Βελτιστοποίηση Συστημάτων & Υδροπληροφορική

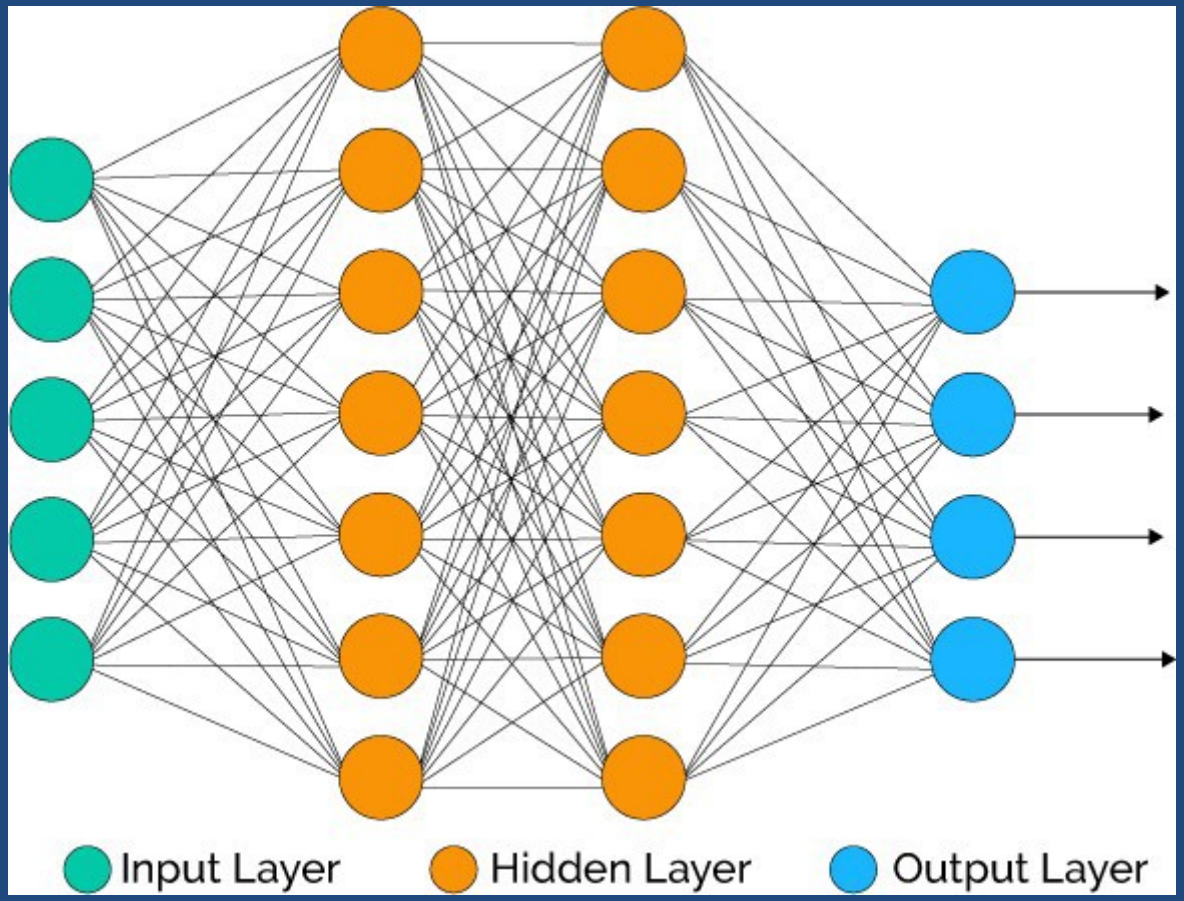
Νευρωνικά Δίκτυα

Χρήστος Μακρόπουλος
Τομέας Υδατικών Πόρων και Περιβάλλοντος
Εθνικό Μετσόβιο Πολυτεχνείο

Νευρωνικά Δίκτυα

- Βασίζονται στη χρήση μιας προσέγγισης (μάλλον απλοϊκής) του τρόπου λειτουργίας του νευρικού συστήματος μέσω μαθηματικών συναρτήσεων.
- Πρώτη αναφορά: *W.S. McCulloh & W. Pitts, "A logical calculus of ideas immanent in Nervous activity", Bull. Mathematical Biophysics, Vol. 5, 1943, pp.115-133*.

Μια από τις πρώτες (πετυχημένες) απόπειρες “machine learning”



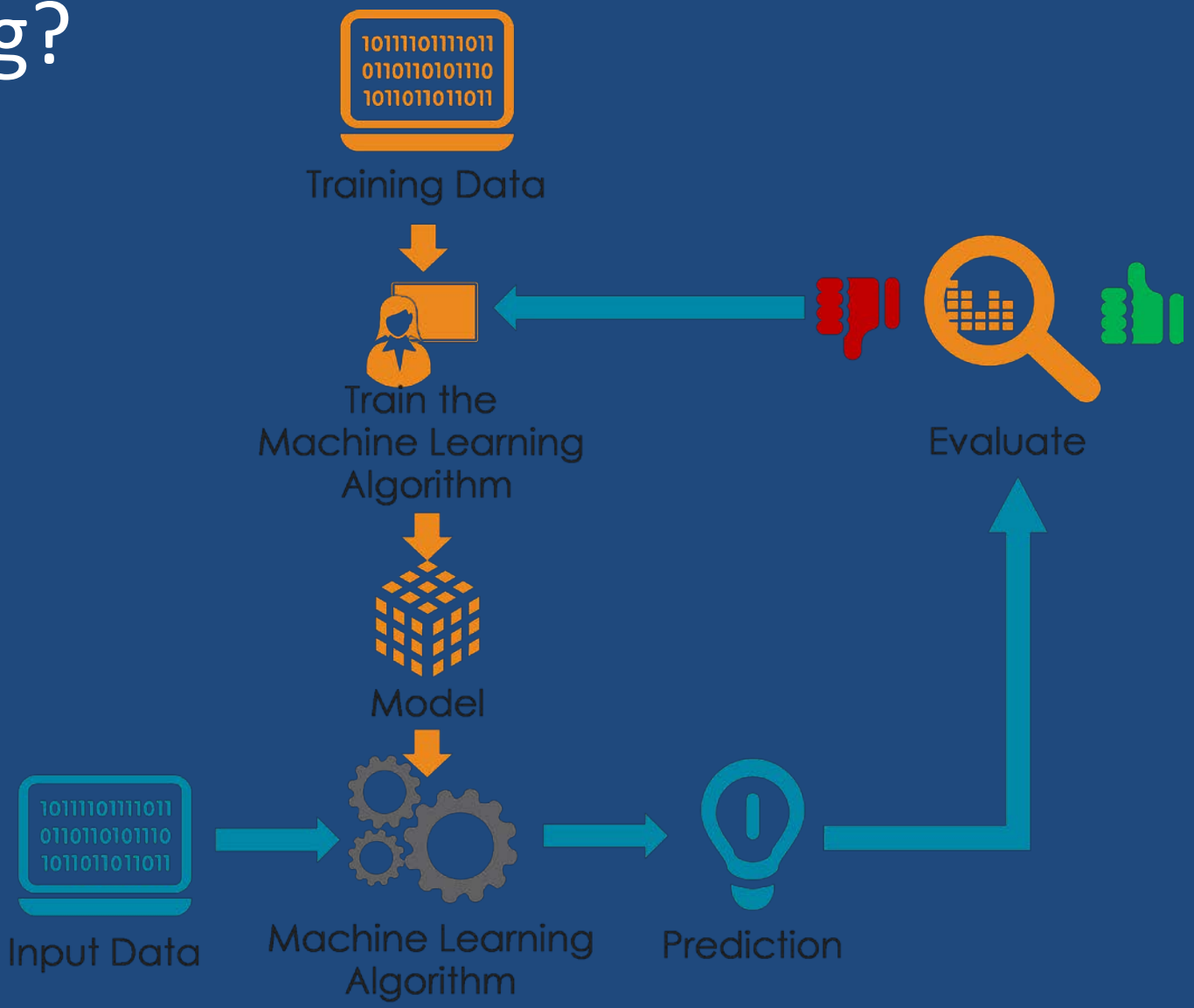
Τί είναι machine learning?

Η προσπάθεια να φτιάξουμε αλγόριθμους που να έχουν τις δεξιότητες του ανθρώπινου εγκεφάλου!

