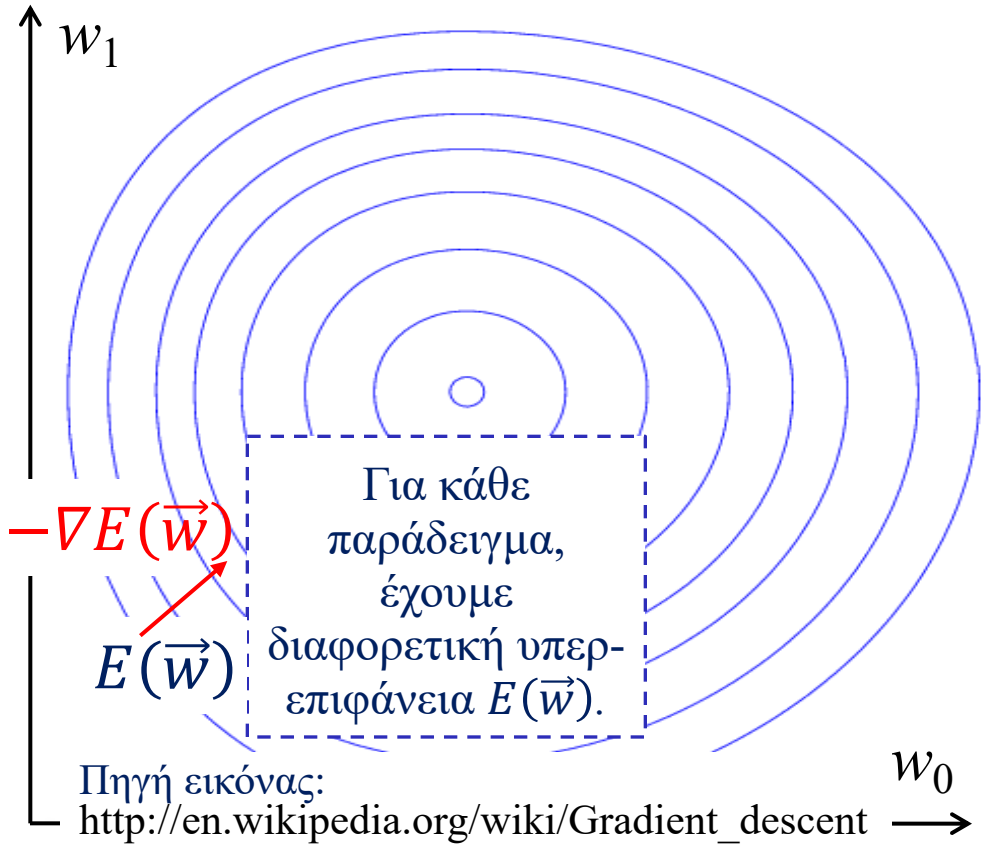
$$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}}$$

Για όλα τα βάρη μαζί:
$\vec{w} \leftarrow \vec{w} - \eta \cdot \nabla_W E$

Κανόνας ενημέρωσης βαρών

- Σφάλμα στο **<u>τρέχον</u>** παράδειγμα: $E = \frac{1}{n}\sum_{i=1}^{n}(o_i - y_i)^2$

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

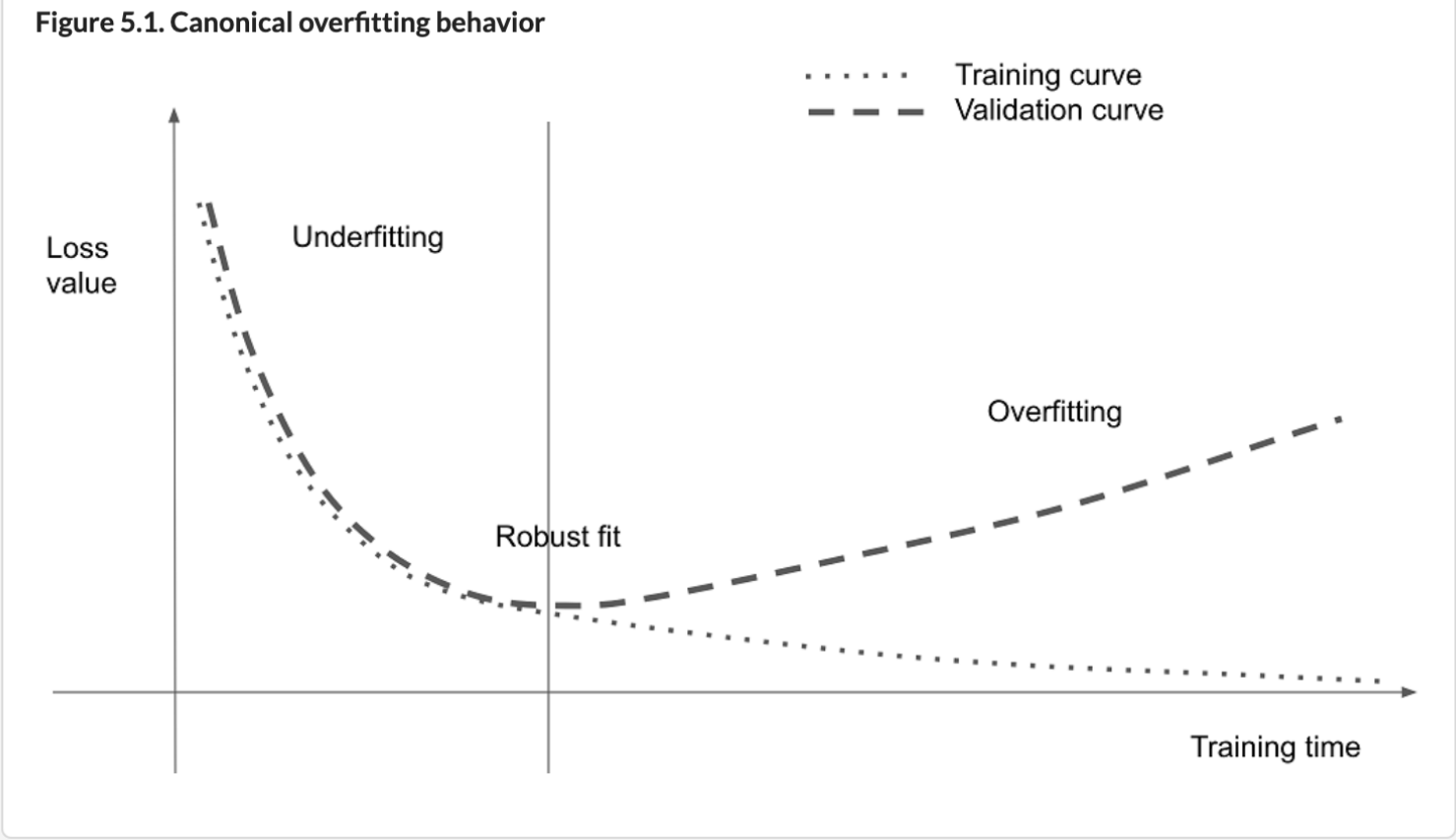


Πηγή εικόνας:
http://en.wikipedia.org/wiki/Gradient_descent

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

Τροποποιούμε τα βάρη $\vec{w}$ κατά η προς την κατεύθυνση που προκαλεί τη μεγαλύτερη μείωση του υψομέτρου E:
$\vec{w} \leftarrow \vec{w} - \eta \cdot \nabla E(\vec{w})$
Στοχαστική κατάβαση κλίσης.

Συνολικό σφάλμα συναρτήσει των εποχών



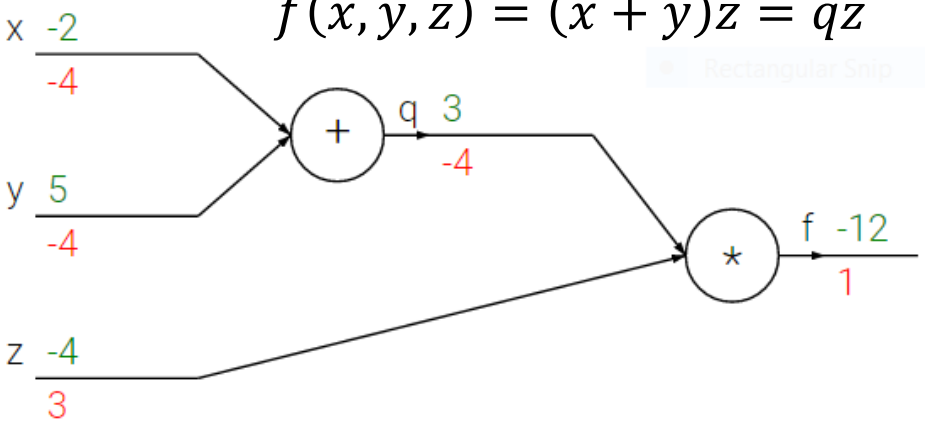
Σχήμα από το **προτεινόμενο βιβλίο «Deep Learning with Python»** του F. Chollet, Manning Publications, 2η έκδοση (υπό προετοιμασία). Η 1η έκδοση παρέχεται δωρεάν.
https://www.manning.com/books/deep-learning-with-python
https://www.manning.com/books/deep-learning-with-python-second-edition

Ανάστροφη μετάδοση: περισσότερα

- Συνθήκη **τερματισμού**:
  - υπερβήκαμε έναν **μέγιστο αριθμό εποχών**
  - ή το **συνολικό σφάλμα εποχής** υπολογισμένο σε (διαφορετικά) παραδείγματα **ανάπτυξης** 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!).

