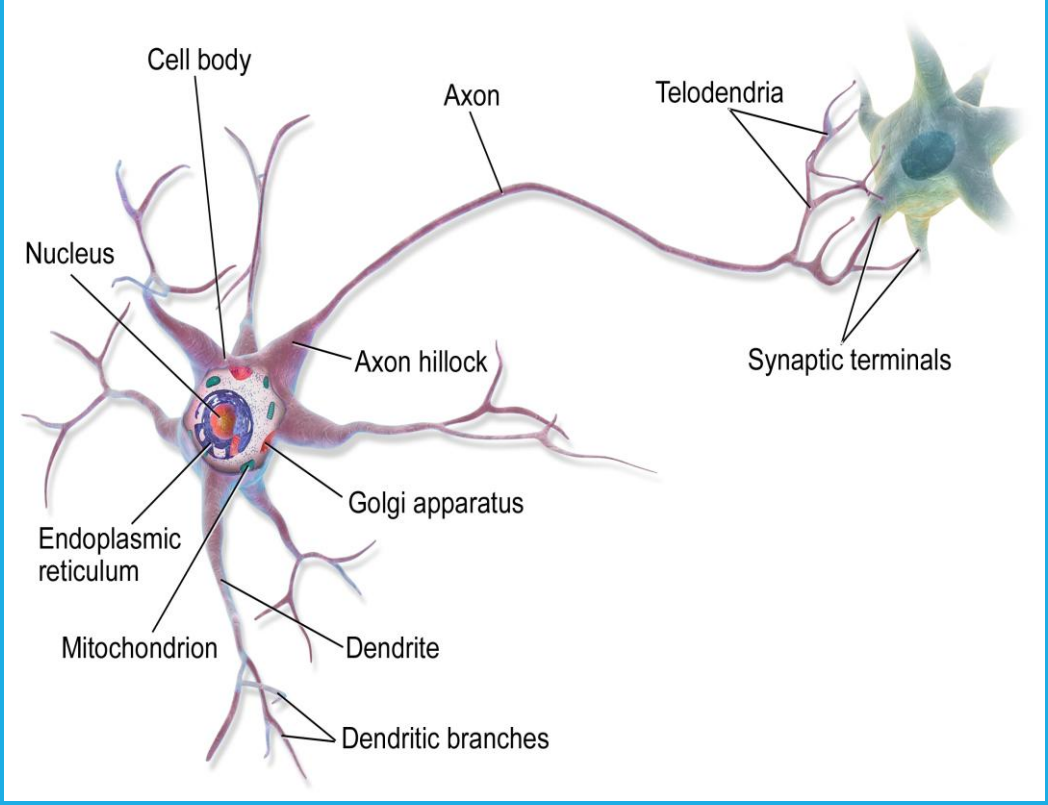
ΓΛΩΣΣΙΚΗ ΤΕΧΝΟΛΟΓΙΑ

Κάτια Κερμανίδου kerman@ionio.gr

ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

Αποσπάσματα κάποιων διαφανειών είναι δανεισμένα από τα slides του Speech and Language Processing