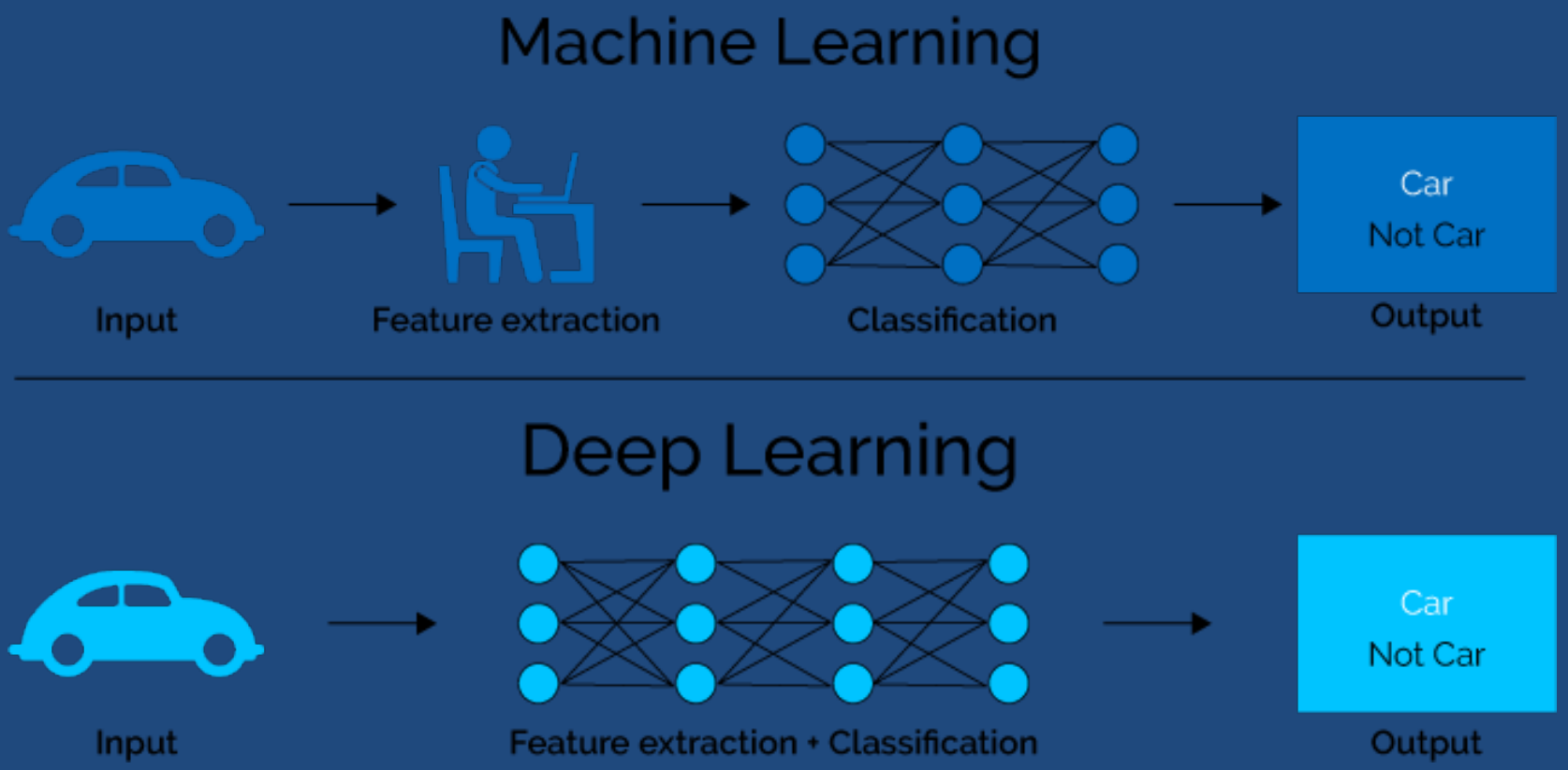
Πχ. Βλέπω, αναγνωρίζω, μαθαίνω, συμπεραίνω, συνθέτω…

elementary my dear . . .

Ποια η γενική διαδικασία του Machine Learning?



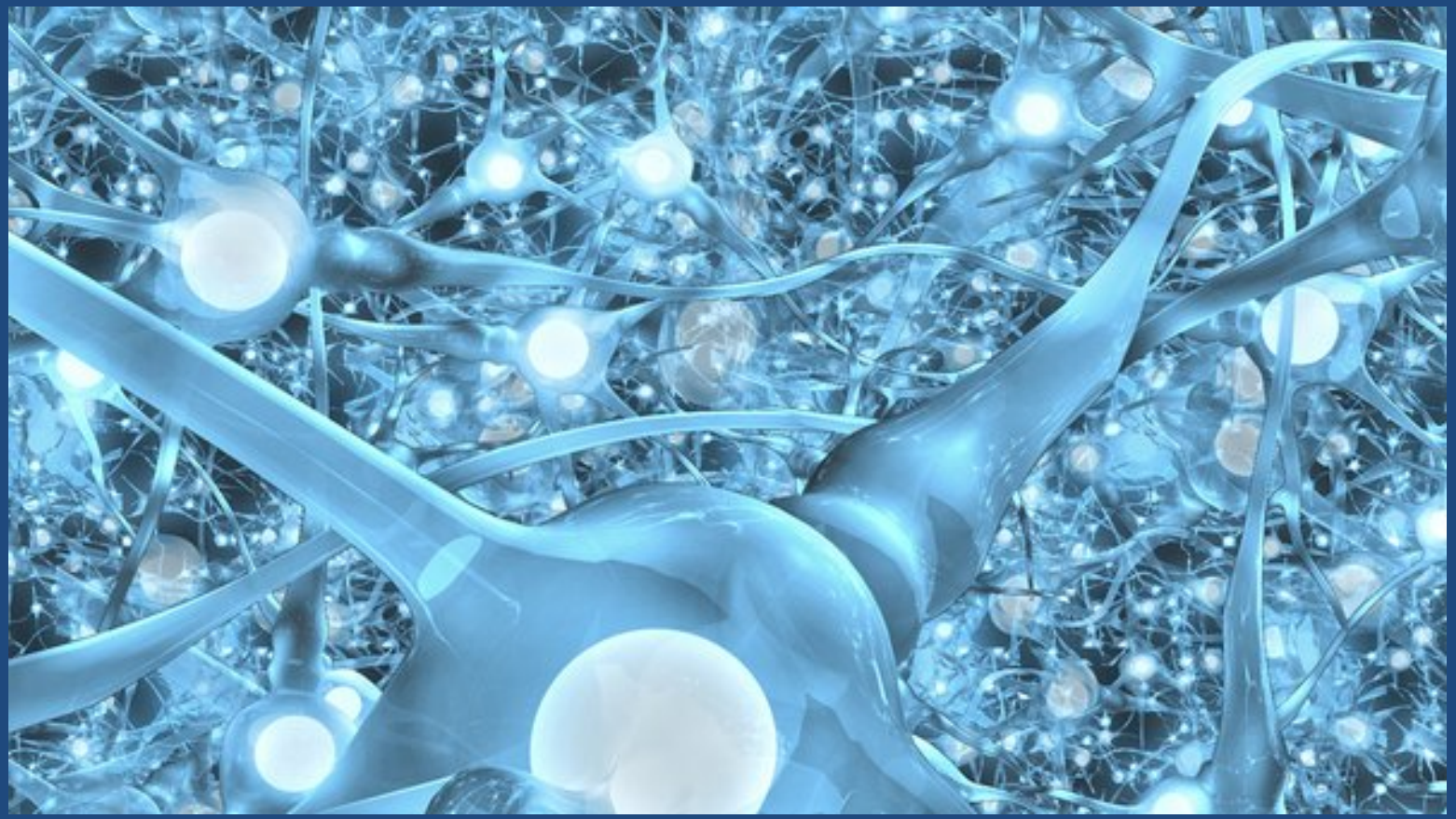
Και σε τί διαφοροποιείται από τον άλλο σημαντικό όρο της εποχής (το **deep learning**)



Ποια η βασική υπόθεση εργασίας στην περίπτωση των ΝΔ;