Παράδειγμα γράφου υπολογισμού

$$f(x, y, z) = (x + y)z = qz$$



Παράδειγμα και σχήμα από το μάθημα "CNNs for Visual Recognition" (2016, F.-F. Li, A. Karpathy, J. Johnson) του Πανεπιστημίου Stanford. http://cs231n.github.io/optimization-2/

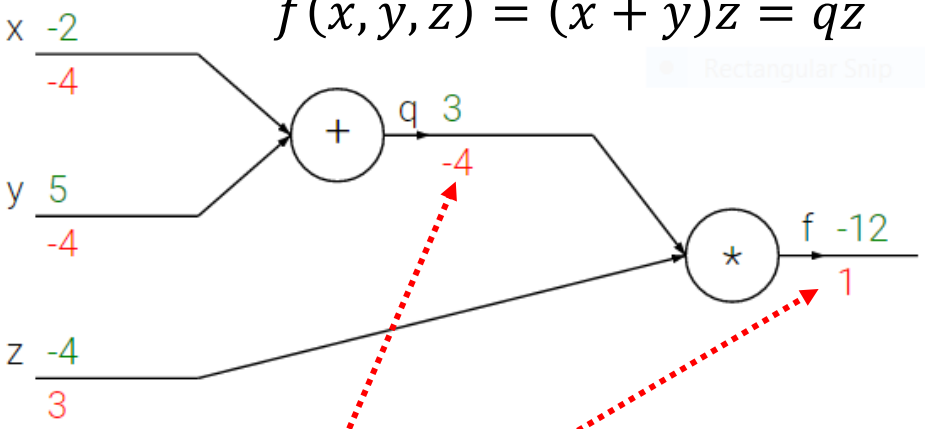
- **Προς τα εμπρός**: $\langle x, y, z \rangle = \langle -2{,}5, -4 \rangle, q = 3, f = -12$
- Ας υποθέσουμε ότι θέλουμε να **ελαχιστοποιήσουμε την *f*** χρησιμοποιώντας **στοχαστική κατάβαση κλίσης** (SGD).
  - Σε ένα πιο ρεαλιστικό σενάριο, η $\boldsymbol{f}$ θα ήταν μια **συνάρτηση σφάλματος** και το $\langle \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \rangle$ το **διάνυσμα βαρών**.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \leftarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \eta \cdot \nabla f(x, y, z) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

Χρειαζόμαστε: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Ανάστροφη μετάδοση στο γράφο

$f(x, y, z) = (x + y)z = qz$

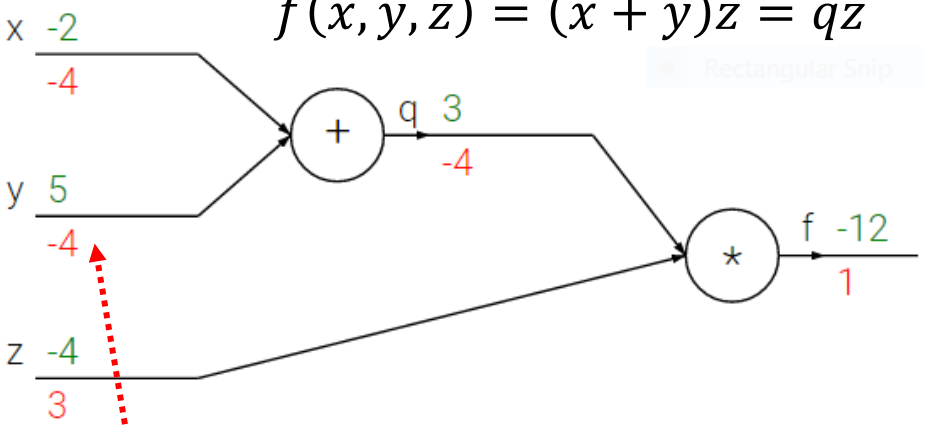


Παράδειγμα και σχήμα από το μάθημα "CNNs for Visual Recognition" (2016, F.-F. Li, A. Karpathy, J. Johnson) του Πανεπιστημίου Stanford. http://cs231n.github.io/optimization-2/

- **Ανάστροφη μετάδοση**: Υπολογίζουμε **παραγώγους** από **δεξιά προς αριστερά**.
  - $\frac{\partial f}{\partial f} = 1$ εξ ορισμού.
  - $\frac{\partial f}{\partial q} = z$. Και για το συγκεκριμένο διάνυσμα εισόδου $\langle x, y, z \rangle$, έχουμε $z = -4$.
  - **Κατά τους προς τα εμπρός υπολογισμούς**, πρέπει να **αποθηκεύουμε τις εξόδους όλων των κόμβων** (π.χ., εδώ χρειαστήκαμε την τιμή του z).

Ανάστροφη μετάδοση στο γράφο

$$f(x, y, z) = (x + y)z = qz$$

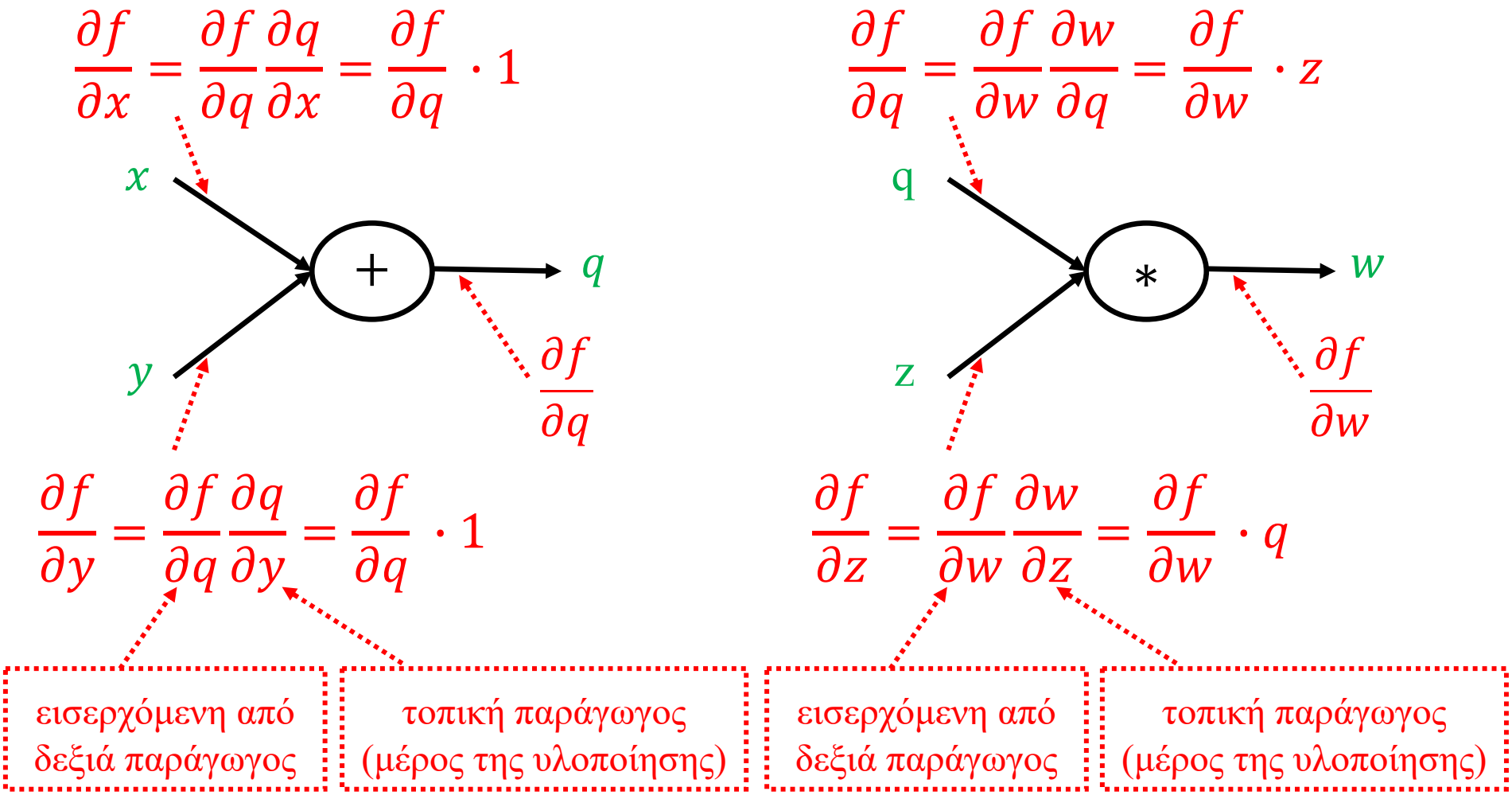


Παράδειγμα και σχήμα από το μάθημα "CNNs for Visual Recognition" (2016, F.-F. Li, A. Karpathy, J. Johnson) του Πανεπιστημίου Stanford. http://cs231n.github.io/optimization-2/

- **Ανάστροφη μετάδοση**:
  - $\frac{\partial f}{\partial f} = 1$ εξ ορισμού.
  - $\frac{\partial f}{\partial q} = z$. Και για το συγκεκριμένο $\langle x, y, z \rangle$, $z = -4$.
  - $\frac{\partial f}{\partial z} = q$. Και για το συγκεκριμένο $\langle x, y, z \rangle$, $q = 3$.
  - $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y} = \frac{\partial f}{\partial q} \cdot 1$. Και εδώ $\frac{\partial f}{\partial q} = -4$.
  - $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} = \frac{\partial f}{\partial q} \cdot 1$. Και εδώ $\frac{\partial f}{\partial q} = -4$.

εισερχόμενη παράγωγος | τοπική παράγωγος

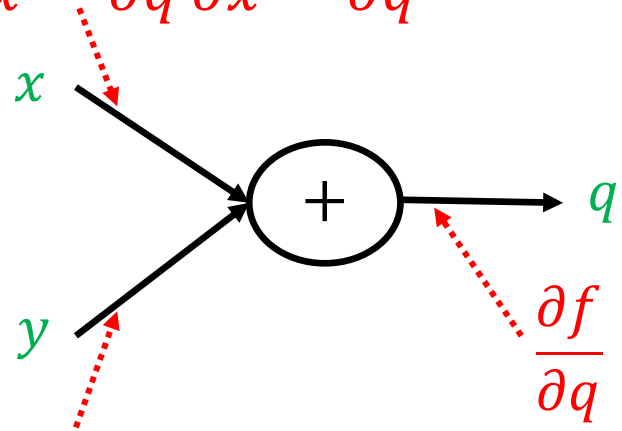
Υλοποιήσεις πυλών που μπορούμε να συνδυάσουμε



- Μπορούμε να υλοποιήσουμε τις **πύλες** ως **τάξεις** (π.χ., σε Java, C++ ή Python).
  - Με **μεθόδους προς τα εμπρός** και **ανάστροφης μετάδοσης**.