Ο ανθρώπινος εγκέφαλος – Ο τεχνητός νευρώνας



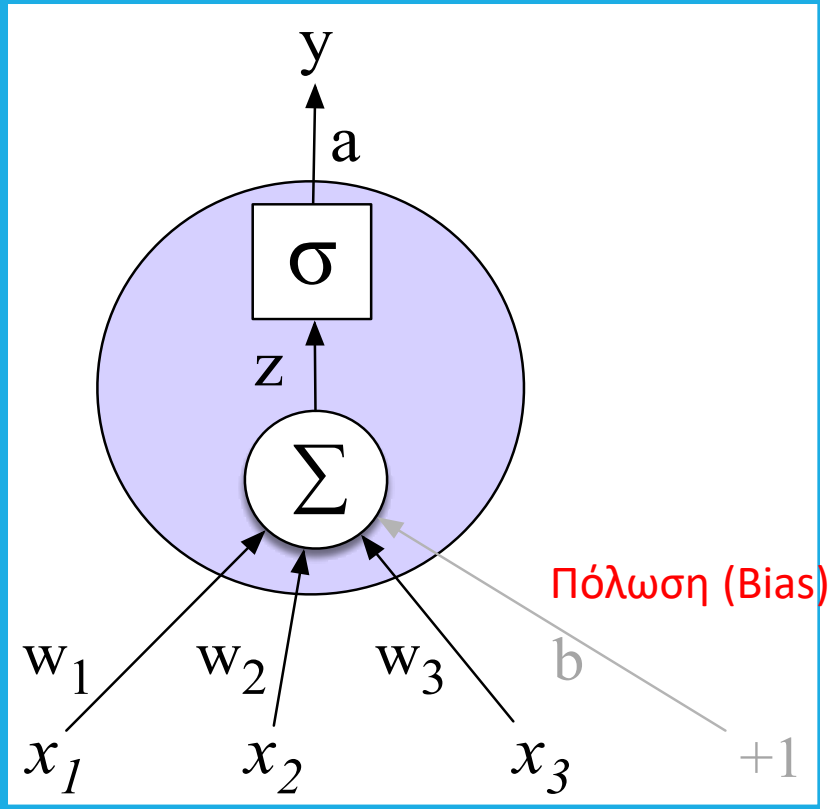
έξοδος

Μη γραμμικός μετασχηματισμός

Άθροισμα

Κάθε είσοδος πολλαπλασιάζεται με ένα βάρος

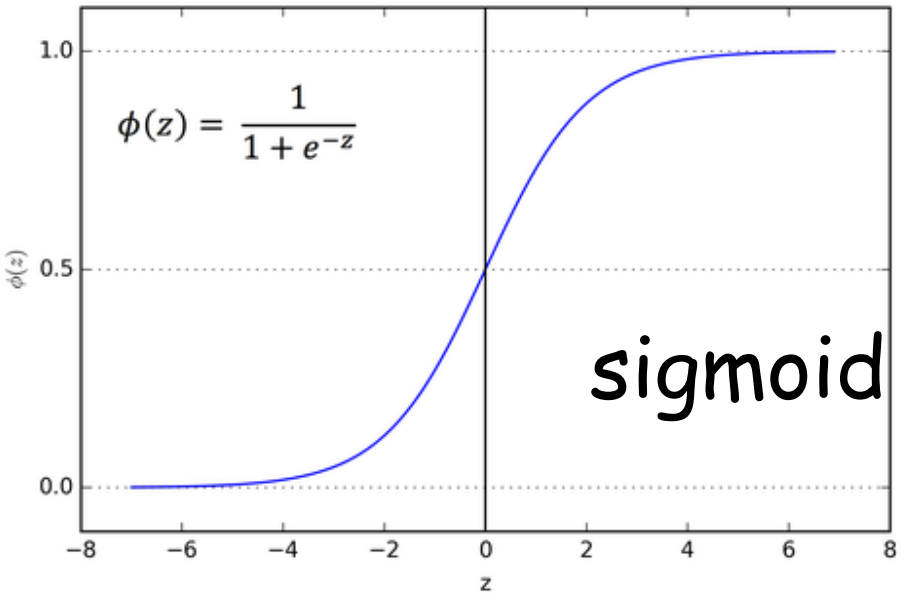
είσοδος

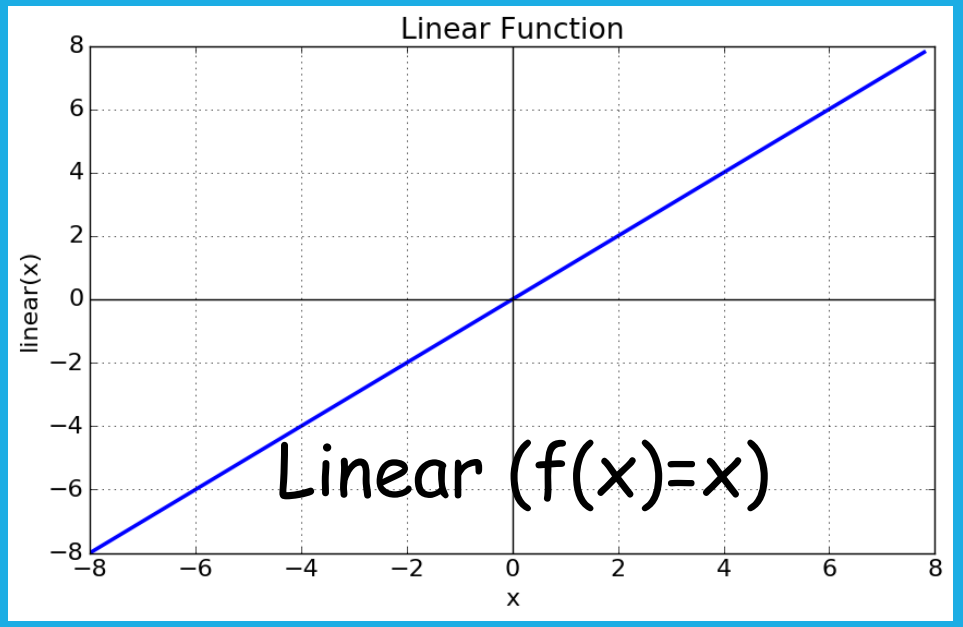


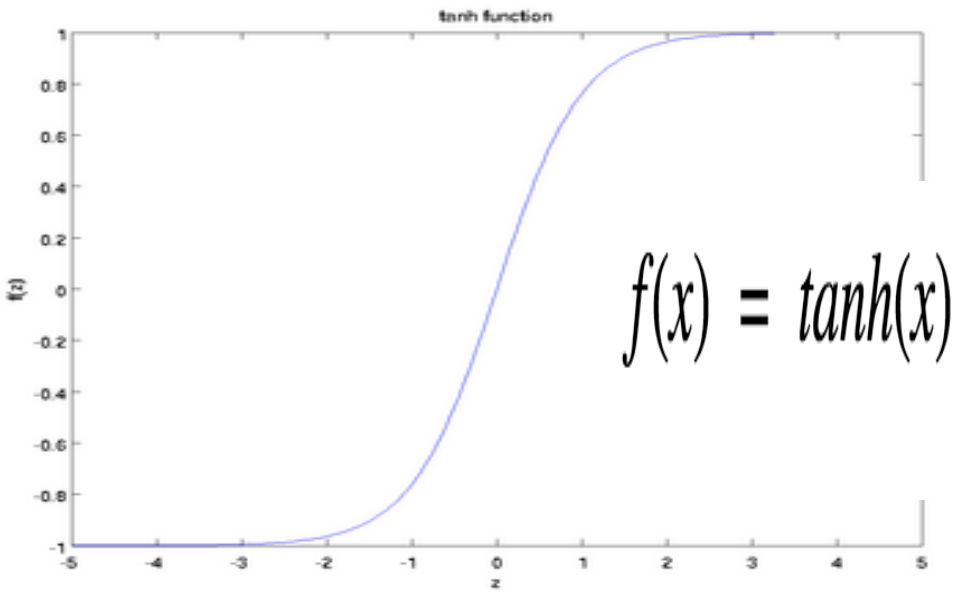
By BruceBlaus - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=28761830

$$y = \sigma\left(\Sigma(w_i x_i + b)\right)$$

Τι μπορεί να είναι η συνάρτηση ενεργοποίησης (activation function);







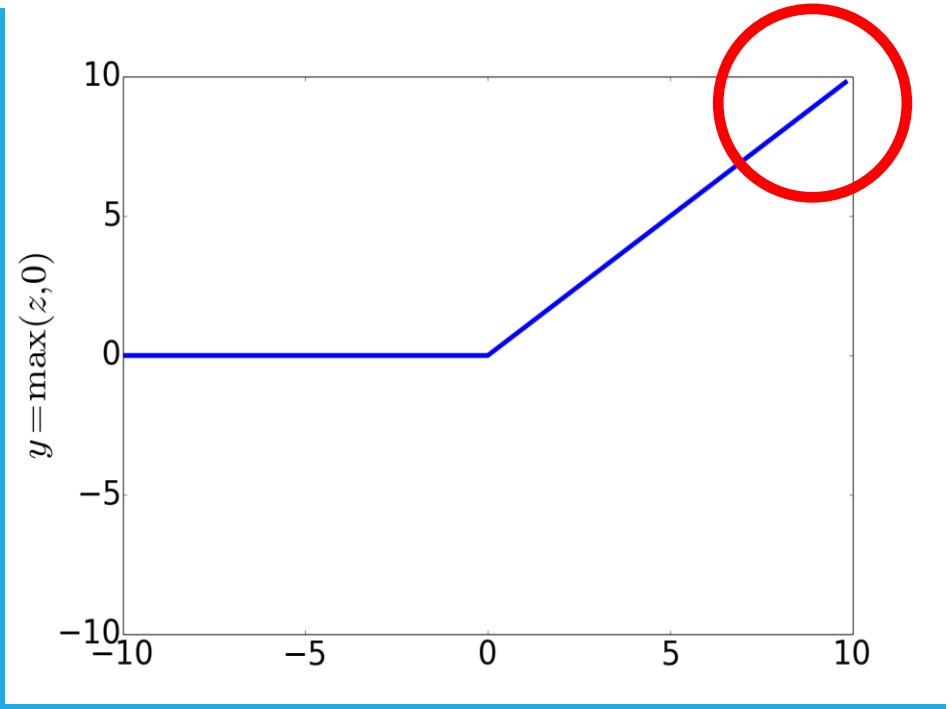
$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

Η πιο συχνά χρησιμοποιούμενη συνάρτηση ενεργοποίησης: Rectified Linear Unit (ReLU)

y = max(wx+b, 0)

Δηλαδή

αν το άθροισμα των επιβαρυμένων εισόδων στον νευρώνα είναι αρνητικό, η έξοδος του νευρώνα θα είναι 0, αλλιώς θα ισούται με το άθροισμα των επιβαρυμένων εισόδων



Η ReLU είναι καλύτερη από τη σιγμοειδή και την tanh, γιατί

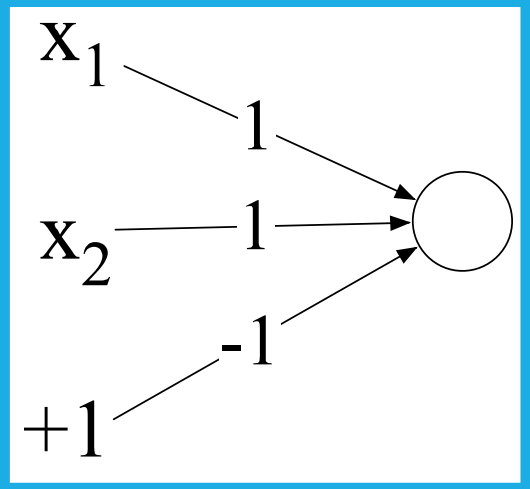
- Για μεγάλες τιμές του wx+b οι συναρτήσεις αυτές παρουσιάζουν κορεσμό με πάρα πολλές τιμές τους να είναι κοντά στο 1), με αποτέλεσμα να παρουσιάζουν μηδενικές παραγώγους, δηλ. σχεδόν μηδενικό σφάλμα, τόσο μικρό που δεν μπορεί να χρησιμοποιηθεί για εκπαίδευση (**vanishing gradient**)
- Η ReLU για μεγάλα wx+b έχει παραγώγους κοντά στο 1.