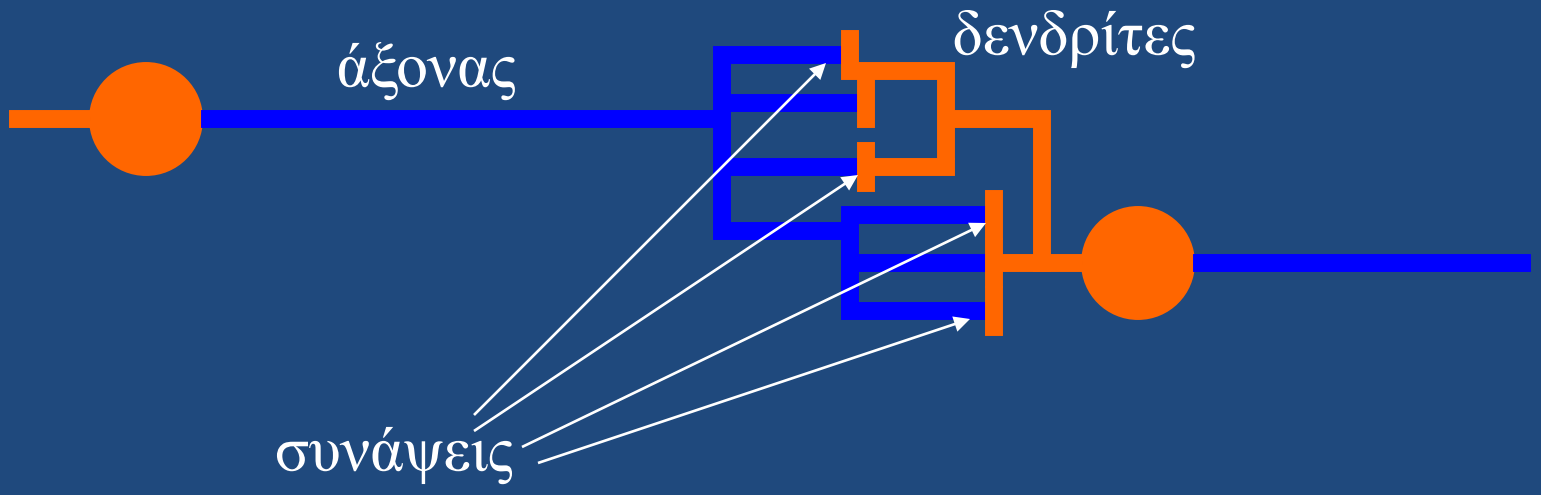
- Βασίζεται στην εξής (απλή και μάλλον αισθητικής φύσης) υπόθεση εργασίας:
  - Γεγονός: οι άνθρωποι μαθαίνουν χρησιμοποιώντας τους νευρώνες τους (εγκέφαλος)
  - Στόχος: θέλουμε να κάνουμε τους υπολογιστές να συμπεριφέρονται όπως ένας εγκέφαλος.
  - Άρα: φτιάχνω ένα σύστημα που **μοιάζει** με εγκέφαλο (δηλ. ένα σύνολο νευρώνων) και **ελπίζω** οτι θα συμπεριφέρεται με τον ίδιο τρόπο...

Το βιολογικό πρότυπο



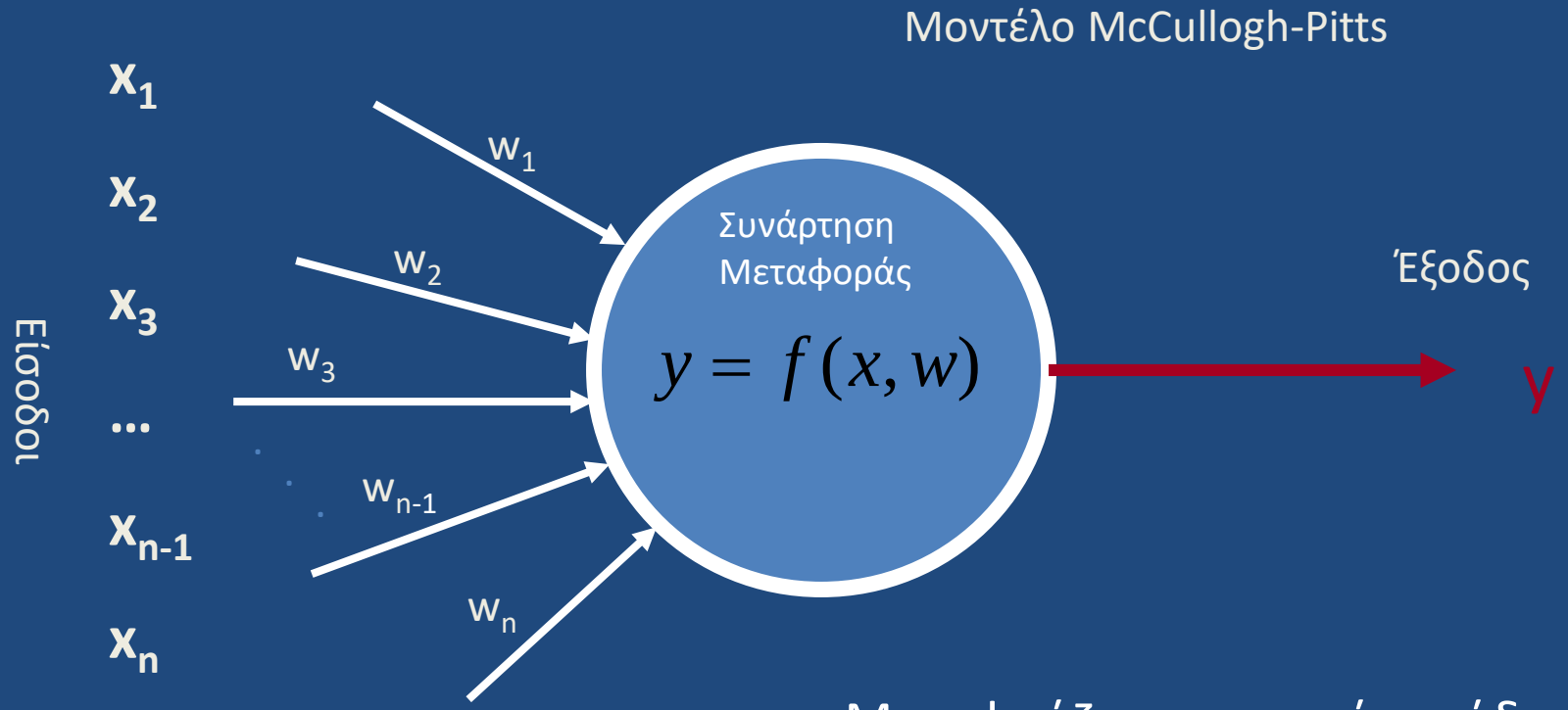
Δενδρίτες & Συνάψεις: προσοχή – όπως και στους γενετικούς εργαζόμαστε σε αρκετά **αφαιρετικό** επίπεδο σε σχέση με την αρχική βιολογική ιδέα

Το (απλοποιημένο) βιολογικό πρότυπο



- Τα **σήματα** ταξιδεύουν μέσω του νευρώνα και προκαλούν την έκκριση νευρο-διαβιβαστικών **ουσιών** στις συνάψεις.
- Οι νευρο-διαβιβαστικές ουσίες προκαλούν **αντιδράσεις** στους δενδρίτες των νευρώνων που βρίσκονται μετά τις συνάψεις.
- Οι αντιδράσεις αυτές προκαλούν **σήματα** στους νευρώνες μετά τις συνάψεις.
- Το **μέγεθος/ένταση** του σήματος εξαρτάται από τη **δύναμη** του συνδέσμου της σύναψης.

Τεχνητός Νευρώνας (ή αντιληπτήρας)

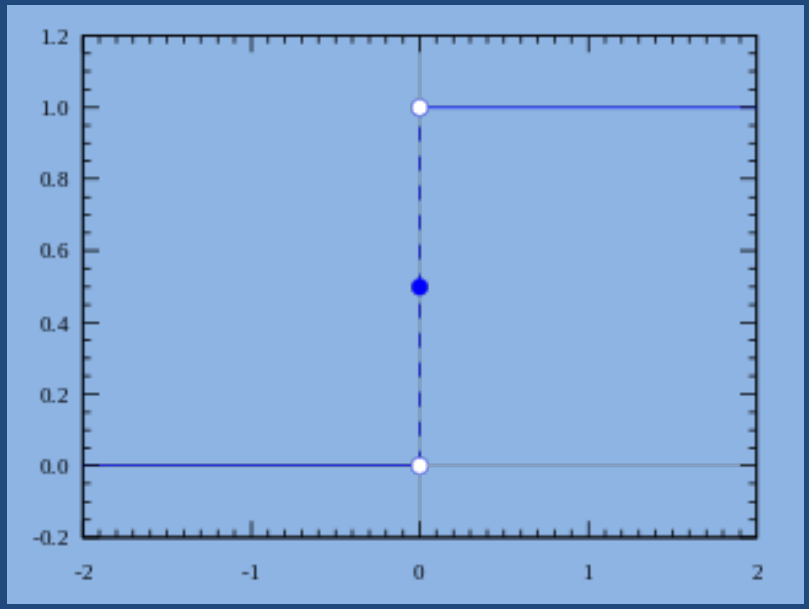


- Μεταφράζει μια σειρά εισόδων σε μια έξοδο με τη βοήθεια:
  - βαρών και
  - μιας συνάρτησης «μεταφοράς» (ή αλλιώς «ενεργοποίησης»)

Παράδειγμα συνάρτησης μεταφοράς: Step Function

Έξοδος:

$$y = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{if } u < \theta \end{cases}$$



Εδώ με θ = 0

Το θ ονομάζεται κατώφλι ενεργοποίησης

Είσοδος:

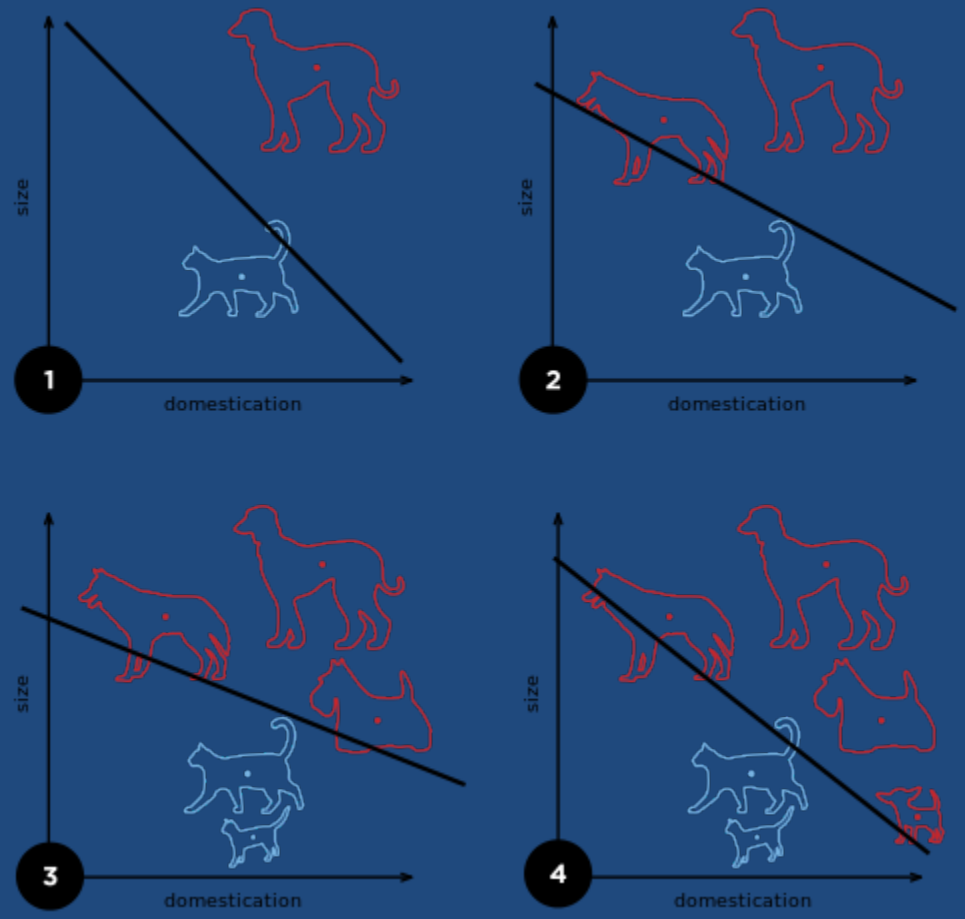
$$u = \sum_{i=1}^{n} w_i x_i$$

Ενεργοποίηση του νευρώνα

- Συνεχίζοντας με την βιολογική αναλογία, ο νευρώνας ενεργοποιείται μόνο όταν το «σήμα» είναι αρκετα «δυνατό».
- Αυτό προκύπτει:
  - Είτε από τη **μορφή της συνάρτησης** ενεργοποίησης (πχ. step function threshold)
  - Είτε μετά από σύγκριση με μια «**συνθήκη ενεργοποίησης**»

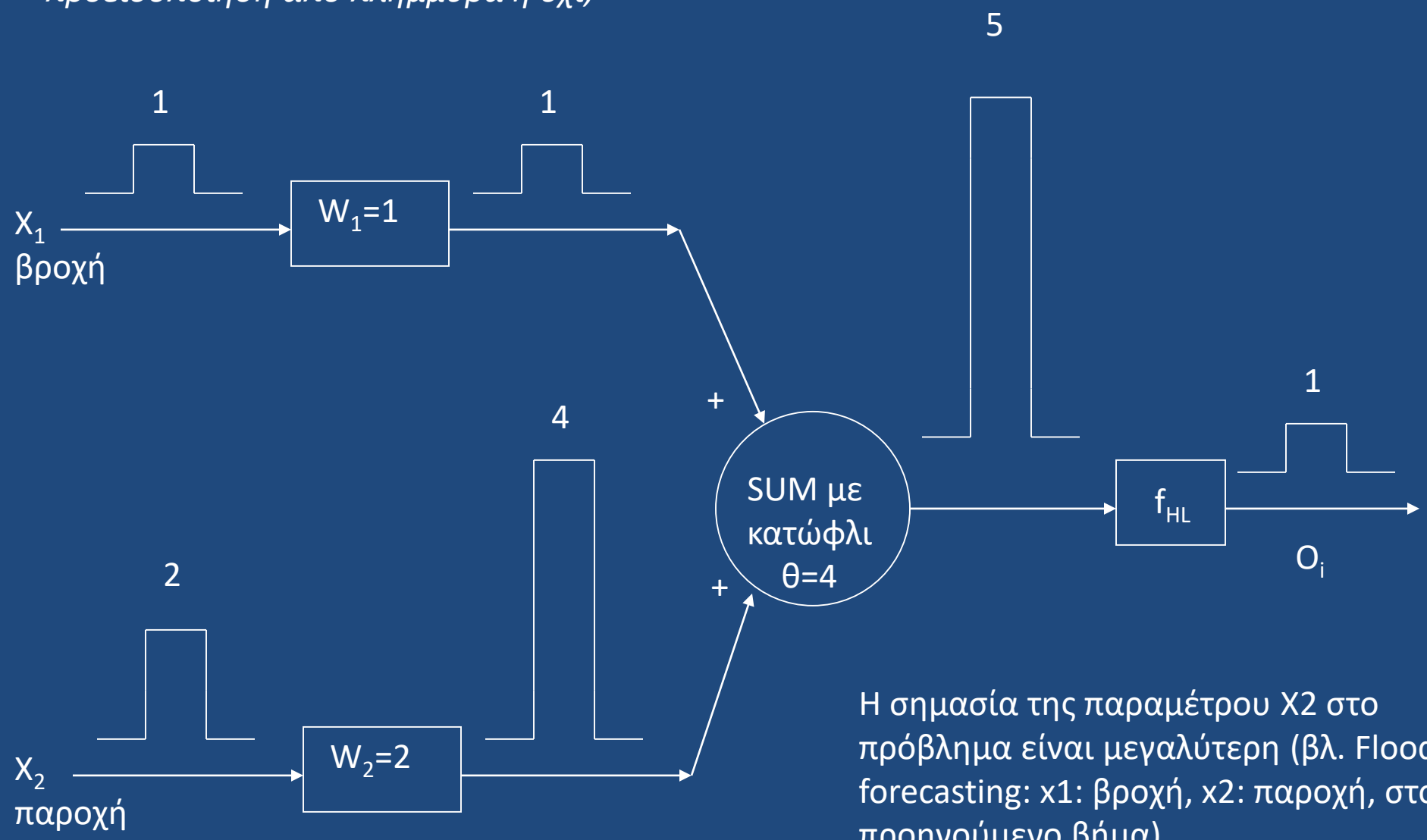
Στην ουσία τι κάνει ένας αντιληπτήρας με αυτή τη (απλή) συνάρτηση μεταφοράς;

- Μπορεί να πεί Ναι/Οχι ή αλλιώς **Ανήκει/Δεν Ανήκει**.
- ‘Αρα μπορεί να «αποφασίσει» αν ένα δείγμα (με κάποια χαρακτηριστικά) ανήκει σε ένα σύνολο ή όχι χωρίζοντας το δείγμα σε δυο μέρη.
- Πως ξέρει ποιό δείγμα ανήκει σε ποιό μέρος; του δίνουμε εμείς ένα σετ **σωστών** απαντήσεων και προσαρμόζουμε τα βάρη έτσι ώστε να απαντά **σωστά**.



Ένα παράδειγμα πιο κοντά στα δικά μας:

πχ σύστημα έγκαιρης προειδοποίησης πλημμυρών: *Να στείλω SMS για έγκαιρη προειδοποίηση από πλημμύρα ή όχι;*



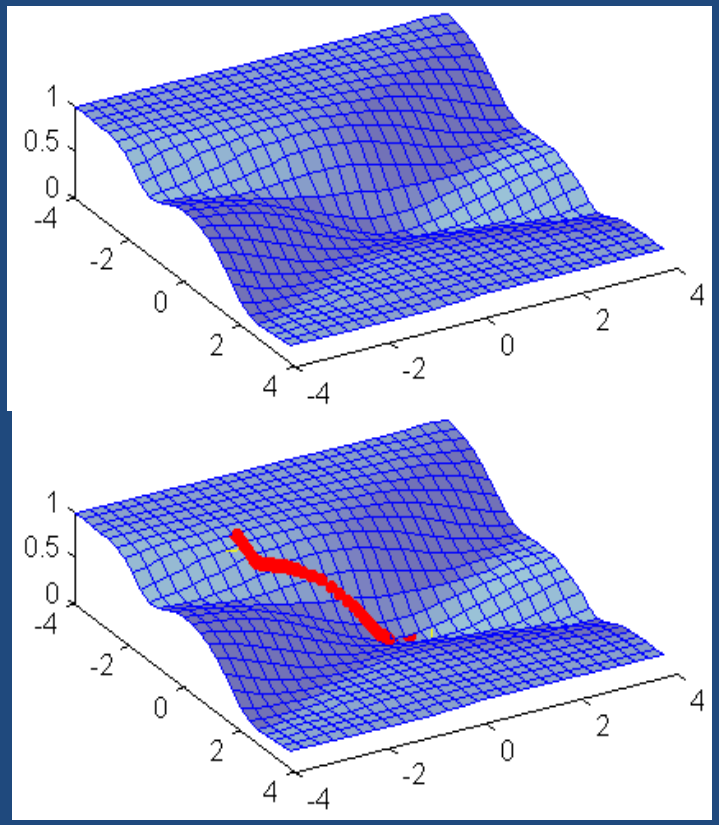
Η σημασία της παραμέτρου X2 στο πρόβλημα είναι μεγαλύτερη (βλ. Flood forecasting: x1: βροχή, x2: παροχή, στο προηγούμενο βήμα)