Plug-and-play gates

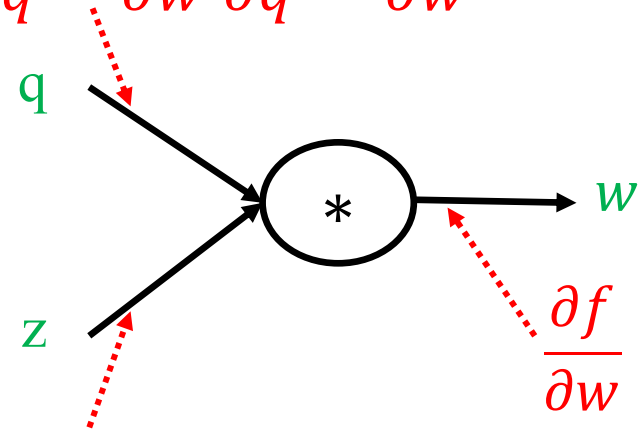
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} = \frac{\partial f}{\partial q} \cdot 1$$



$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y} = \frac{\partial f}{\partial q} \cdot 1$$

```
class PlusGate:
  forward(x, y):
    return x+y
  backward(∂f/∂q):
    return <∂f/∂q, ∂f/∂q>
```

$$\frac{\partial f}{\partial q} = \frac{\partial f}{\partial w}\frac{\partial w}{\partial q} = \frac{\partial f}{\partial w} \cdot z$$

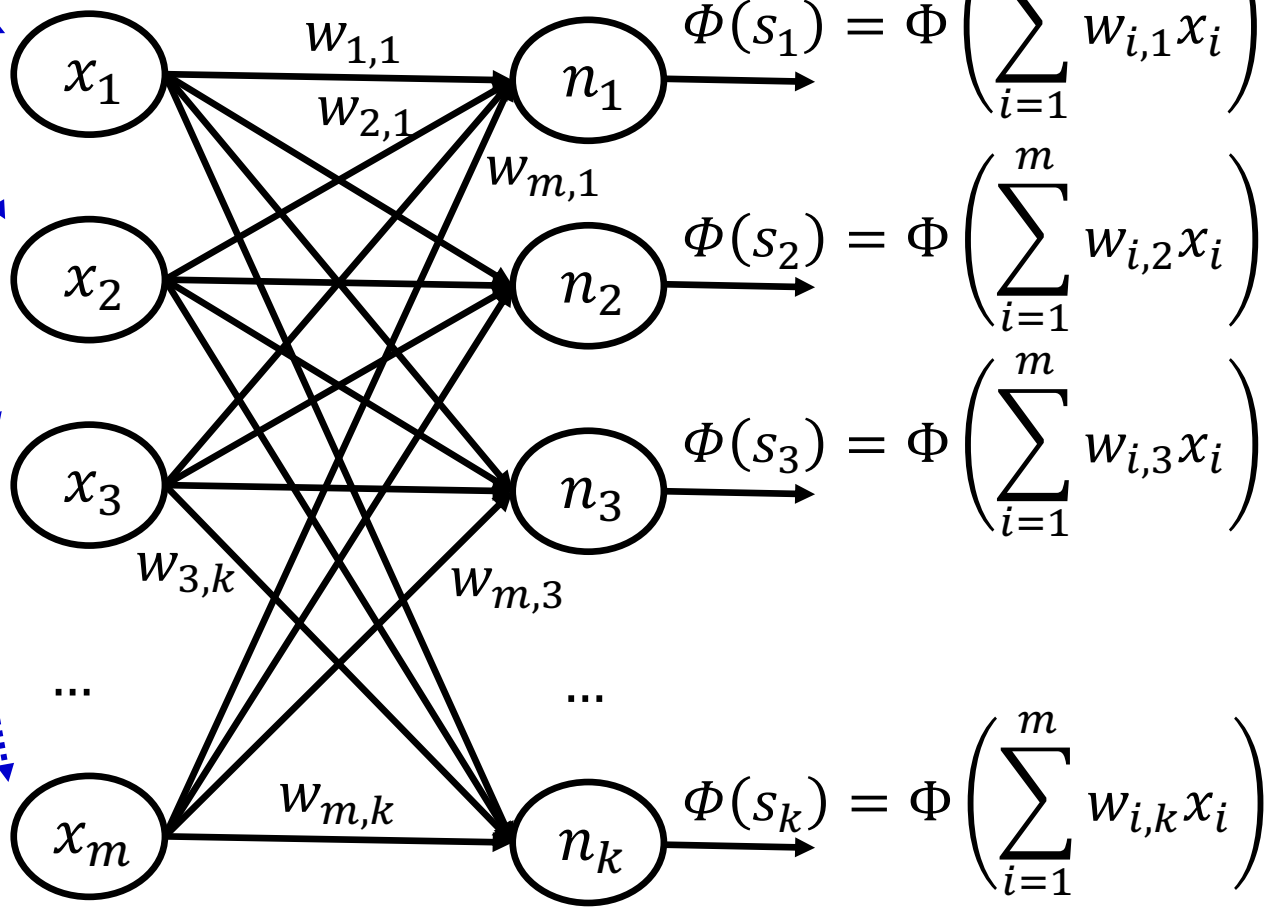


$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial w}\frac{\partial w}{\partial z} = \frac{\partial f}{\partial w} \cdot q$$

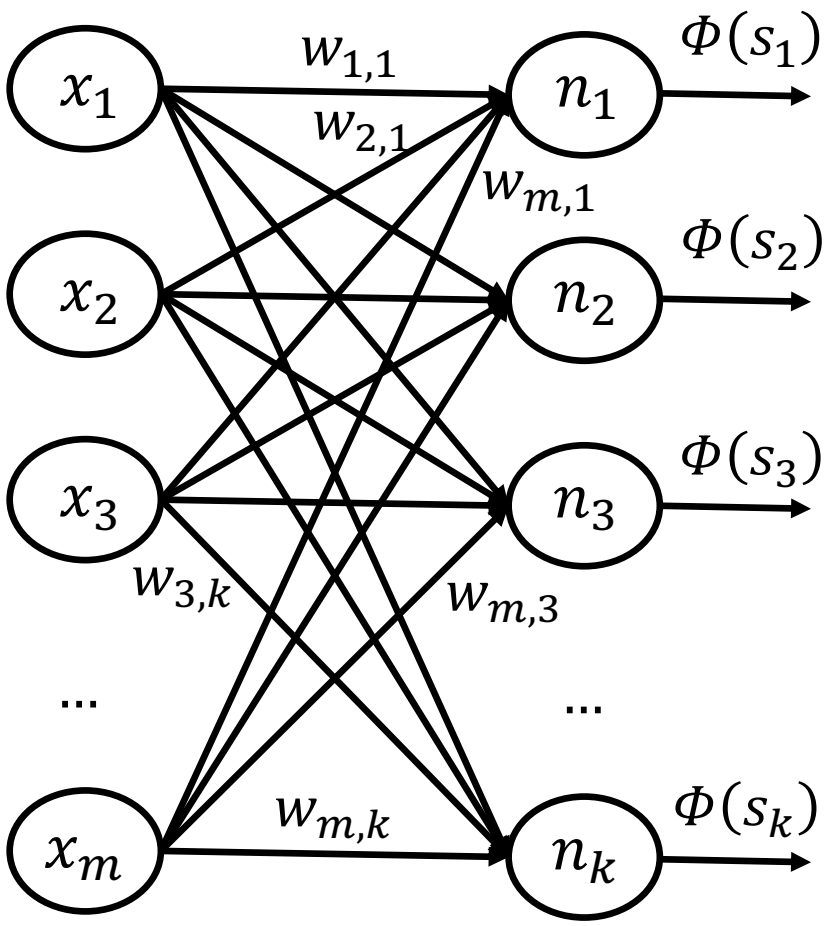
```
class StarGate:
  forward(q, z):
    return q * z
  backward(∂f/∂w):
    return <∂f/∂w · z, ∂f/∂w · q>
```

Πιο συμπαγής συμβολισμός

Διάνυσμα εισόδου (π.χ. ενδείξεις αισθητήρων ή συχνότητες εμφάνισης λέξεων ή τιμές pixels)



Πιο συμπαγής συμβολισμός



$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \dots \\ s_k \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{2,1}x_2 + \cdots + w_{m,1}x_m \\ w_{1,2}x_1 + w_{2,2}x_2 + \cdots + w_{m,2}x_m \\ w_{1,3}x_1 + w_{2,3}x_2 + \cdots + w_{m,3}x_m \\ \dots \\ w_{1,k}x_1 + w_{2,k}x_2 + \cdots + w_{m,k}x_m \end{bmatrix}$$

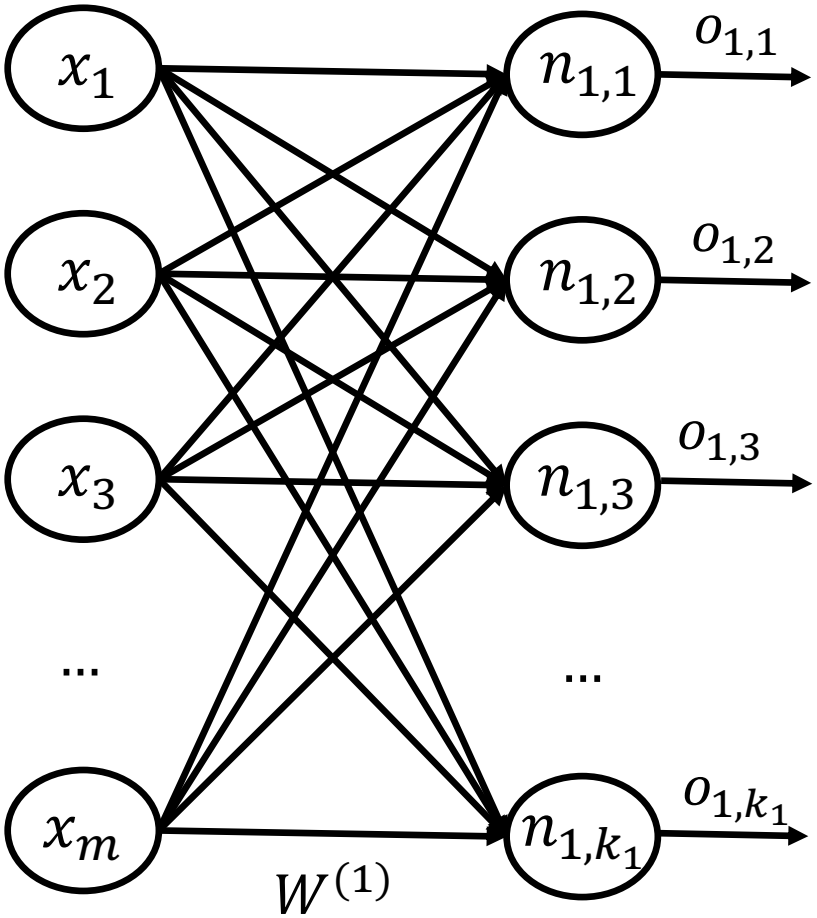
$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \dots \\ s_k \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{2,1} & \dots & w_{m,1} \\ w_{1,2} & w_{2,2} & \dots & w_{m,2} \\ w_{1,3} & w_{2,3} & \dots & w_{m,3} \\ \dots & \dots & \dots & \dots \\ w_{1,k} & w_{2,k} & \dots & w_{m,k} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_m \end{bmatrix}$$

$$\vec{s} = W\vec{x}$$

Μαθαίνουμε τον $\boldsymbol{W}$ με **ανάστροφη μετάδοση**.