Το Perceptron

Ένα πολύ απλός νευρώνας,

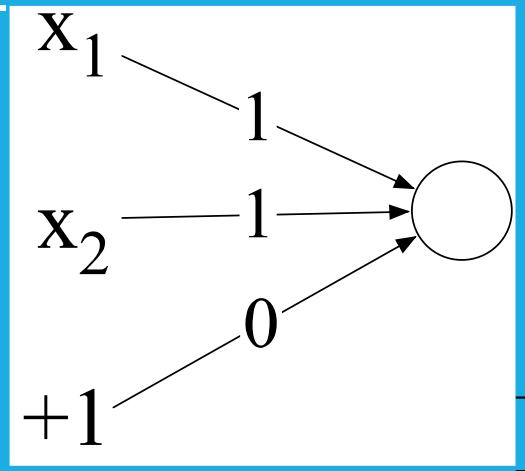
- **με την βηματική (step function) σαν συνάρτηση ενεργοποίησης**
- **με δυαδική έξοδο**

$f(x) = 1$, αν $b+\Sigma(xw) > 0$
$f(x)=0$, αλλιώς



AND

AND		
x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1



OR

OR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Αριθμητικό Παράδειγμα

Ας υποθέσουμε ότι ένας νευρώνας έχει:

$w = [0.2, 0.3, 0.9]$

$b = 0.5$

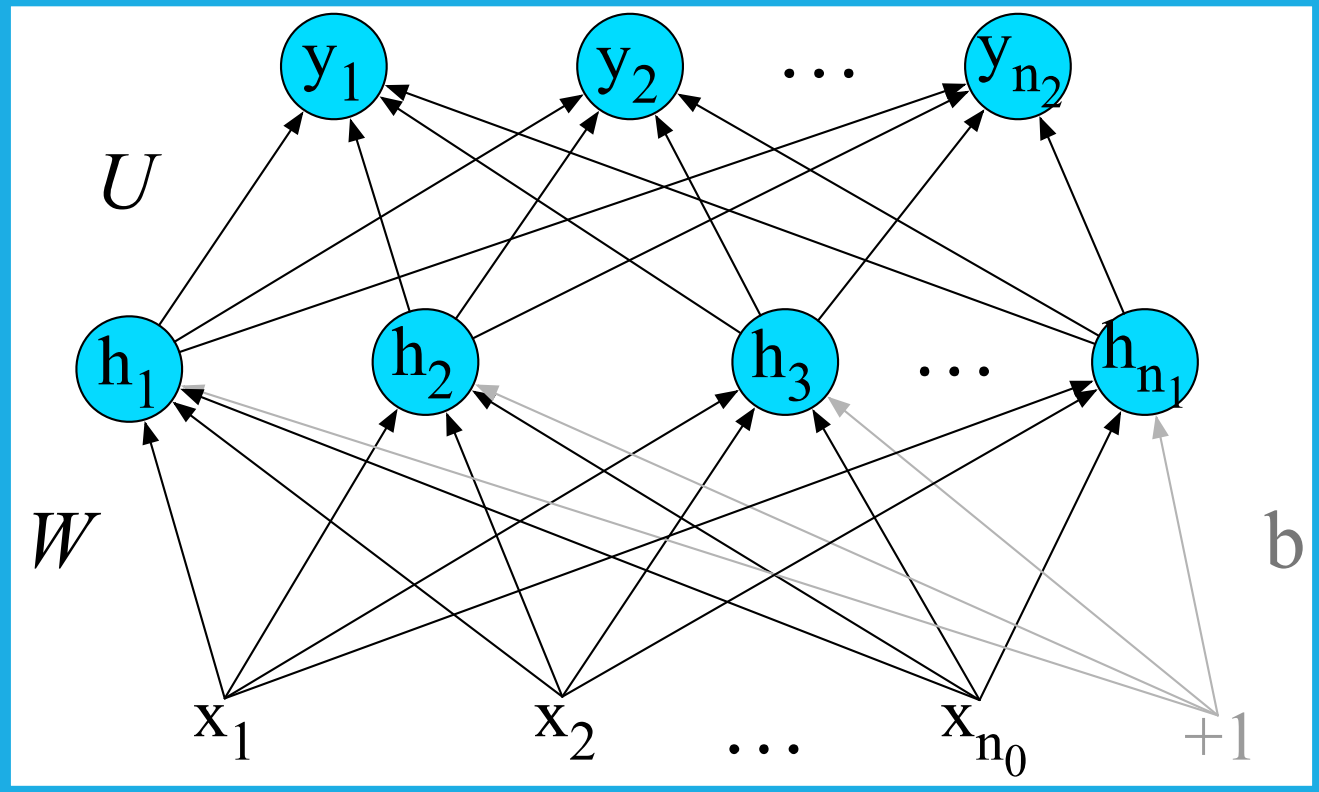
Τι θα συμβεί στην παρακάτω είσοδο x? Υποθέτουμε σιγμοειδή συνάρτηση ενεργοποίησης.

$x = [0.5, 0.6, 0.1]$

$$y = s(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}} = \frac{1}{1 + e^{-(.5 \times .2 + .6 \times .3 + .1 \times .9 + .5)}} = \frac{1}{1 + e^{-0.87}}$$

$$= .70$$

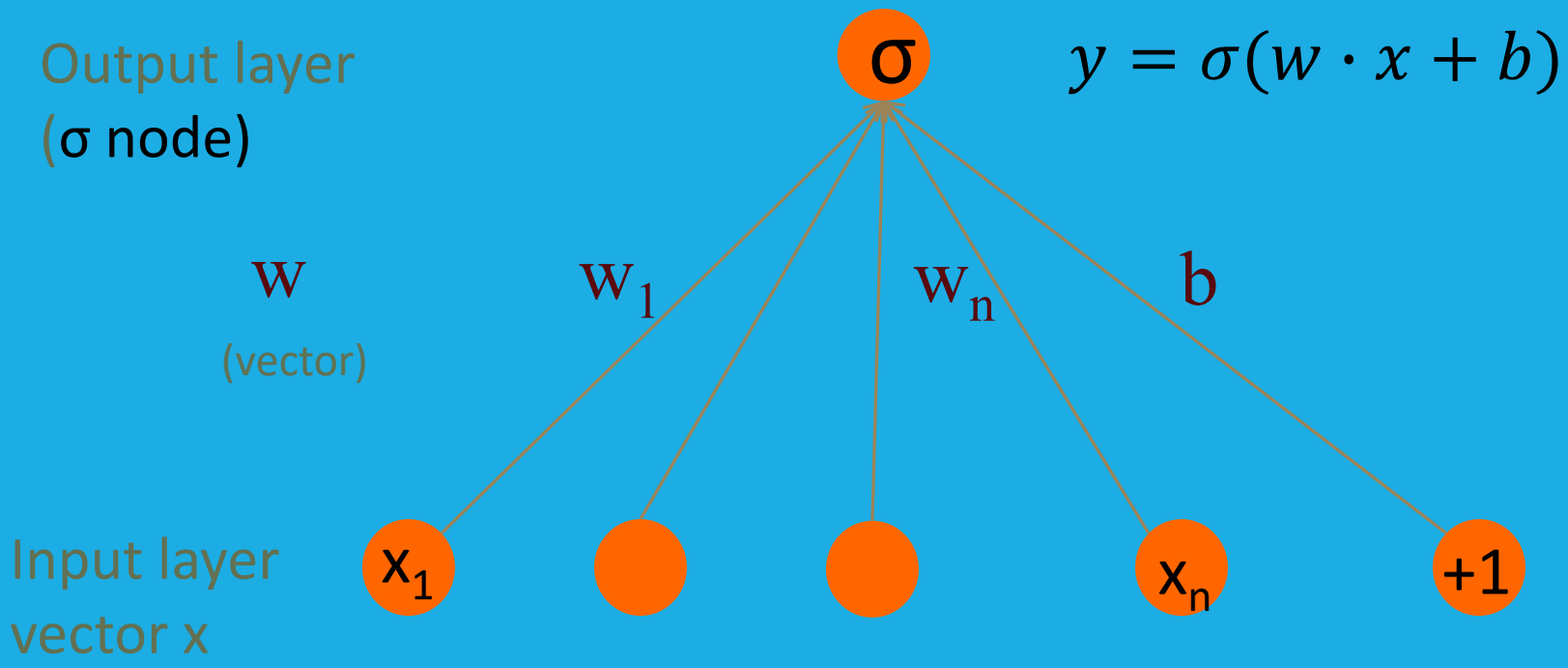
Feed-forward Neural Networks = Multi-layer Perceptrons