Πώς όμως υπολογίζουμε τα βάρη; Εκπαίδευση

- Διαδικασία η οποία με δεδομένα «σωστά» **σύνολα εισόδων-εξόδων**, τροποποιεί τις μεταβλητές του ΝΔ (πχ τα βάρη) για να αναγκάσει το ΝΔ να δώσει τη «σωστή» απάντηση εξόδου.
- Στην ουσία: ένα πρόβλημα **βαθμονόμησης** σε ένα «γενικό» μοντέλο.
- Τύποι εκπαίδευσης: Αυτόνομη, Μη Αυτόνομη, Επιβεβλημένη, Ανταγωνιστική, Κανόνας Δέλτα (LMS), Hill-Climbing

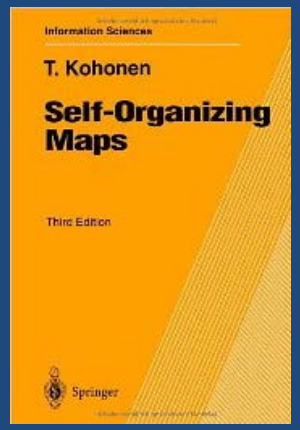
Εποπτευόμενη Εκπαίδευση (supervised training)

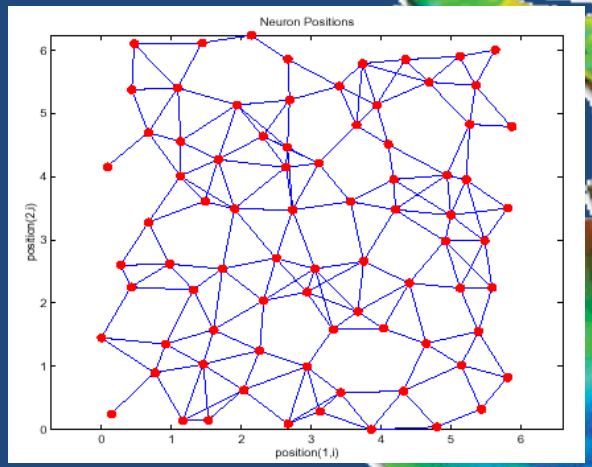
- Κατά την εποπτευόμενη εκπαίδευση η απόκριση του ΝΔ σε ένα σύνολο εισόδου συγκρίνεται με την **επιθυμητή απόκριση**.
- Αν αυτές διαφέρουν τότε το ΝΔ υπολογίζει ένα **σφάλμα**.
- Το σφάλμα χρησιμοποιείται ώστε να γίνουν οι απαραίτητες **τροποποιήσεις των συντελεστών βάρους**, ώστε η απόκριση να πλησιάσει την επιθυμητή και το σφάλμα να προσεγγίσει ή δυνατόν το μηδέν.
- Η τροποποίηση γίνεται με κάποιο **βήμα** (βλ. κάναβο δεξιά) που ονομάζεται **learning rate**
- Οι χρησιμοποιούμενες μέθοδοι για την ελαχιστοποίηση του σφάλματος είναι οι ίδιες που εφαρμόζονται σε διάφορες τεχνικές **ελαχιστοποιήσεως**

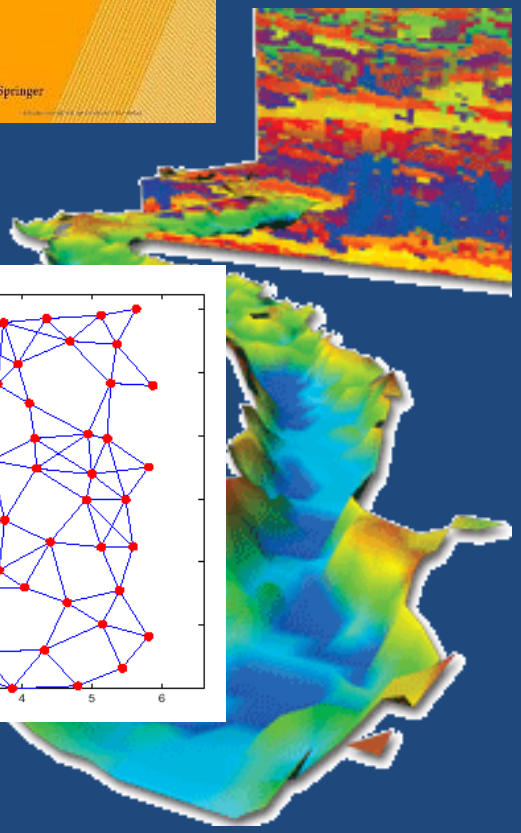


Μη εποπτευόμενη εκπαίδευση (unsupervised training)

- Δεν δίνουμε «σωστές» τιμές εξόδου και το ANN πρέπει να **αποφασίσει μόνο του** τί είναι σωστό: στην ουσία να βρεί κρυμμένες δομές.
- Self-organization: self-organizing map (SOM), Kohonen networks/maps







Περιγραφή του προβλήματος

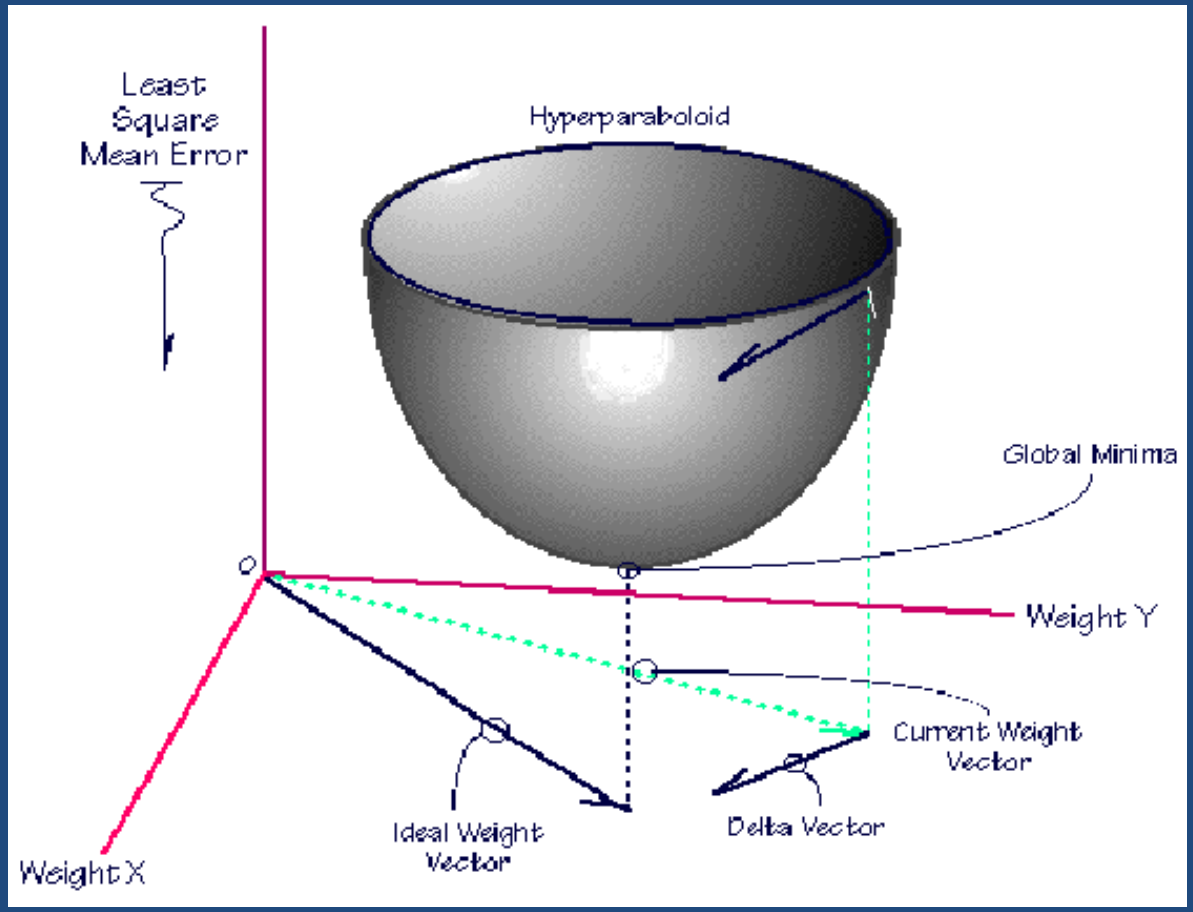
Δεδομένα:

- Σύνολα τιμών (είσοδοι, έξοδος): (x^t, y_t),
  - Για παράδειγμα ως: $y_t=g(x^t) + z_t$;
  - Όπου z_t τυχαίος θόρυβος

Πρόβλημα:

- Υπολογίστε τα βάρη [w1, w2, … wk] με τα οποία ελαχιστοποιείται το σφάλμα.
- Με άλλα λόγια: υπολογίστε ένα ΝΔ (δηλ. ένα F(x,w)) το οποίο προσομοιώνει την (άγνωστη) συνάρτηση g(x) για κάθε x.