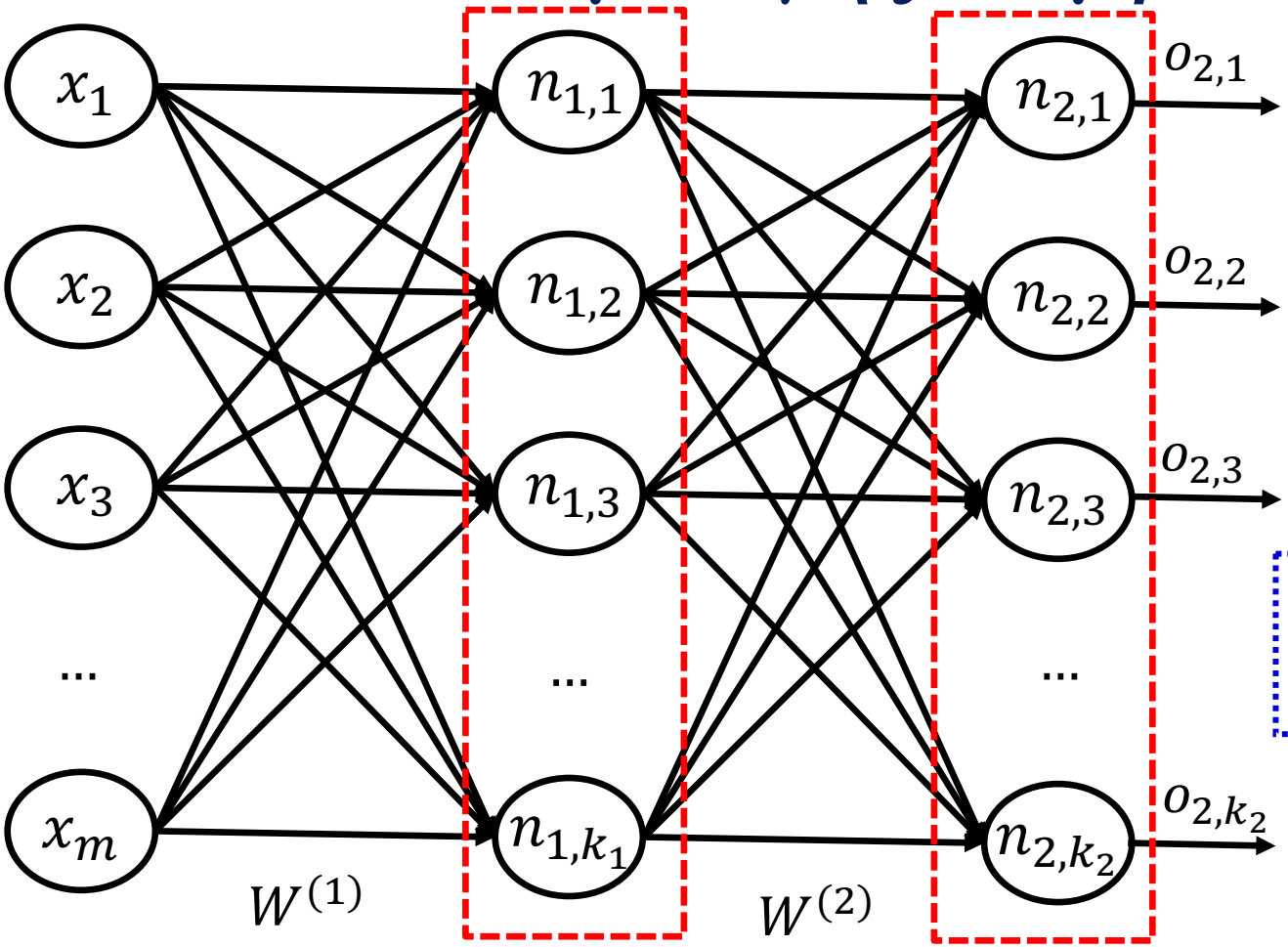
$$\vec{o} = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ \dots \\ o_k \end{bmatrix} = \begin{bmatrix} \Phi(s_1) \\ \Phi(s_2) \\ \Phi(s_3) \\ \dots \\ \Phi(s_k) \end{bmatrix} = \Phi(\vec{s}) = \Phi(W\vec{x})$$

Πιο συμπαγής συμβολισμός



$$\vec{o}^{(1)} = \begin{bmatrix} o_{1,1} \\ o_{1,2} \\ \dots \\ o_{1,k_1} \end{bmatrix} = \Phi\left(\vec{s}^{(1)}\right) = \Phi\left(W^{(1)}\vec{x}\right)$$

Πιο συμπαγής συμβολισμός



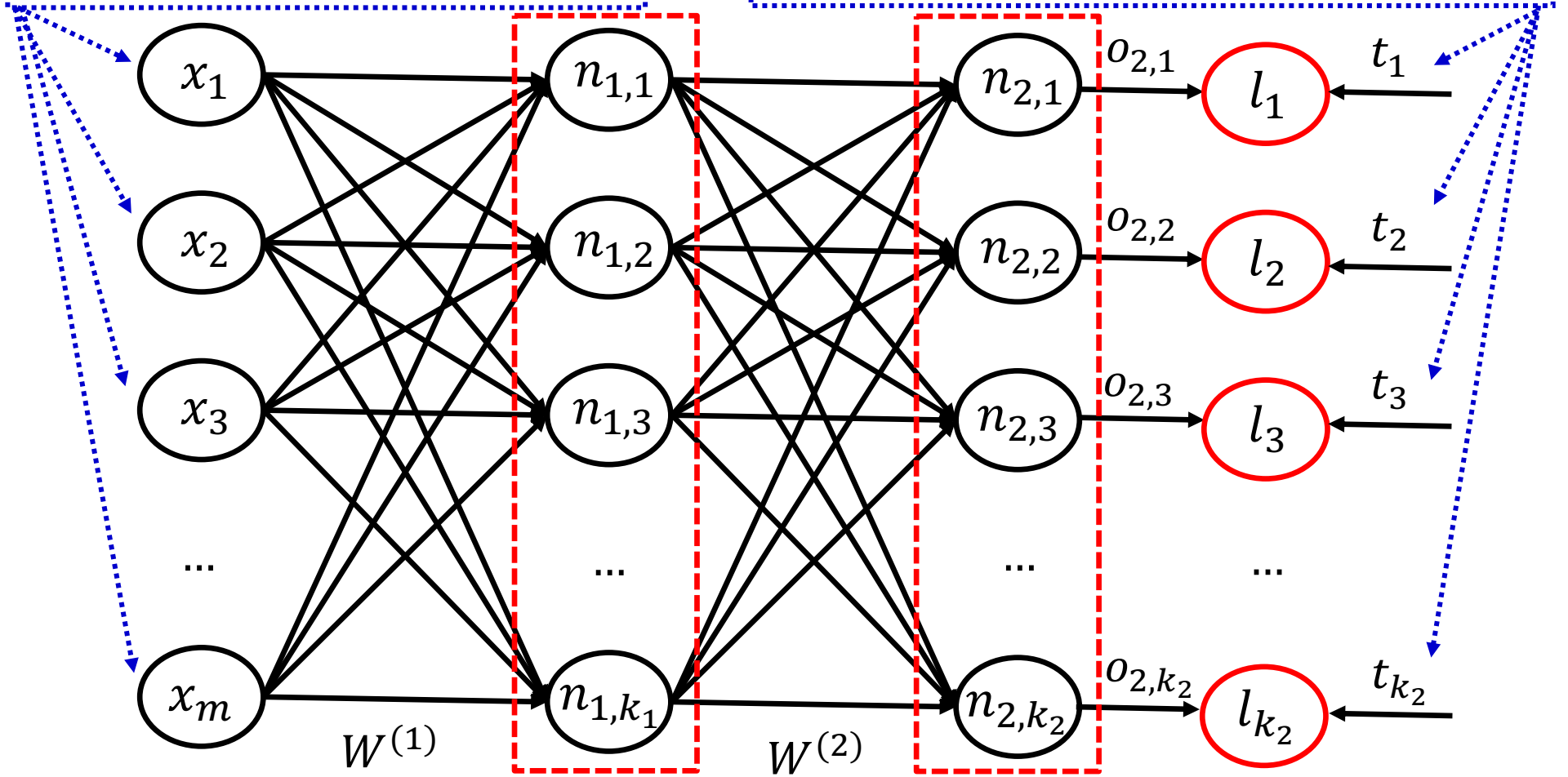
Μαθαίνουμε τους $\boldsymbol{W^{(1)}}, \boldsymbol{W^{(2)}}$ με **ανάστροφη μετάδοση**.

$$\vec{o}^{(1)} = \begin{bmatrix} o_{1,1} \\ o_{1,2} \\ \dots \\ o_{1,k_1} \end{bmatrix} = \Phi\left(W^{(1)}\vec{x}\right) \qquad \vec{o}^{(2)} = \begin{bmatrix} o_{2,1} \\ o_{2,2} \\ \dots \\ o_{2,k_2} \end{bmatrix} = \Phi\left(W^{(2)}\vec{o}^{(1)}\right)$$

Παράδειγμα παλινδρόμησης (regression)

Διάνυσμα εισόδου (π.χ. συχνότητες λέξεων ή τιμές pixels)

Σωστές έξοδοι (π.χ. τιμές διαστάσεων προσωπικότητας, συντεταγμένες ματιών)



$$\vec{o}^{(1)} = \tanh\left(W^{(1)}\vec{x}\right)$$
$$\vec{o}^{(2)} = W^{(2)}\vec{o}^{(1)}$$