Feed-forward Neural Networks

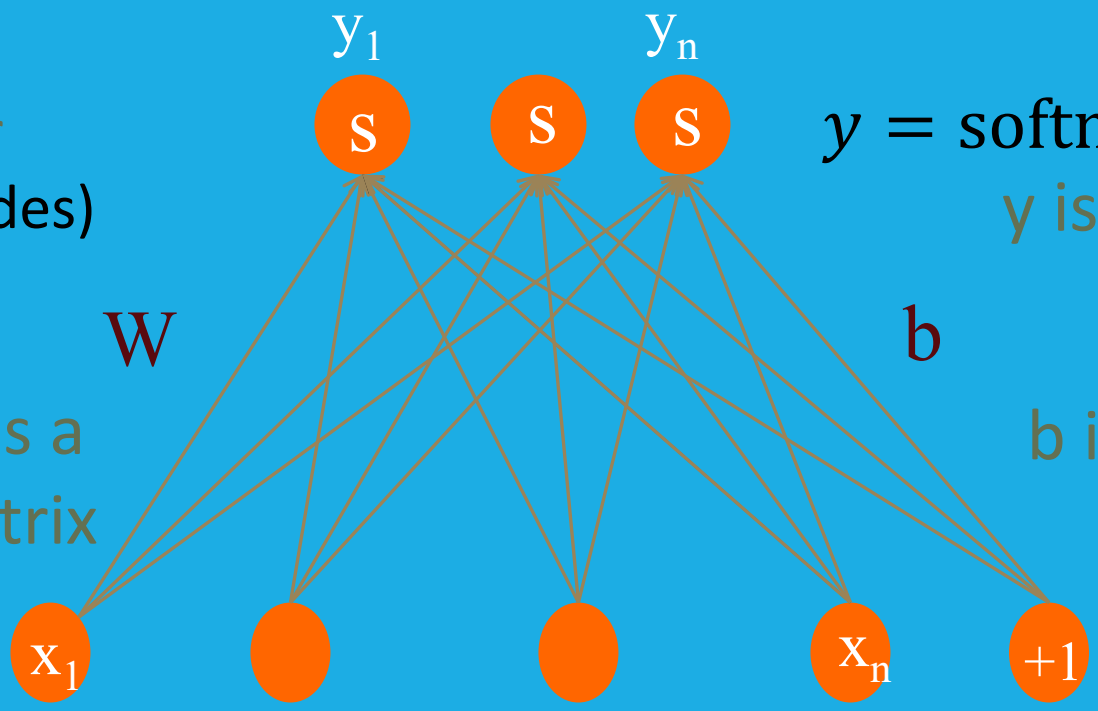
- Σε ένα FFNN, σε κάθε κρυφό κόμβο
  - Υπολογίζεται το άθροισμα των σταθμισμένων με βάρη εισόδων του κόμβου
    - Τα βάρη αυτά είναι οι παράμετροι που πρέπει το δίκτυο να μάθει κατά την εκπαίδευσή του
  - Εφαρμόζεται στο άθροισμα αυτό η συνάρτηση ενεργοποίησης (activation function)
  - Η συνάρτηση αυτή μπορεί να είναι γραμμική ή μη γραμμική

Παραδείγματα FFNN: Δίκτυο ενός επιπέδου (Logistic Regression) – δυαδική έξοδος



Παραδείγματα FFNN: Δίκτυο ενός επιπέδου (Multinomial Logistic Regression) – ονοματική έξοδος

Fully connected single layer network



$$y = \text{softmax}(Wx + b)$$

y is a vector

Η softmax κανονικοποιεί τις τιμές των εξόδων σε πιθανότητες (τι πιθανότητα έχει μια είσοδος να κατηγοριοποιηθεί σε μια τιμή εξόδου

Η πόλωση θεωρείται πολλές φορές σαν μια επιπλέον είσοδος με τιμή 1, και βάρος b.

Softmax

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k} \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{k} \exp(z_i)}, \ldots, \frac{\exp(z_k)}{\sum_{i=1}^{k} \exp(z_i)} \right]$$

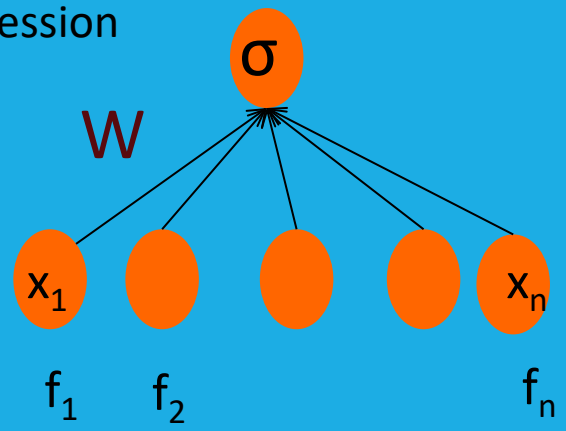
$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)} \quad 1 \leq i \leq k$$

ΝΔ για ταξινόμηση: Ανάλυση Συναισθήματος

Η είσοδος

Var	Definition
x_1	count(positive lexicon) $\in$ doc)
x_2	count(negative lexicon) $\in$ doc)
x_3	$\begin{cases} 1 & \text{if } \text{“no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	log(word count of doc)