Πρόκειται για περίπτωση βελτιστοποίησης (ελαχιστοποίησης) με μέγιστη κλιση (steepest descend)



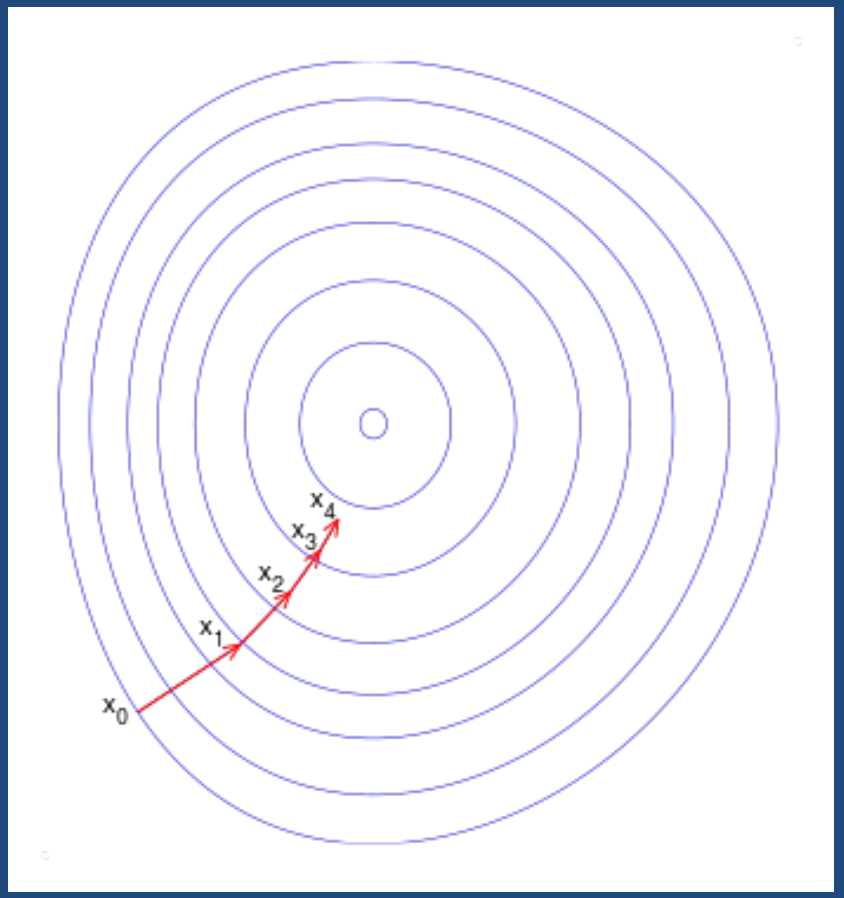
Με άλλα λόγια:

Αν ακολουθήσουμε την (αρνητική) κλίση, κατεβαίνουμε πιο γρήγορα και άρα για κάθε:

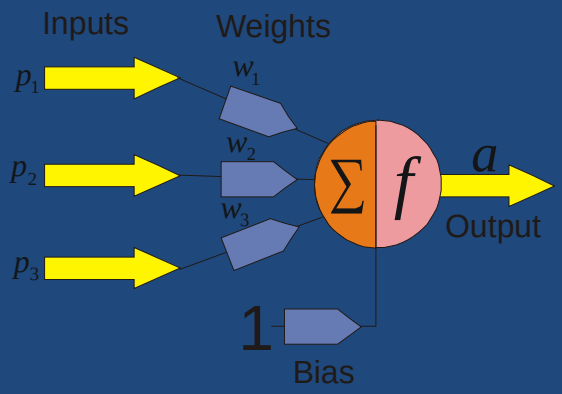
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \ n \geq 0.$$

Έχουμε πάντα:

$$F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots,$$



Steepest descent για ένα νευρώνα



$$a = f(p_1 w_1 + p_2 w_2 + p_3 w_3 + b) = f\left(\sum p_i w_i + b\right)$$

Νευρώνας: $y_{out} = w^T x$

Δεδομένα: $(x^1, y_1), (x^2, y_2), \ldots, (x^N, y_N)$

Σφάλμα: $E(t) = (y(t)_{out} - y_t)^2 = (w(t)^T x^t - y_t)^2$

Μάθηση: ελαχιστοποίηση με σταδιακή (με κάποιο βήμα) μεταβολή βαρών κατα τη μέγιστη κλίση της επιφάνειας απόκρισης:

$$w_i(t+1) = w_i(t) - c \cdot \frac{\partial E(t)}{\partial w_i} = w_i(t) - c \cdot \frac{\partial (w(t)^T x^t - y_t)^2}{\partial w_i}$$

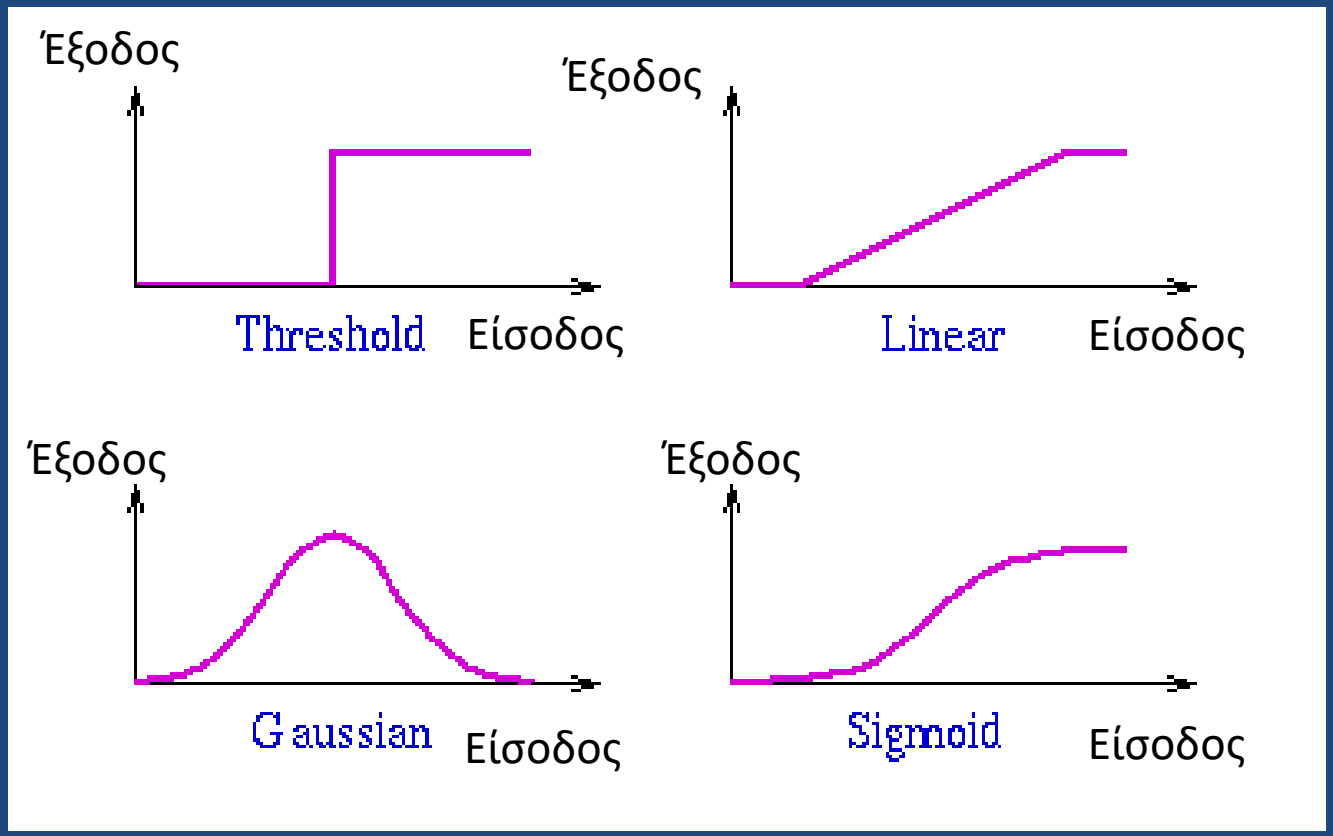
$$w_i(t+1) = w_i(t) - c \cdot (w(t)^T x^t - y_t) \cdot x_i^t$$