Μέσο τετραγωνικό σφάλμα για το **τρέχον διάνυσμα εισόδου**

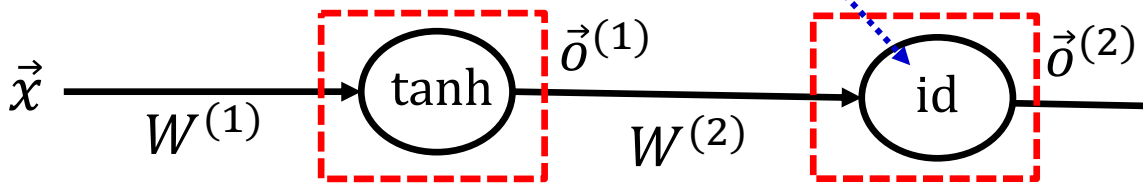
$$E = \frac{1}{k_2}\sum_{j=1}^{k_2} l_j^2 = \frac{1}{k_2}\sum_{j=1}^{k_2}\left(o_{2,j} - t_j\right)^2$$

Παράδειγμα παλινδρόμησης – πιο συμπαγής συμβολισμός

Διάνυσμα εισόδου (π.χ. συχνότητες λέξεων, τιμές pixels)

$\Phi(s) = s$ (identity)

Σωστές έξοδοι (π.χ. διαστάσεις προσωπικότητας, συντεταγμένες ματιών)



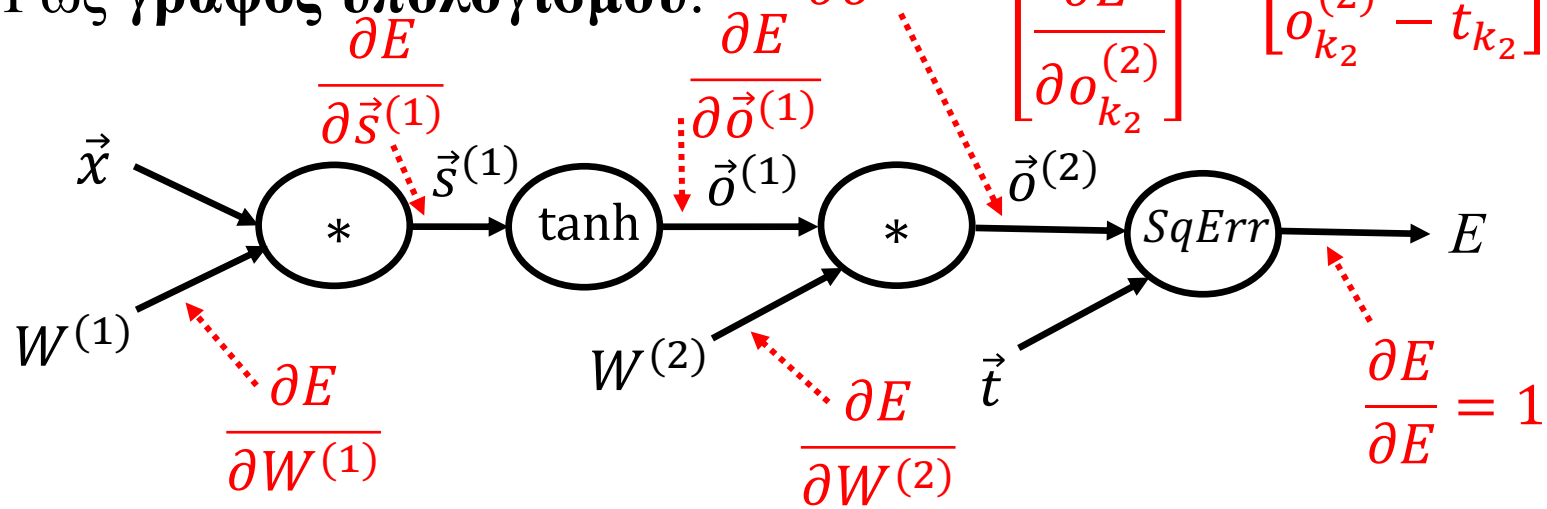
$$\vec{o}^{(1)} = \tanh\left(W^{(1)}\vec{x}\right) \quad \vec{o}^{(2)} = W^{(2)}\vec{o}^{(1)}$$

Μέσο τετραγωνικό σφάλμα για το **τρέχον διάνυσμα εισόδου**

Μαθαίνουμε τους πίνακες $\boldsymbol{W^{(1)}}, \boldsymbol{W^{(2)}}$ με **ανάστροφη μετάδοση**.

$$\frac{\partial E}{\partial \vec{o}^{(2)}} = \begin{bmatrix} \frac{\partial E}{\partial o_1^{(2)}} \\ \dots \\ \frac{\partial E}{\partial o_{k_2}^{(2)}} \end{bmatrix} = \begin{bmatrix} o_1^{(2)} - t_1 \\ \dots \\ o_{k_2}^{(2)} - t_{k_2} \end{bmatrix} = \vec{o}^{(2)} - \vec{t}$$

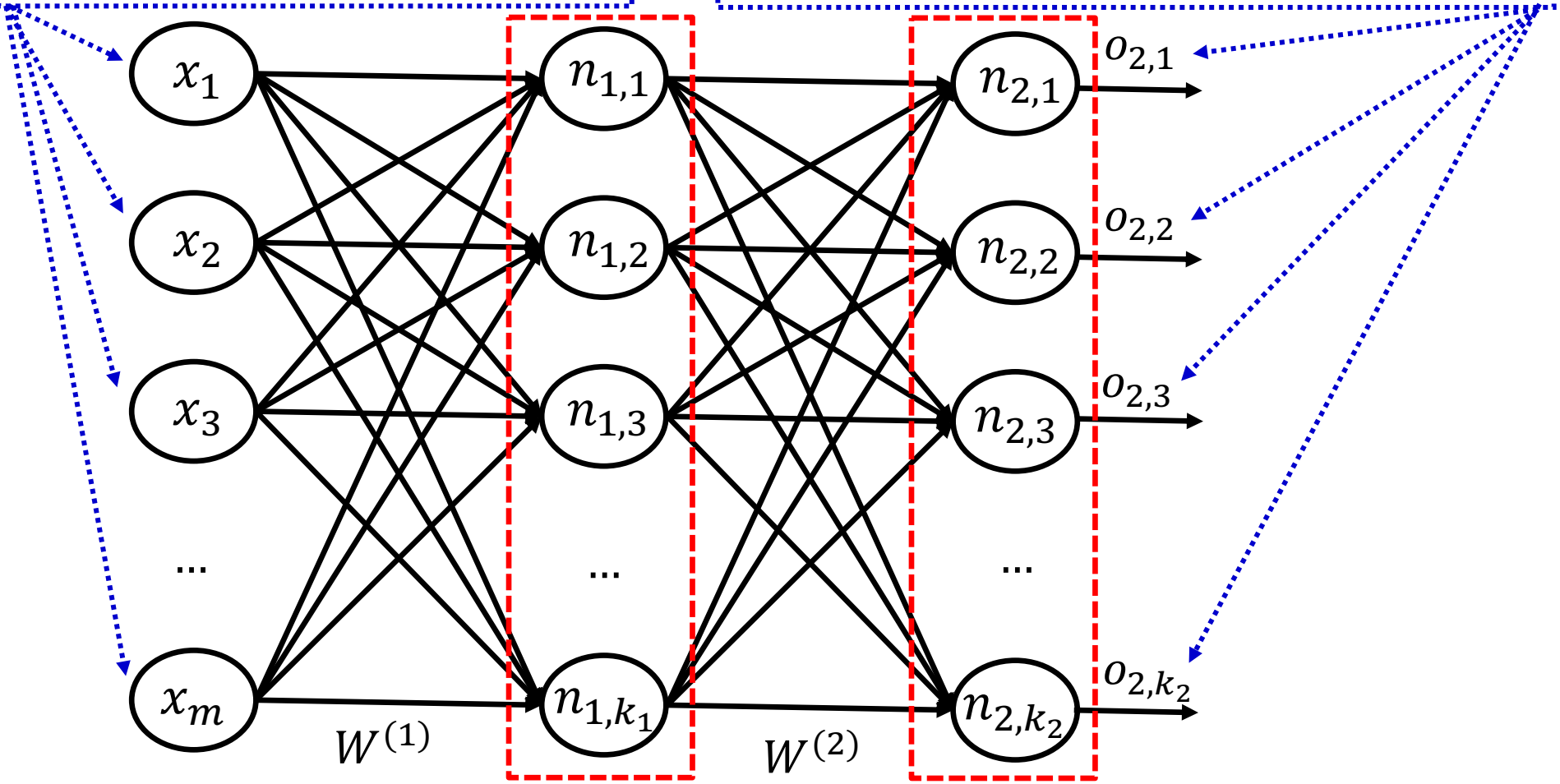
Ή ως **γράφος υπολογισμού**:



Παράδειγμα κατηγοριοποίησης

Διάνυσμα εισόδου (π.χ. συχνότητες λέξεων, τιμές pixels)

Πόσο πιθανό θεωρεί το σύστημα να ανήκει η είσοδος σε κάθε μία από τις k_2 **κατηγορίες**.



$$\vec{o}^{(1)} = \tanh\left(W^{(1)}\vec{x}\right)$$

$$\vec{o}^{(2)} = \mathrm{softmax}\left(W^{(2)}\vec{o}^{(1)}\right)$$

Θεωρούμε εδώ ότι κάθε αντικείμενο προς κατάταξη (π.χ. κείμενο, εικόνα) ανήκει σε **ακριβώς μία κατηγορία**.