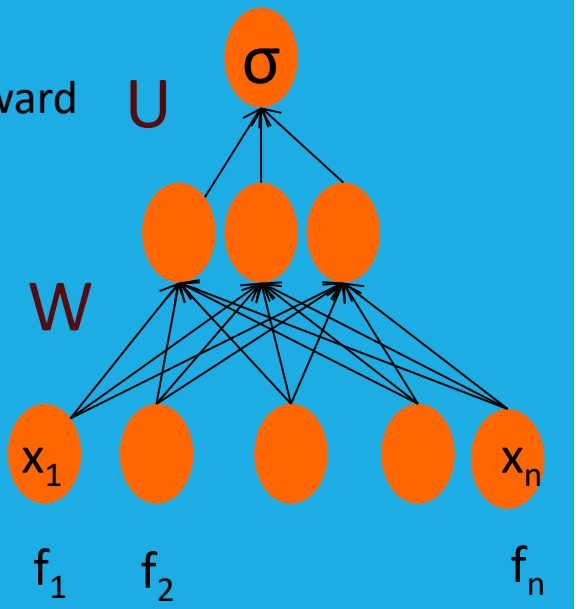
Logistic Regression



12

Η έξοδος είναι γραμμικός συνδυασμός των εισόδων

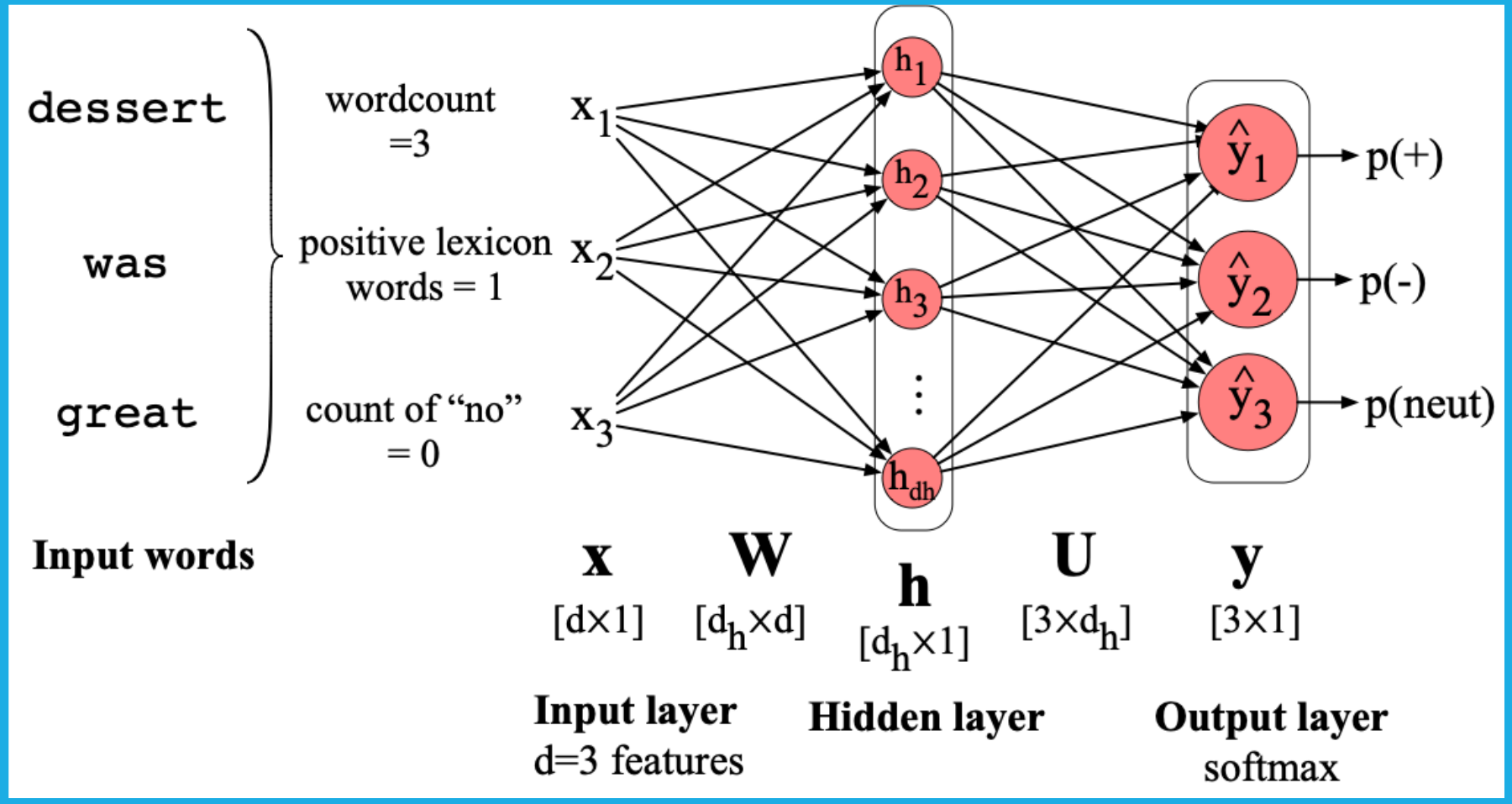
2-layer feedforward network



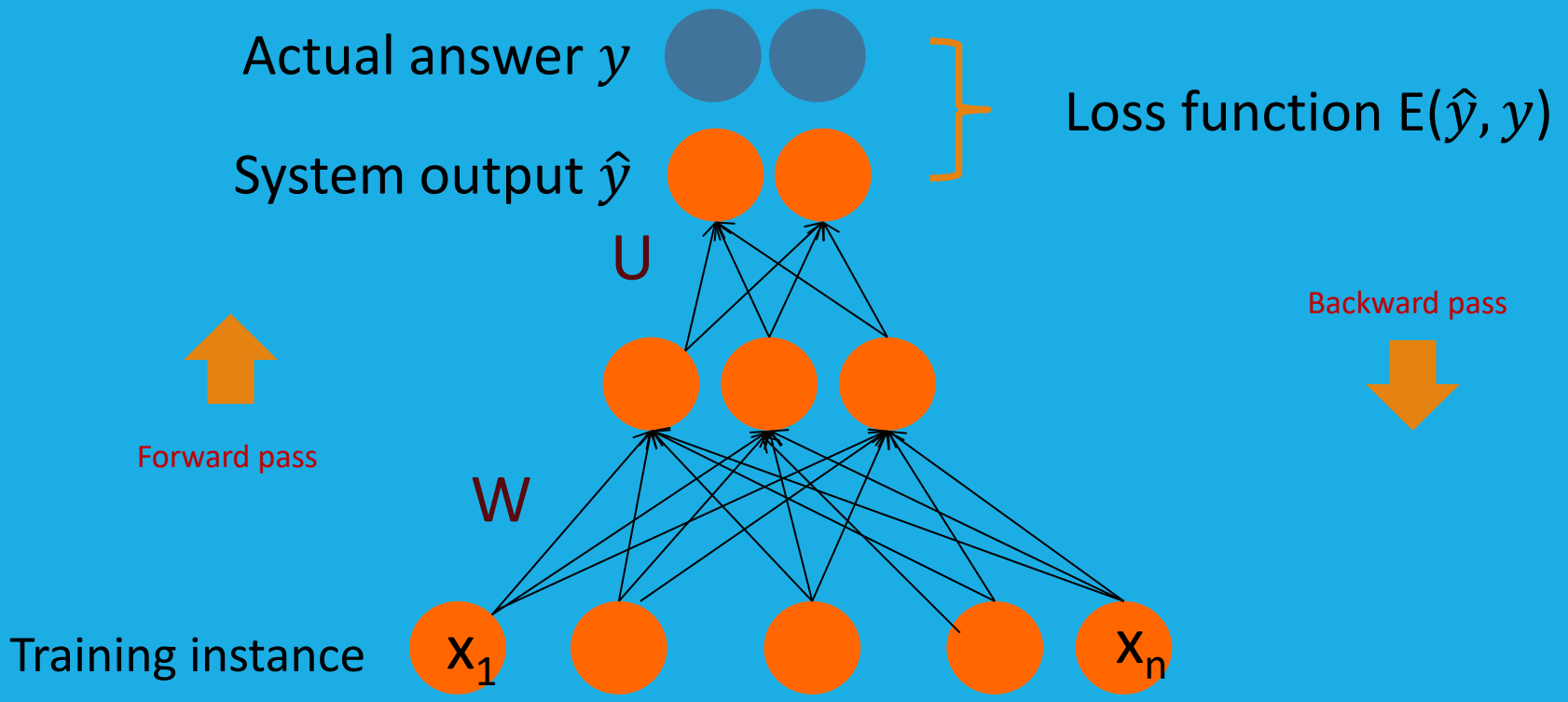
12

Προσθέτοντας ένα κρυφό επίπεδο μπορεί το δίκτυο να αναγνωρίσει και μη γραμμικές εξαρτήσεις ανάμεσα στις εισόδους

ΝΔ για ταξινόμηση: Ανάλυση Συναισθήματος



ΝΔ: Εκπαίδευση



Παράδειγμα συνάρτησης σφάλματος E($\hat{y}, y$): cross entropy loss = $-[y \log \hat{y} + (1-y) \log(1-\hat{y})]$ =