(απλή) εκπαίδευση ΝΔ

Threshold	Learning Rate	Sensor values		Desired Output	Weights					Response of NN
TH	LR	X1	X2	Z	w1	w2	C1	C2	S	N
							X1 x w1	X2 x w2	C1+C2	IF(S>TH,1,0)
0,50	0,20	0,00	0,00	0,00	0,10	0,30	0,00	0,00	0,00	0,00
0,50	0,20	0,00	1,00	1,00	0,10	0,30	0,00	0,30	0,30	0,00
0,50	0,20	1,00	0,00	1,00	0,30	0,50	0,30	0,00	0,30	0,00
0,50	0,20	1,00	1,00	1,00	0,50	0,70	0,50	0,70	1,20	1,00
0,50	0,20	0,00	0,00	0,00	0,50	0,70	0,00	0,00	0,00	0,00
0,50	0,20	0,00	1,00	1,00	0,50	0,70	0,00	0,70	0,70	1,00
0,50	0,20	1,00	0,00	1,00	0,50	0,70	0,50	0,00	0,50	0,00
0,50	0,20	1,00	1,00	1,00	0,70	0,90	0,70	0,90	1,60	1,00
0,50	0,20	0,00	0,00	0,00	0,70	0,90	0,00	0,00	0,00	0,00
0,50	0,20	0,00	1,00	1,00	0,70	0,90	0,00	0,90	0,90	1,00
0,50	0,20	1,00	0,00	1,00	0,70	0,90	0,70	0,00	0,70	1,00
0,50	0,20	1,00	1,00	1,00	0,70	0,90	0,70	0,90	1,60	1,00

Error	Correction	New Weights	
E	R	W1	W2
Z-N	LR x E	R+w1	R+w2
0,00	0,00	0,10	0,30
1,00	0,20	0,30	0,50
1,00	0,20	0,50	0,70
0,00	0,00	0,50	0,70
0,00	0,00	0,50	0,70
0,00	0,00	0,50	0,70
1,00	0,20	0,70	0,90
0,00	0,00	0,70	0,90
0,00	0,00	0,70	0,90
0,00	0,00	0,70	0,90
0,00	0,00	0,70	0,90
0,00	0,00	0,70	0,90

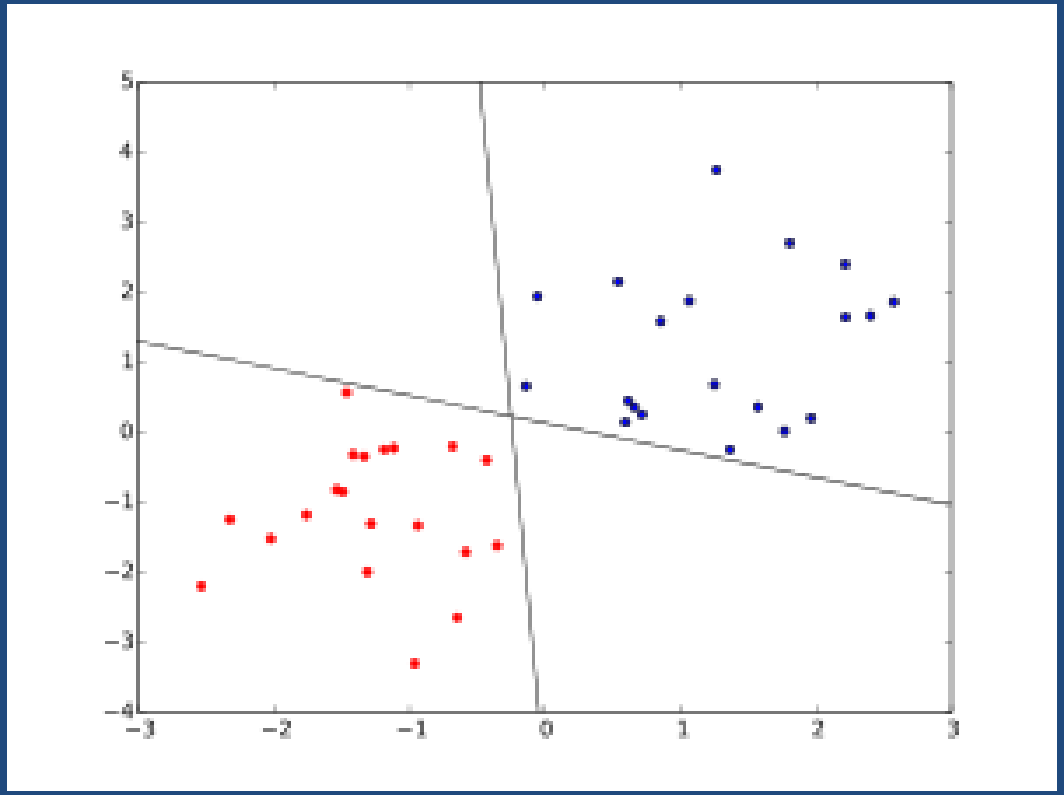
Συνηθισμένες Συναρτήσεις Μεταφοράς (ή αλλιώς Ενεργοποίησης)



- Προφανώς το αποτέλεσμα δεν χρειάζεται να είναι 0-1.
- Μπορεί να είναι και **οποιαδήποτε ενδιάμεση** τιμή
- Με τον κατάλληλο πολλαπλασιαστή μπορεί να είναι επίσης **τιμή σε οποιοδήποτε διάστημα**
- Επίσης το αποτέλεσμα μπορεί να είναι (ή να μεταφράζεται ως) **πιθανότητα**: μια Gauss συνάρτηση μεταφοράς μπορεί να θεωρηθεί συνάρτηση πυκνότητας πιθανότητας.
- Σε αυτή τη περίπτωση το αποτέλεσμα είναι;

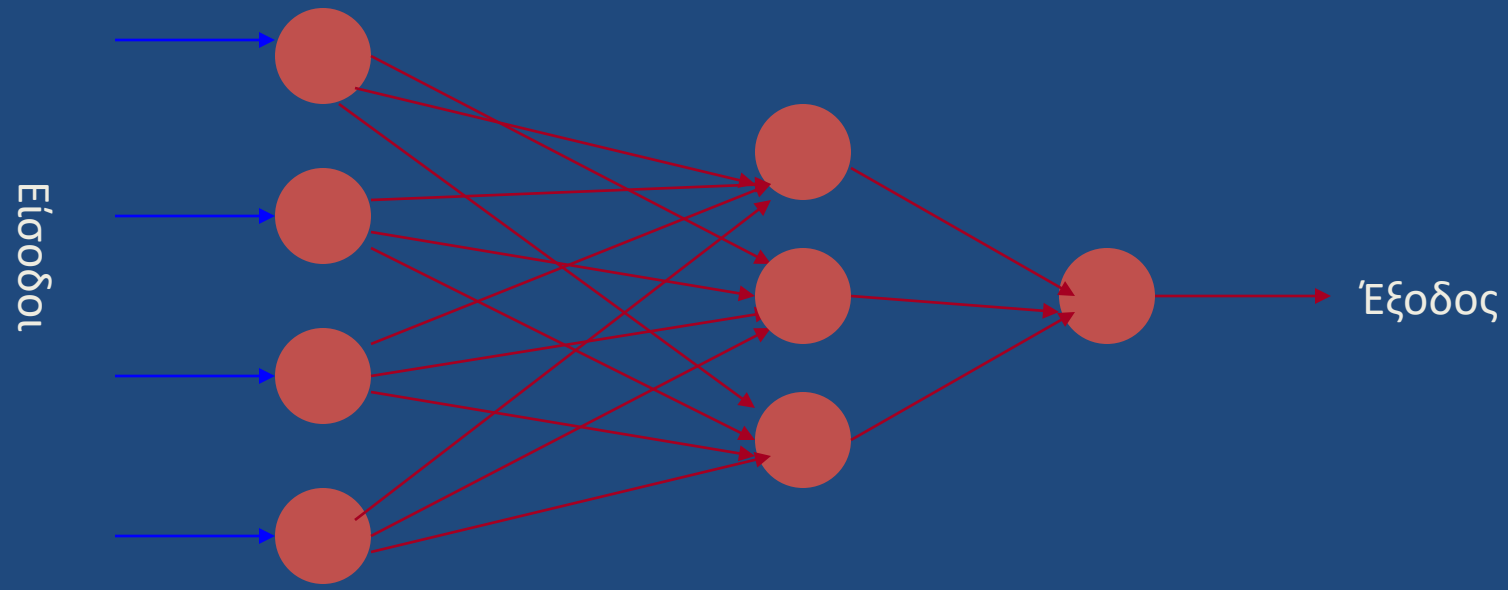
Παρόλα αυτά, τι δεν μπορεί να κάνει ποτέ ένας αντιληπτήρας (μόνος του) όσο καλά κι αν ορίσουμε τα βάρη;

- Οι αντιληπτήρες είναι γραμμικοί διαχωριστές και κατα συνέπεια δεν μπορούν να διαχωρίσουν κατηγορίες που δεν διαχωρίζονται γραμμικά
- πχ περνώντας μια ευθεία/ένα επίπεδο κτλ ανάμεσα στις κατηγορίες