Softmax

$$W^{(2)}\vec{o}^{(1)} = \vec{s}^{(2)} = \begin{bmatrix} s_{2,1} \\ s_{2,2} \\ \dots \\ s_{2,k_2} \end{bmatrix}$$

Έξοδοι τελευταίου επιπέδου χωρίς συνάρτηση ενεργοποίησης. **Βαθμοί βεβαιότητας** (πραγματικοί αριθμοί) **για κάθε κατηγορία**. Θέλουμε να τους **μετατρέψουμε σε πιθανότητες με άθροισμα 1**.

$$\text{softmax}\left(W^{(2)}\vec{o}^{(1)}\right) = \text{softmax}\left(\vec{s}^{(2)}\right) = \text{softmax}\left(\begin{bmatrix} s_{2,1} \\ s_{2,2} \\ \dots \\ s_{2,k_2} \end{bmatrix}\right)$$

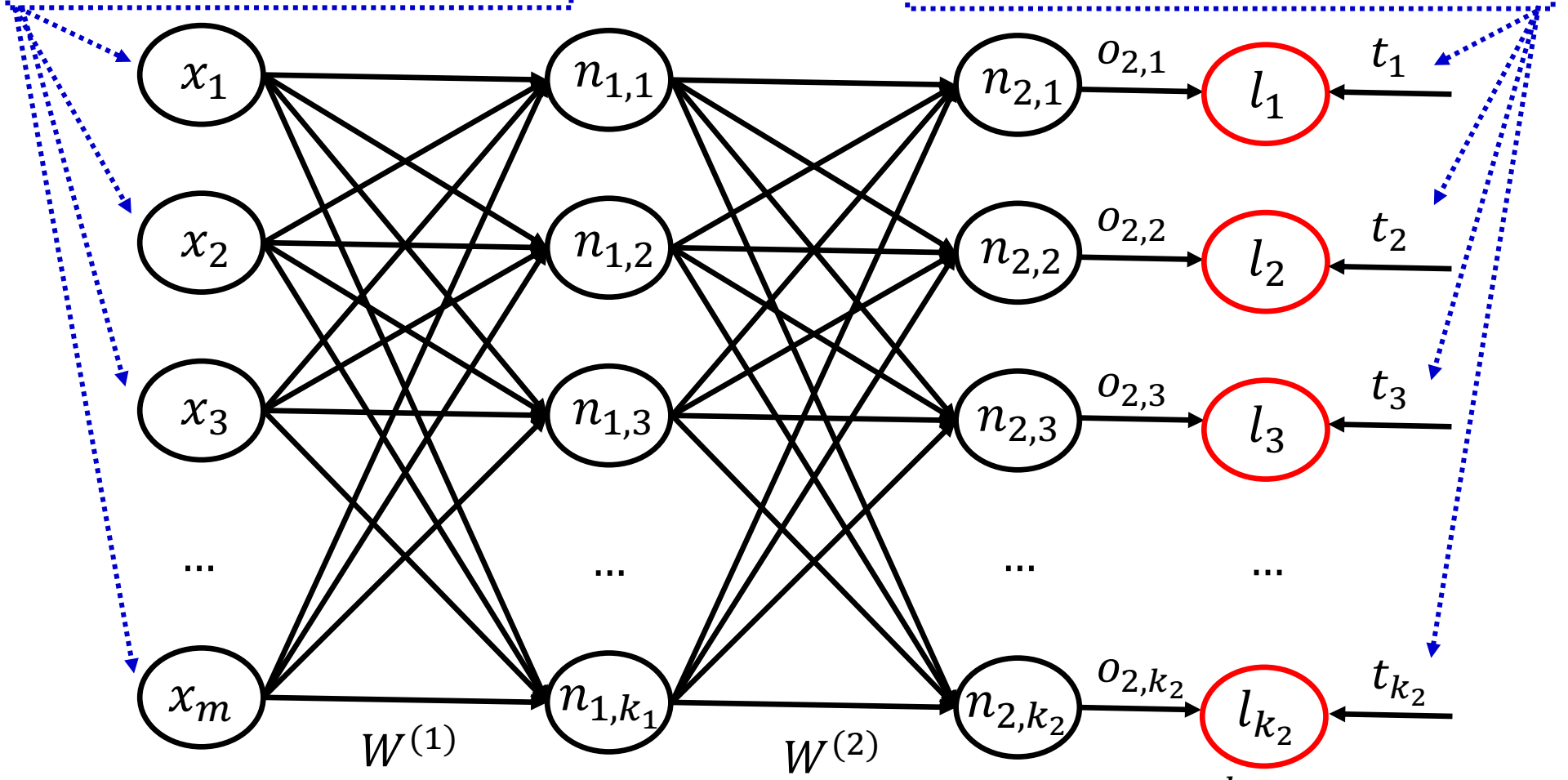
$$= \begin{bmatrix} \dfrac{\exp\left(s_{2,1}\right)}{\sum_{j=1}^{k_2}\exp\left(s_{2,j}\right)} \\ \dfrac{\exp\left(s_{2,2}\right)}{\sum_{j=1}^{k_2}\exp\left(s_{2,j}\right)} \\ \dots \\ \dfrac{\exp\left(s_{2,k_2}\right)}{\sum_{j=1}^{k_2}\exp\left(s_{2,j}\right)} \end{bmatrix}$$

Η softmax **μετακινεί επίσης τις μεγαλύτερες από τις εισόδους της προς το 1**, ενώ τις **υπόλοιπες προς το 0**. Διαισθητικά **soft argmax**!

Single-label multi-class classification example

Input instance (e.g., word frequencies or pixel values)

Correct output ($t_j = 1$ means the **single** correct class is the j-th one)



$$\vec{o}^{(1)} = \tanh\left(W^{(1)}\vec{x}\right)$$

$$\vec{o}^{(2)} = \mathrm{softmax}\left(W^{(2)}\vec{o}^{(1)}\right)$$

Cross entropy loss at the **current** training instance. See slides below.

$$l = -\sum_{j=1}^{k_2} t_j \log\left(o_{2,j}\right)$$

Διασταυρωμένη εντροπία

$$\vec{o}^{(2)} = \begin{bmatrix} P_m(C = c_1) \\ P_m(C = c_2) \\ P_m(C = c_3) \\ \dots \\ P_m(C = c_k) \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.12 \\ 0.08 \\ \dots \\ 0.14 \end{bmatrix}$$

Οι **εκτιμήσεις πιθανοτήτων** του **ταξινομητή** για τις κατηγορίες.

$$\vec{t} = \begin{bmatrix} P(C = c_1) \\ P(C = c_2) \\ P(C = c_3) \\ \dots \\ P(C = c_k) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

Οι **σωστές «πιθανότητες»** για τις κατηγορίες. Διάνυσμα **1-hot**.

Η **λογαριθμική πιθανοφάνεια** της **σωστής κατηγορίας** σύμφωνα με τον **ταξινομητή** (αλλά με αρνητικό πρόσημο).

$$H_{P_m}(C) = -\sum_{i=1}^{k} P(C = c_i) \log_2 P_m(C = c_i) = -\log_2 P_m(C = c_2)$$

Ελαχιστοποιούμε τη διασταυρωμένη εντροπία ή ισοδύναμα **μεγιστοποιούμε τη λογαριθμική πιθανοφάνεια** (την πιθανότητα που δίνει το μοντέλο στη σωστή απάντηση).

Διασταυρωμένη εντροπία (cross-entropy)

- Η **εντροπία** μιας τυχαίας μεταβλητής C δείχνει **πόσο αβέβαιοι** είμαστε για την τιμή της.

$$H(C) = -\sum_{c_i} P(C = c_i) \cdot \log_2 P(C = c_i)$$

  - **Πόσα bits** (αναμενόμενη τιμή) χρειάζεται να **μεταδώσουμε** με **ιδανική κωδικοποίηση** για να μεταδώσουμε την τιμή της.
  - Χρησιμοποιούμε $-\log_2 P(c_i)$ **bits** για κάθε δυνατή **τιμή** c_i.

- Αν χρησιμοποιούμε **κωδικοποίηση** βασισμένη σε **ανακριβείς εκτιμήσεις πιθανοτήτων** $P_m(c_i)$:

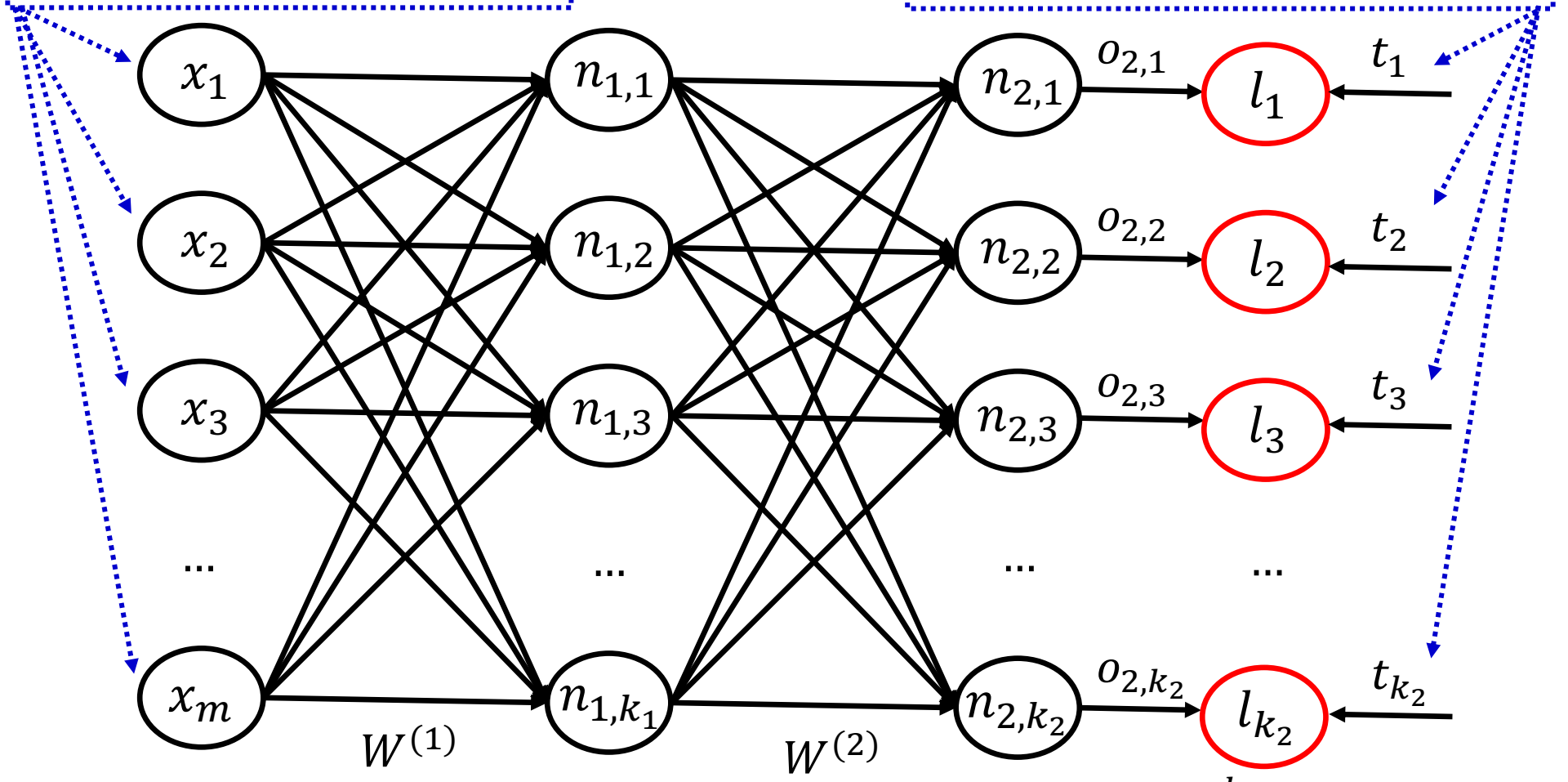
$$H_{P_m}(C) = -\sum_{c_i} P(C = c_i) \cdot \log_2 P_m(C = c_i) \geq H(C)$$

  - Η **διασταυρωμένη εντροπία** μάς λέει **πόσα bits μεταδίδουμε**.
  - Όσο **πιο λανθασμένες** είναι οι **εκτιμήσεις** $P_m(c_i)$, τόσο **πιο πολλά bits μεταδίδουμε**.