$$-[y \log \sigma(w \cdot x + b) + (1-y) \log(1-\sigma(w \cdot x + b))]$$

ΝΔ: Back Propagation

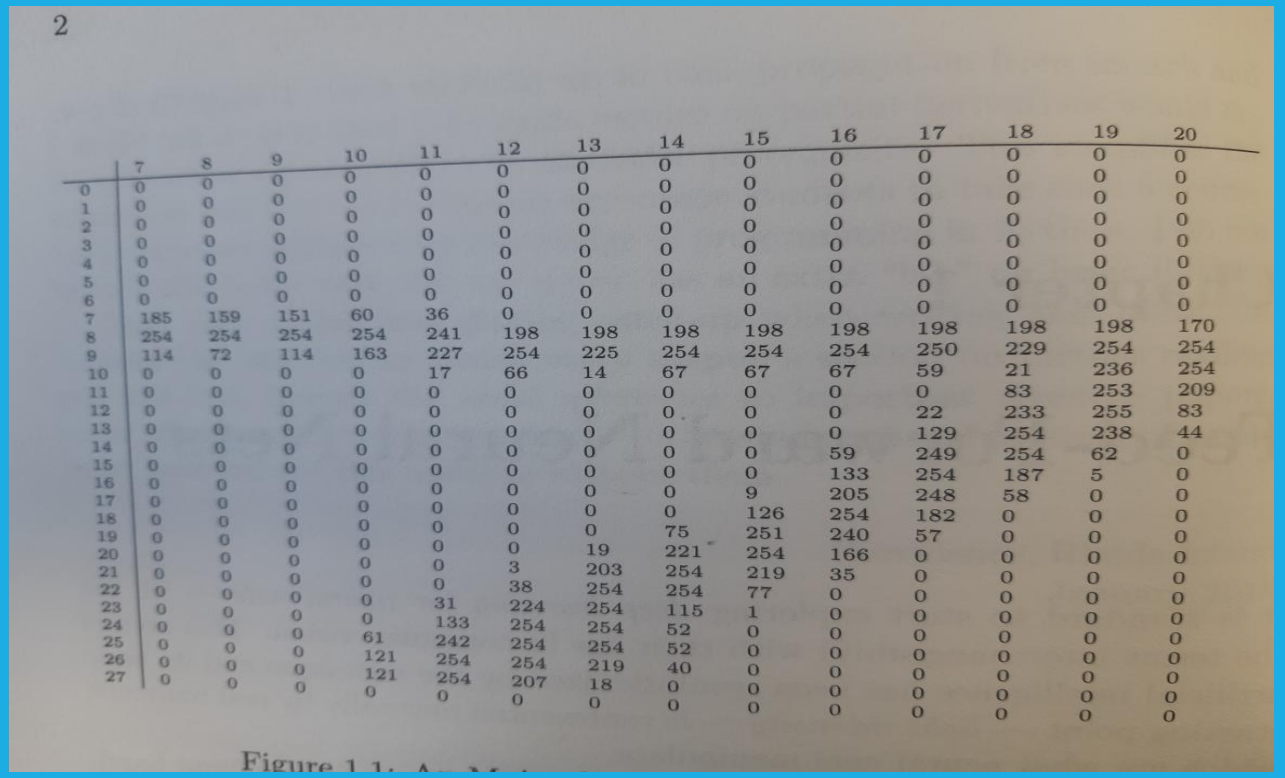
- Αρχικά τα βάρη παίρνουν τυχαίες τιμές
- Υπολογίζονται οι έξοδοι του τελευταίου επιπέδου
- Υπολογίζεται η διαφορά (σφάλμα Ε) αυτών των εξόδων από τις επιθυμητές εξόδους
- Παίρνουμε την παράγωγο του σφάλματος
- Τα καινούρια βάρη υπολογίζονται από το τελευταίο επίπεδο προς τα πίσω από τα προηγούμενα βάρη, βάσει της σχέσης

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta}$$

η παράγωγος της συνάρτησης σφάλματος ως προς τη μεταβλητή του βάρους

- όπου α ένας παράγοντας που ονομάζεται **learning rate**
- μικρό α: κάθε εποχή (επανάληψη) προκαλεί μικρή μεταβολή στα βάρη – χρειάζονται περισσότερες επαναλήψεις (υπερβολικά μικρό α μπορεί να «κολλήσει» την διαδικασία)
- μεγάλο α: μεγάλες μεταβολές στα βάρη, λιγότερες επαναλήψεις (υπερβολικά μεγάλο α μπορεί να φέρει πολύ γρήγορα σύγκλιση σε τοπικά βέλτιστα – suboptimal solution)

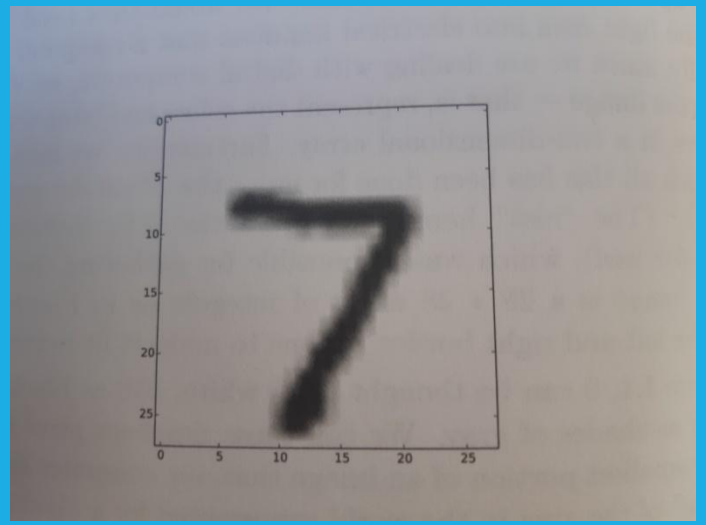
Παράδειγμα με νούμερα
(*Introduction to Deep Learning*, Eugene Charniak)



2

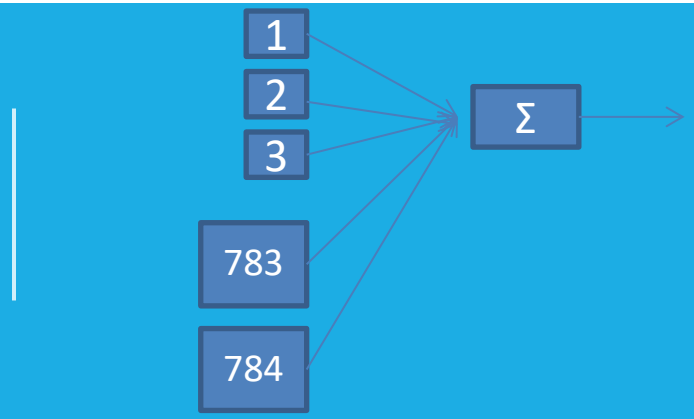
	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	185	159	151	60	36	0	0	0	0	0	0	0	0	0
8	254	254	254	254	241	198	198	198	198	198	198	198	198	170
9	114	72	114	163	227	254	225	254	254	254	250	229	254	254
10	0	0	0	0	17	66	14	67	67	67	59	21	236	254
11	0	0	0	0	0	0	0	0	0	0	0	83	253	209
12	0	0	0	0	0	0	0	0	0	0	22	233	255	83
13	0	0	0	0	0	0	0	0	0	0	129	254	238	44
14	0	0	0	0	0	0	0	0	0	59	249	254	62	0
15	0	0	0	0	0	0	0	0	0	133	254	187	5	0
16	0	0	0	0	0	0	0	0	9	205	248	58	0	0
17	0	0	0	0	0	0	0	0	126	254	182	0	0	0
18	0	0	0	0	0	0	0	75	251	240	57	0	0	0
19	0	0	0	0	0	0	19	221	254	166	0	0	0	0
20	0	0	0	0	0	3	203	254	219	35	0	0	0	0
21	0	0	0	0	0	38	254	254	77	0	0	0	0	0
22	0	0	0	0	31	224	254	115	1	0	0	0	0	0
23	0	0	0	0	133	254	254	52	0	0	0	0	0	0
24	0	0	0	61	242	254	254	52	0	0	0	0	0	0
25	0	0	0	121	254	254	219	40	0	0	0	0	0	0
26	0	0	0	121	254	207	18	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0

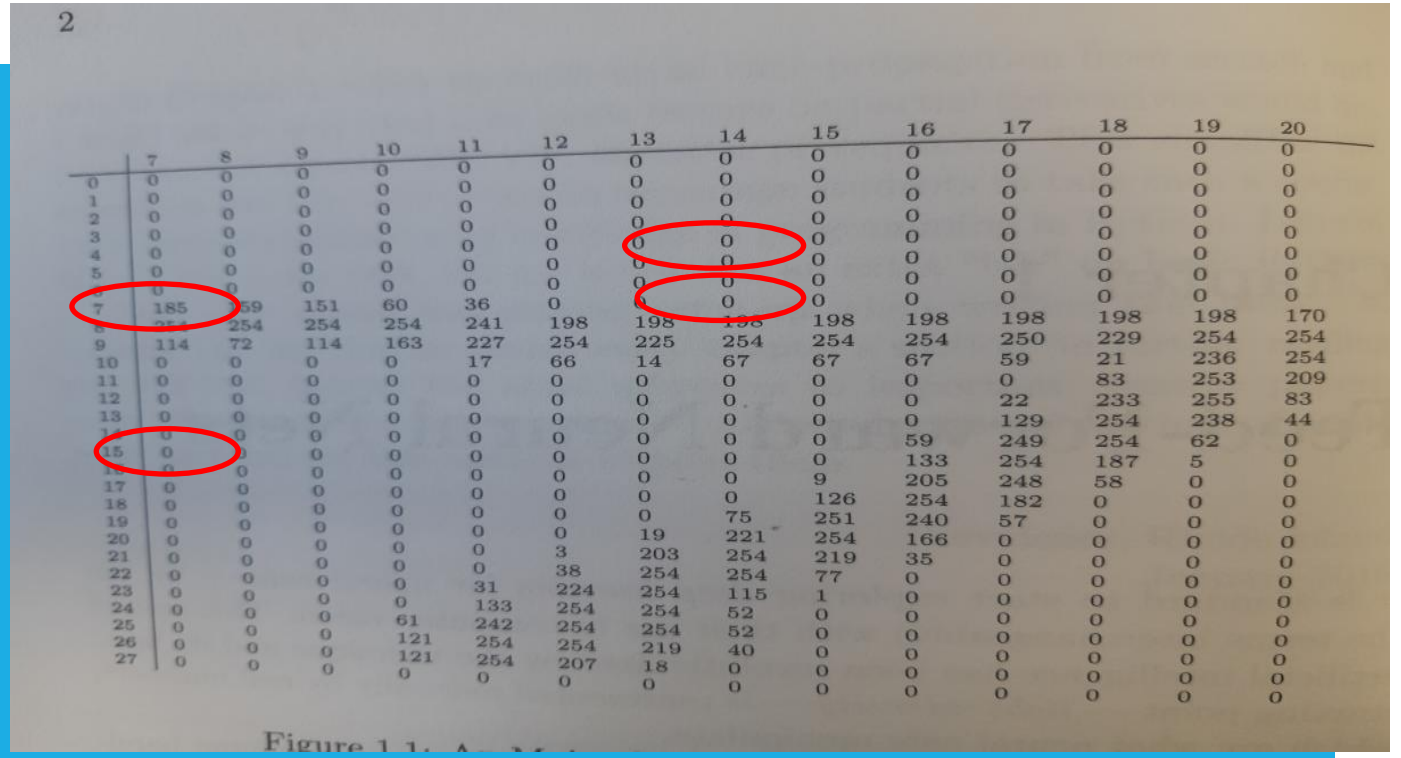
Figure 1.1:



Δεξιά: η εικόνα του χειρόγραφου ψηφίου 7

Αριστερά: ένας πίνακας 28x28 ακεραίων (οι στήλες δεν φαίνονται όλες γιατί οι αριστερές και οι δεξιές είναι λευκό περιθώριο) που απεικονίζουν την ένταση των pixel στην απεικόνιση του 7 στην εικόνα δεξιά





Έστω ένα δίκτυο που θα αποφασίζει αν μια εικόνα είναι το ψηφίο 0 ή όχι.

y=1, αν η είσοδος είναι το ψηφίο 0

y=0, αλλιώς

Η είσοδος είναι 784 (28x28) κόμβοι, και η έξοδος 1 κόμβος.

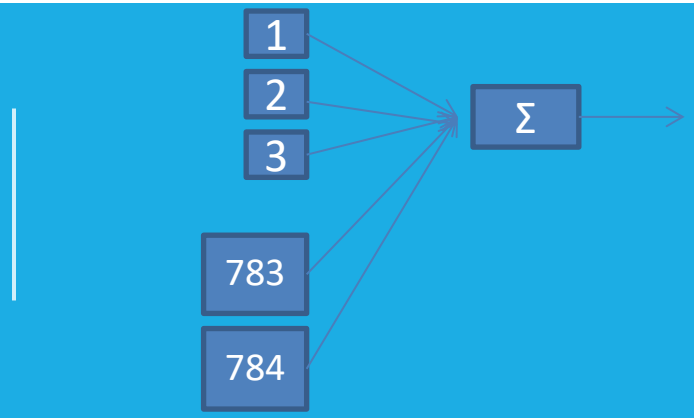
Έχω βηματική (step) activation function. Αν b+Σ(wx) >0 τότε f(x)=1, αλλιώς f(x)=0.

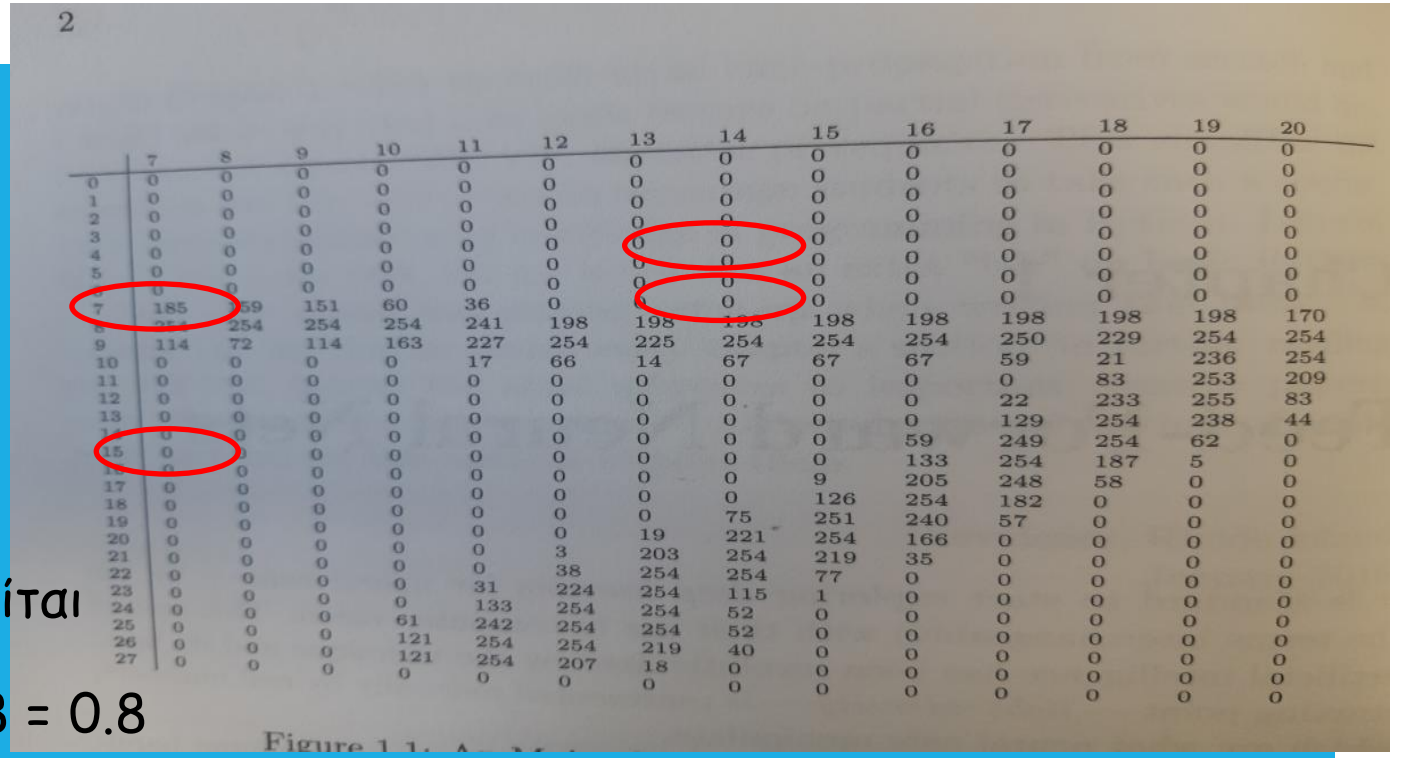
Αρχικά όλα τα βάρη (w) και η πόλωση (b) έχουν τιμές 0. (Η πόλωση είανι μια επιπλέον είσοδος με τιμήγ 1 και βάρος b).