Single-label multi-class classification example

Input instance (e.g., word frequencies or pixel values)

Correct output ($t_j = 1$ means the **single** correct class is the j-th one)



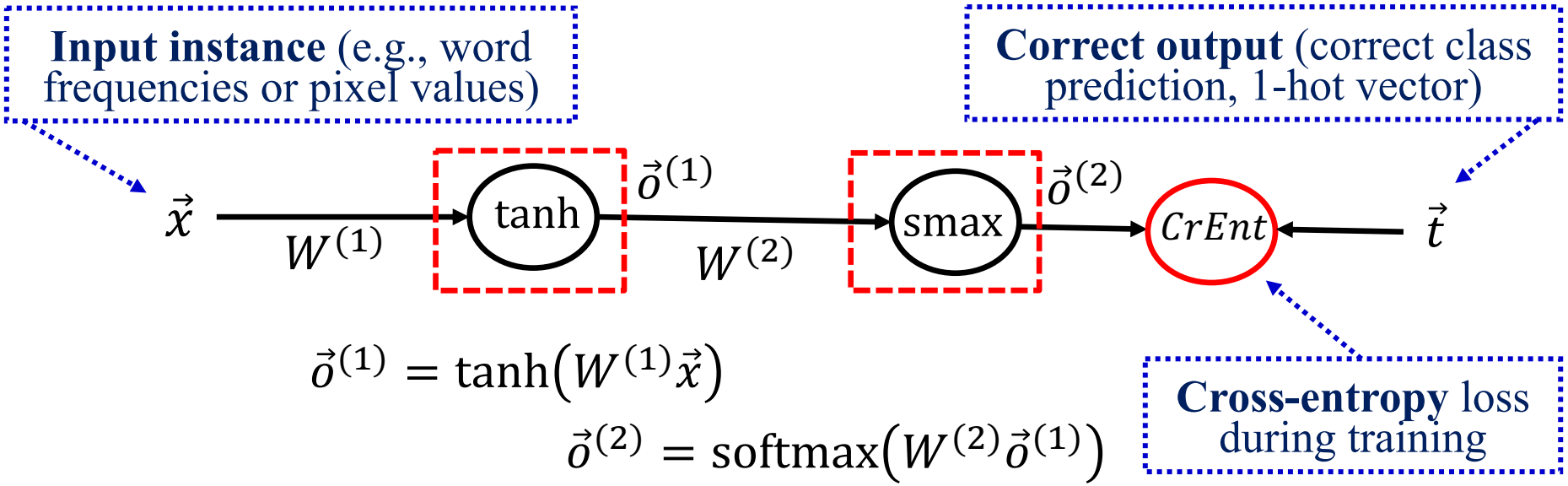
$$\vec{o}^{(1)} = \tanh\left(W^{(1)}\vec{x}\right)$$

$$\vec{o}^{(2)} = \mathrm{softmax}\left(W^{(2)}\vec{o}^{(1)}\right)$$

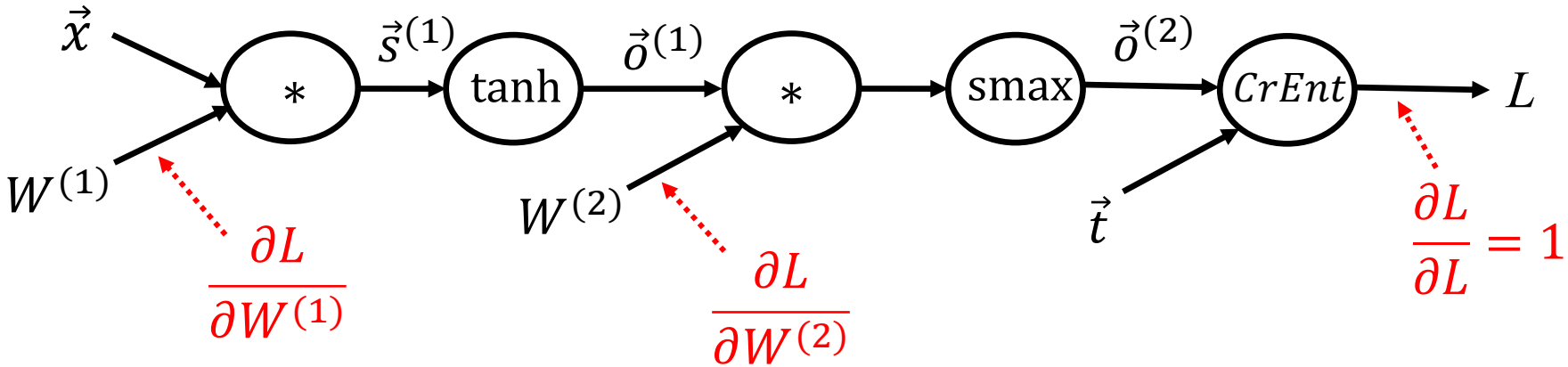
Cross entropy loss at the **current** training instance.

$$l = -\sum_{j=1}^{k_2} t_j \log\left(o_{2,j}\right)$$

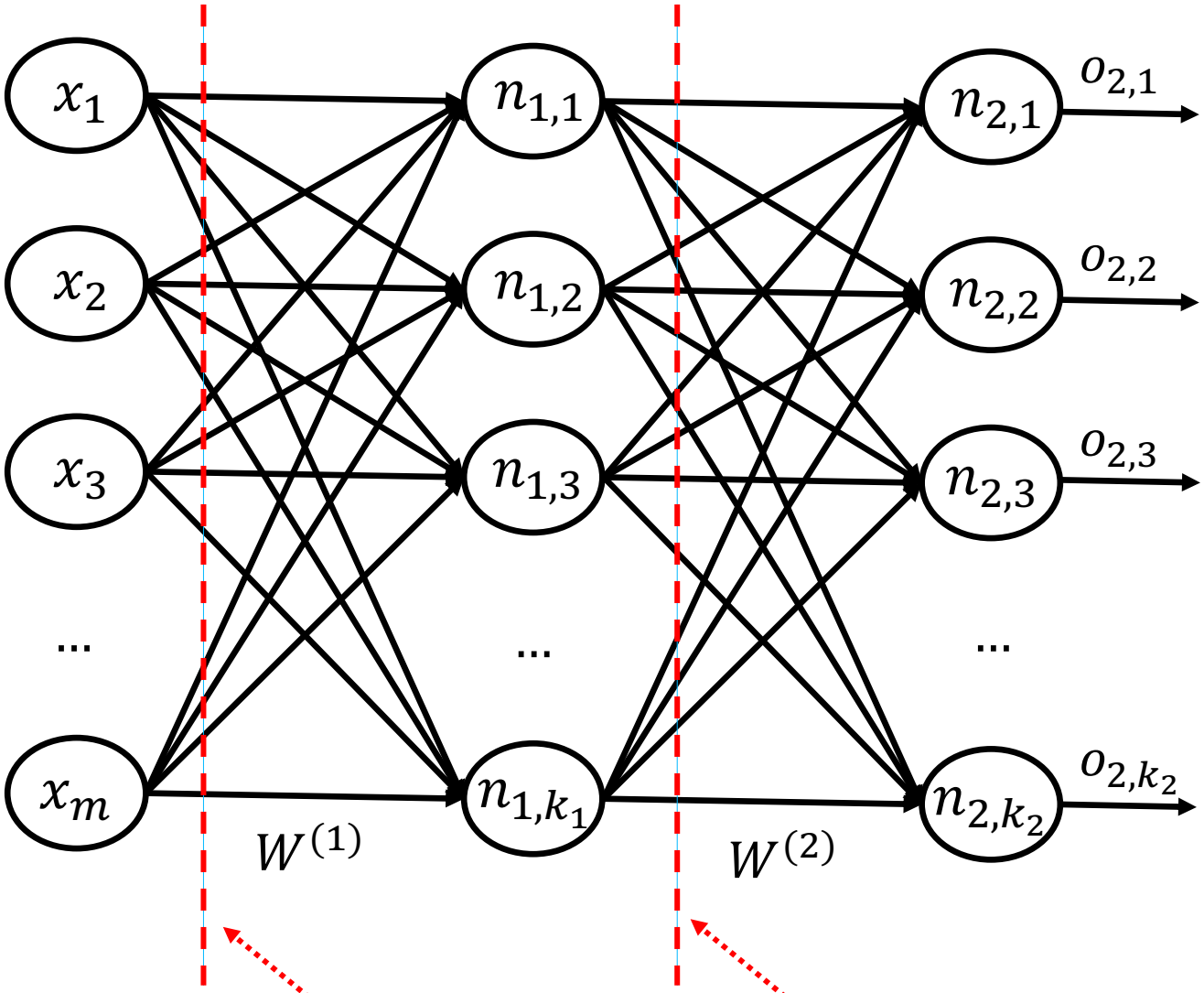
Classification example – more compact



Or as a **computation graph**:



Dropout



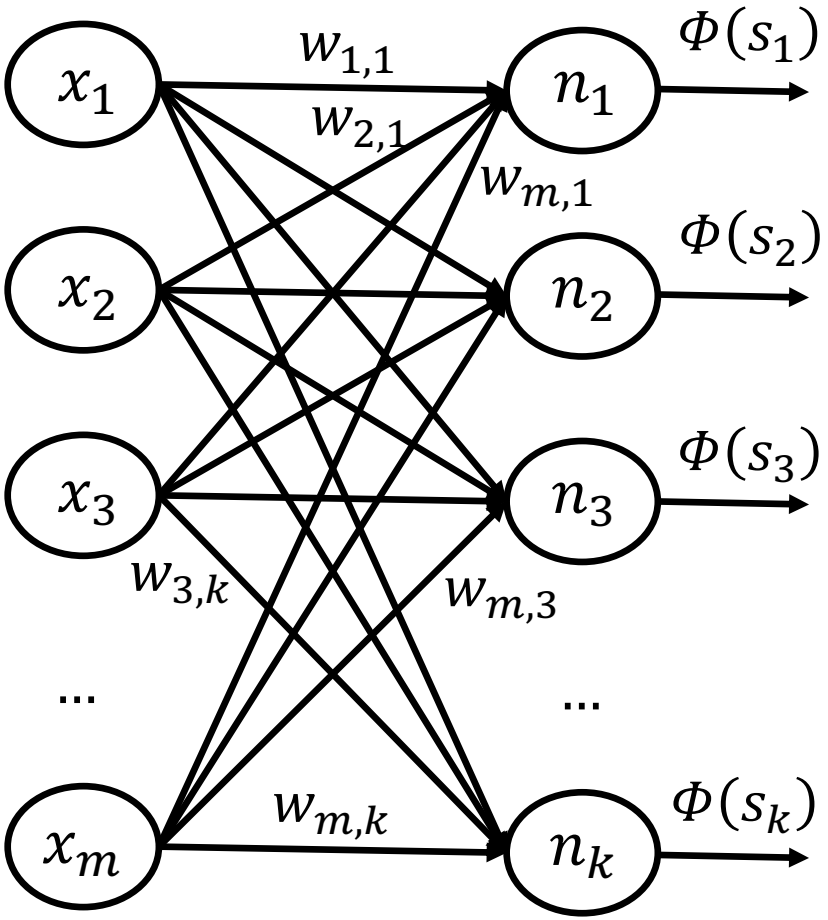
Dropout at the input layer. E.g. $p_{drop} = 0.2$.

Dropout at the output of a hidden layer. E.g., $p_{drop} = 0.5$.

Dropout

- **For each training instance** (or mini-batch), we **drop** (remove) **each neuron** of the layer where dropout is applied with **probability** $\boldsymbol{p_{drop} = 1 - p_{keep}}$.
  - Helps the neural net **avoid relying too much on particular neurons** (or inputs). A form of **regularization**. Works well!
  - **Gradients** also **do not flow** through dropped neurons.
  - Alternative explanation: we train an **ensemble** of networks, containing **all the pruned networks** that dropout creates.
- **During testing**, we **multiply the output** of each neuron (of the layer where dropout was applied) by $\boldsymbol{p_{keep}}$, so that the neuron's **expected output value** will be **as in training**.
  - **Or we divide** the output by $\boldsymbol{p_{keep}}$ **during training** instead.
  - We **don't drop neurons during testing** (only during training).

Batch normalization



At each layer, instead of:

$$s_j = \sum_{i=1}^{m} w_{i,j}\, x_i$$

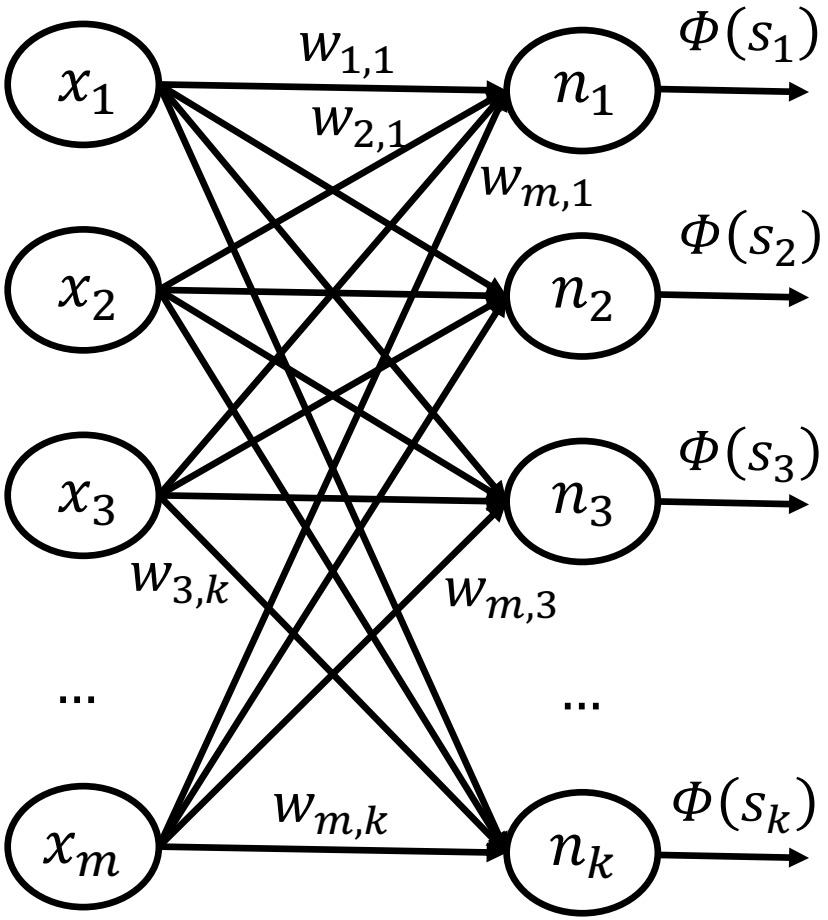
we use:

$$\bar{s}_j = \frac{g_j}{\sigma_j}\left(s_j - \mu_j\right) + b_j$$

- μ_j, σ_j are the **mean** and **std. dev. of s_j** in the **mini-batch**,
- g_j, b_j are **learned** parameters (constant after training).
- Φ is now applied to $\bar{s}_j$.

See https://arxiv.org/pdf/1607.06450.pdf for **batch vs. layer normalization**.

Layer normalization



At each layer, instead of:

$$s_j = \sum_{i=1}^{m} w_{i,j}\, x_i$$

we use:

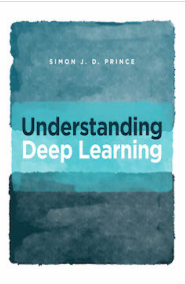
$$\bar{s}_j = \frac{g_j}{\sigma}\left(s_j - \mu\right) + b_j$$

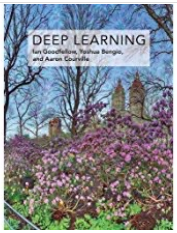
- μ, σ are the **mean** and **std. dev. of $s_1, \ldots, s_k$** in the **layer**,
- g_j, b_j are **learned** parameters (constant after training).

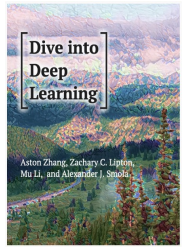
See https://arxiv.org/pdf/1607.06450.pdf for **batch vs. layer normalization**. The latter is better for RNNs (next part), where layers are time-steps.

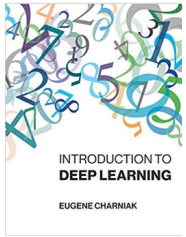
Προτεινόμενη βιβλιογραφία

- 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.

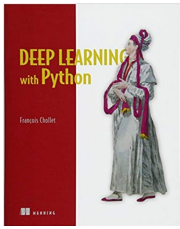








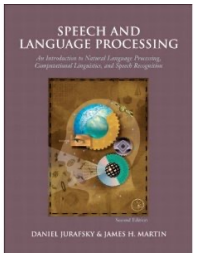
Προτεινόμενη βιβλιογραφία – συνέχεια



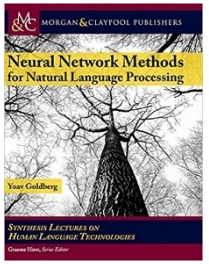
- Ως πιο πρακτική εισαγωγή, προτείνεται ιδιαίτερα το βιβλίο του F. Chollet «Deep Learning in Python», Manning Publications.
  - Καλύπτει σε μεγάλο βαθμό την ύλη αυτών των διαφανειών (βλ. κεφ. 1–4). Αποτελεί και εισαγωγή στο Keras (πλέον μέρος της βιβλιοθήκης Tensorflow), που θα χρησιμοποιηθεί στα φροντιστήρια 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/

Προτεινόμενη βιβλιογραφία – συνέχεια

- Όσοι ενδιαφέρονται ιδιαίτερα για την επεξεργασία φυσικής γλώσσας (και φωνής) αξίζει να μελετήσουν σταδιακά το εξαιρετικό βιβλίο «Speech and Language Processing» των D. Jurafsky and J.H. Martin, 2η έκδοση, Prentice Hall, 2008.
  - Υπάρχει και στη βιβλιοθήκη του ΟΠΑ.
  - Διατίθεται ελεύθερα η υπό προετοιμασία 3η έκδοση. Βλ. http://web.stanford.edu/~jurafsky/slp3/.

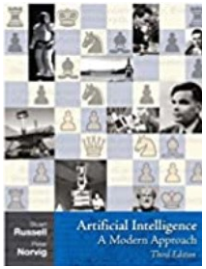


- Μια πολύ καλή εισαγωγή στη χρήση βαθιάς μάθησης για επεξεργασία φυσικής γλώσσας είναι το βιβλίο του Y. Goldberg «Neural Network Models for Natural Language Processing», Morgan & Claypool Publishers, 2017.
  - Υπάρχει και στη βιβλιοθήκη του ΟΠΑ.



Βιβλιογραφία – συνέχεια

- Αν έχετε από το μάθημα της ΤΝ το βιβλίο των Russel & Norvig «Τεχνητή Νοημοσύνη – Μια σύγχρονη προσέγγιση», 4η έκδοση, Κλειδάριθμος, 2021, μπορείτε να συμβουλευτείτε τα κεφάλαια 21 και 24.
  - Κυρίως τις ενότητες 21.1, 21.2, 21.4, 21.5, 24.1.
  - Άλλες ενότητες αυτών των κεφαλαίων θα καλυφθούν σε επόμενες διαλέξεις.



Other recommended resources

- Useful maths background: T. Parr και J. Howard, *The Matrix Calculus You Need for Deep Learning*.
  - https://explained.ai/matrix-calculus
- PyTorch tutorials: https://pytorch.org/tutorials/
- C. Manning's (Stanford) *NLP with Deep Learning* course.
  - Course material: http://web.stanford.edu/class/cs224n/
  - Videos: available on YouTube.
- J. Johnson's (U Michigan) *Deep Learning for Computer Vision* course.
  - See: https://www.youtube.com/playlist?list=PL5-TkQAfAZFbzxjBHtzdVCWE0Zbhomg7r