Στο παράδειγμα θα εστιάσουμε στα pixel [7,7], [7,14], [14,7], [4, 14]

Διαιρούμε τις τιμές των κελιών με το 255, ώστε να κυμαίνονται ανάμεσα στο πεδίο [0,1].

Έστω ότι έχω μια εικόνα του ψηφίου 0, και οι τιμές των παραπάνω κελιών είναι 0.8, 0.9, 0.6 και 0 αντίστοιχα. Η έξοδος θέλω να είναι y=1, μια και το ψηφίο εισόδου μου είναι το 0.





Αρχικά f(x)=wx+b=0, οπότε η έξοδος υπολογίζεται 0, άρα η εικόνα μου ταξινομείται λανθασμένα και y – f(x) =1-0=1.

$W_{7,7\ new}=W_{7,7\ old} + (y-f(x))*x_{7,7} = 0 + 1*0.8 = 0.8$

$W_{7,14\ new}=W_{7,14\ old} + (y-f(x))*x_{7,14} = 0 + 1*0.9 = 0.9$

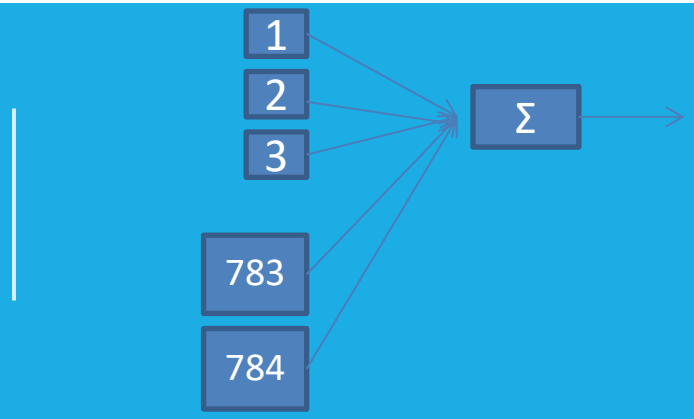
$W_{14,7\ new}=W_{14,7\ old} + (y-f(x))*x_{14,7} = 0 + 1*0.6 = 0.6$

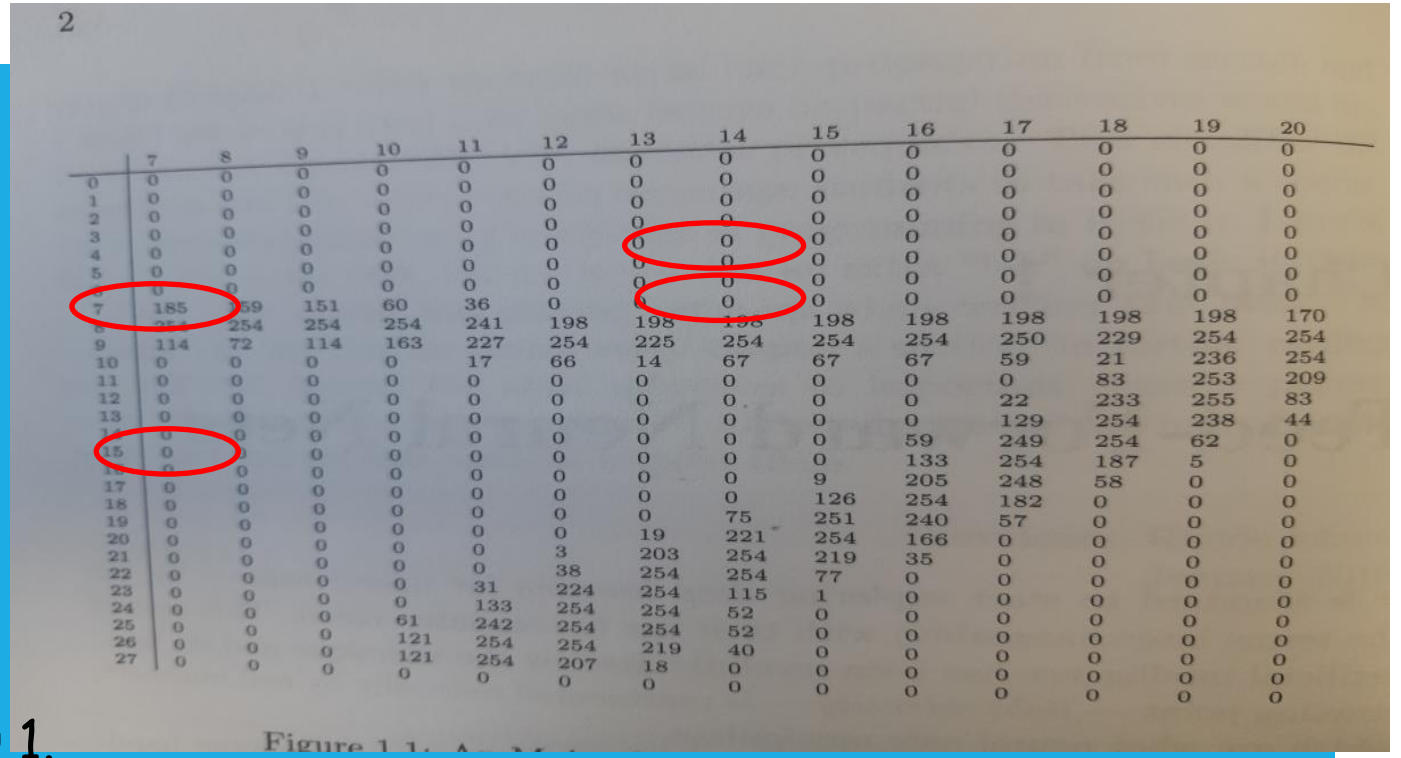
$W_{4,14\ new}=W_{4,14\ old} + (y-f(x))*x_{4,14} = 0 + 1*0 = 0$

Η παραπάνω απλοϊκή σχέση για την ανανέωση των βαρών βασίζεται στην zero-one loss function, (η οποία δεν παραγωγίζεται), και έχει στόχο (δεδομένου ότι οι είσοδοι x είναι θετικοί αριθμοί)

- Να μικραίνει τα βάρη όταν θέλω στην έξοδο 0, αλλά παίρνω 1 (y-f(x)=-1)
- Να μεγαλώνει τα βάρη όταν θέλω στην έξοδο 1, αλλά παίρνω 0 (y-f(x)=1)
- Να αφήνει τα βάρη ίδια όταν δεν γίνεται λάθος

Το βάρος της πόλωσης γίνεται $w_{b\ new}= w_{b\ old} +(y-f(x))*1=1$





Έστω ότι τώρα έρχεται μια εικόνα του ψηφίου 1.

$x_{7,7}=0$

$x_{7,14}=1$

$x_{14,7}=0$

$x_{4,14}=1$

$w_{7,7}x_{7,7} = 0.8*0$

$w_{7,14}x_{7,14} = 0.9*1$

$w_{14,7}x_{14,7} = 0.6*0$

$w_{4,14}x_{4,14} = 0*1$

b+Σ(wx) = 1+ 0.9 = 1.9 >0, άρα f(x) = 1. Η εικόνα κατατάσσεται λανθασμένα ότι είναι το ψηφίο 0.

y-f(x) = 0-1 = -1.

Επαναϋπολογίζονται τα βάρη.