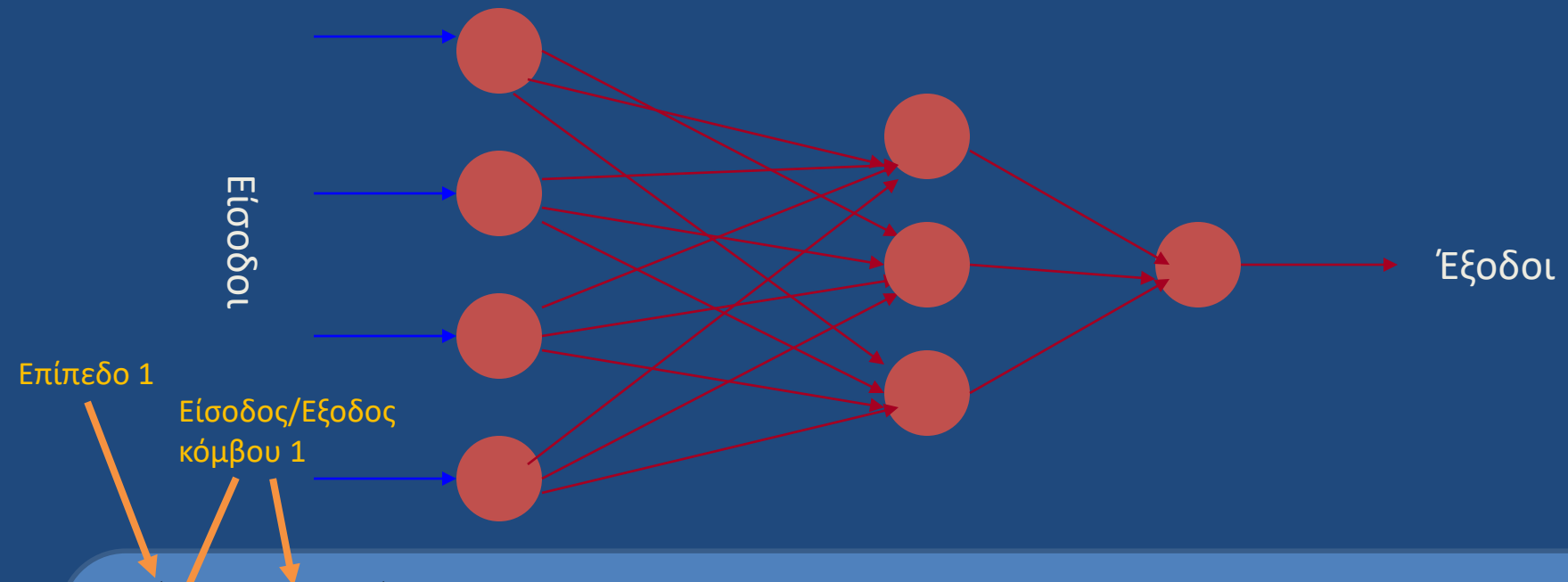
Νευρωνικά Δίκτυα

Σε ποιο σύνθετα προβλήματα μπορώ βάλω πολλούς νευρώνες (αντιληπτήρες) μαζί σε ένα δίκτυο:



Ένα ΝΔ αποτελείται από **πολλούς νευρώνες** οι οποίοι συνδέονται μέσω μιας συγκεκριμένης αρχιτεκτονικής. Στόχος είναι η «μετάφραση» των εισόδων σε συγκεκριμένες «σωστές» εξόδους (μέσω – κυρίως - του προσδιορισμού βαρών).

Συναρτήσεις ΝΔ



$$y_1^1 = f(x_1, w_1^1)$$
$$y_2^1 = f(x_2, w_2^1)$$
$$y_3^1 = f(x_3, w_3^1)$$
$$y_4^1 = f(x_4, w_4^1)$$

$$y^1 = \begin{pmatrix} y_1^1 \\ y_2^1 \\ y_3^1 \\ y_4^1 \end{pmatrix}$$

$$y_1^2 = f(y^1, w_1^2)$$
$$y_2^2 = f(y^1, w_2^2)$$
$$y_3^2 = f(y^1, w_3^2)$$

$$y^2 = \begin{pmatrix} y_1^2 \\ y_2^2 \\ y_3^2 \end{pmatrix}$$

$$y_{Out} = f(y^2, w_1^3)$$

Βασικά Χαρακτηριστικά

- Συσχετίζει (οποιεσδήποτε) εισόδους-εξόδους.
- Υπολογίζει άγνωστες (και εν δυνάμη πολυδάπανες) συναρτήσεις σε ελάχιστο χρόνο (RT, GA)
- Εκπαιδεύεται (συγκρίνει την νέα πληροφορία με την ήδη υπάρχουσα γνώση)
  - Ταξινομεί την νέα πληροφορία
  - Δημιουργεί νέες κατηγορίες όταν πρέπει
  - Έχει Bayesian “attitude” (belief functions)

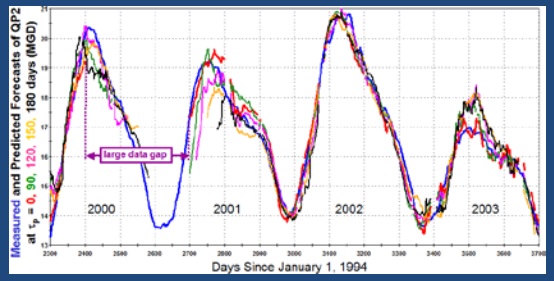
Τι κάνουμε με ΝΔ (και τι όχι)

- Οι δύο βασικές χρήσεις είναι: (α) **κατηγοριοποίηση** και (β) **προσομοίωση άγνωστης συνάρτησης**

- Συγκεκριμένα με ένα ΝΔ μπορούμε να προσομοιώσουμε **οποιαδήποτε συνάρτηση** μέχρι οποιοδήποτε βαθμό ακρίβειας (Universal Approximation Theorem, (Cybenko, 1989))
- Δεν χρησιμοποιούνται για την λύση προβλημάτων που έχουν να κάνουν με συμβολικά μαθηματικά
- Χρησιμοποιούνται μόνο αν υπάρχουν δεδομένα (data driven methods) που να χρησιμοποιούνται για μάθηση.
- Δεν μπορούν να προσομοιώσουν φαινόμενα που δεν «περιέχονται» μέσα στα δεδομένα μάθησης (training data).

Έχουμε αγνωστες συναρτήσεις που θα θέλαμε να υπολογίσουμε (χωρίς να γράψουμε τις εξισώσεις τους;)

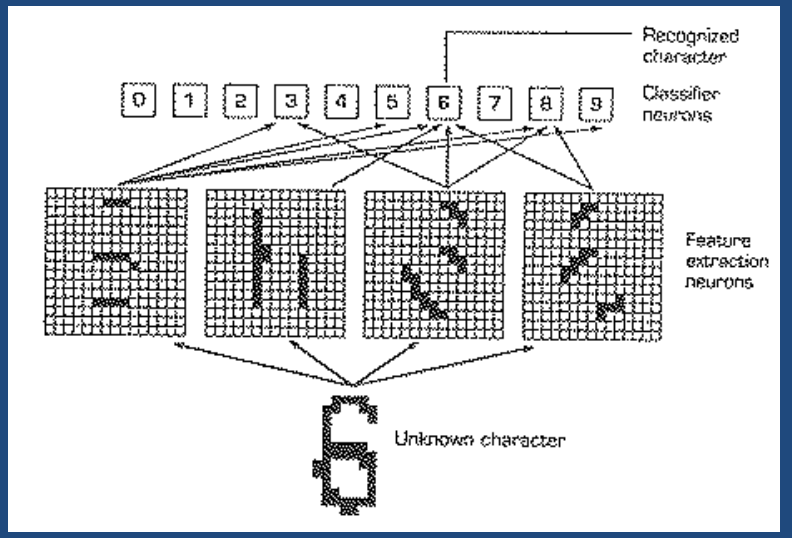
1. **Μουσώνες:** snowfall in Asia, temperature degrees in Northwest Europe, pressure zones occurring in Europe and Asia, surface temperatures occurring in Arabian sea, temperature values occurring a year before in Indian Ocean
2. **Βραχυπρόθεσμη πρόβλεψη ζήτησης:** previous day demand, population size, last year's monthly average, week day
3. **Ανίχνευση προβλήματος στην ποιότητα νερού ύδρευσης:** (πχ ρίψη ουσιών – τρομοκρατικό χτύπημα) (pH, turbidity, temperature, chlorine residual, total organic carbon)



Και βέβαια…

Εφαρμογές:

- **Αναγνώριση γραφής**: OCR (Optical Character Recognition)
- Που βρίσκει ο αλγόριθμος τις τιμές εκπαίδευσης;
- Το ίδιο και με τη φωνή (Siri…)



Τύποι ΝΔ

- Δίκτυα πρόσθιας τροφοδότησης (feed-forward ANN)
  - Αντιληπτήρας Πολλαπλών Επιπέδων (multilayer perceptron)
- Δίκτυα ανάστροφης τροφοδότησης (feed-back/recurrent ANN)

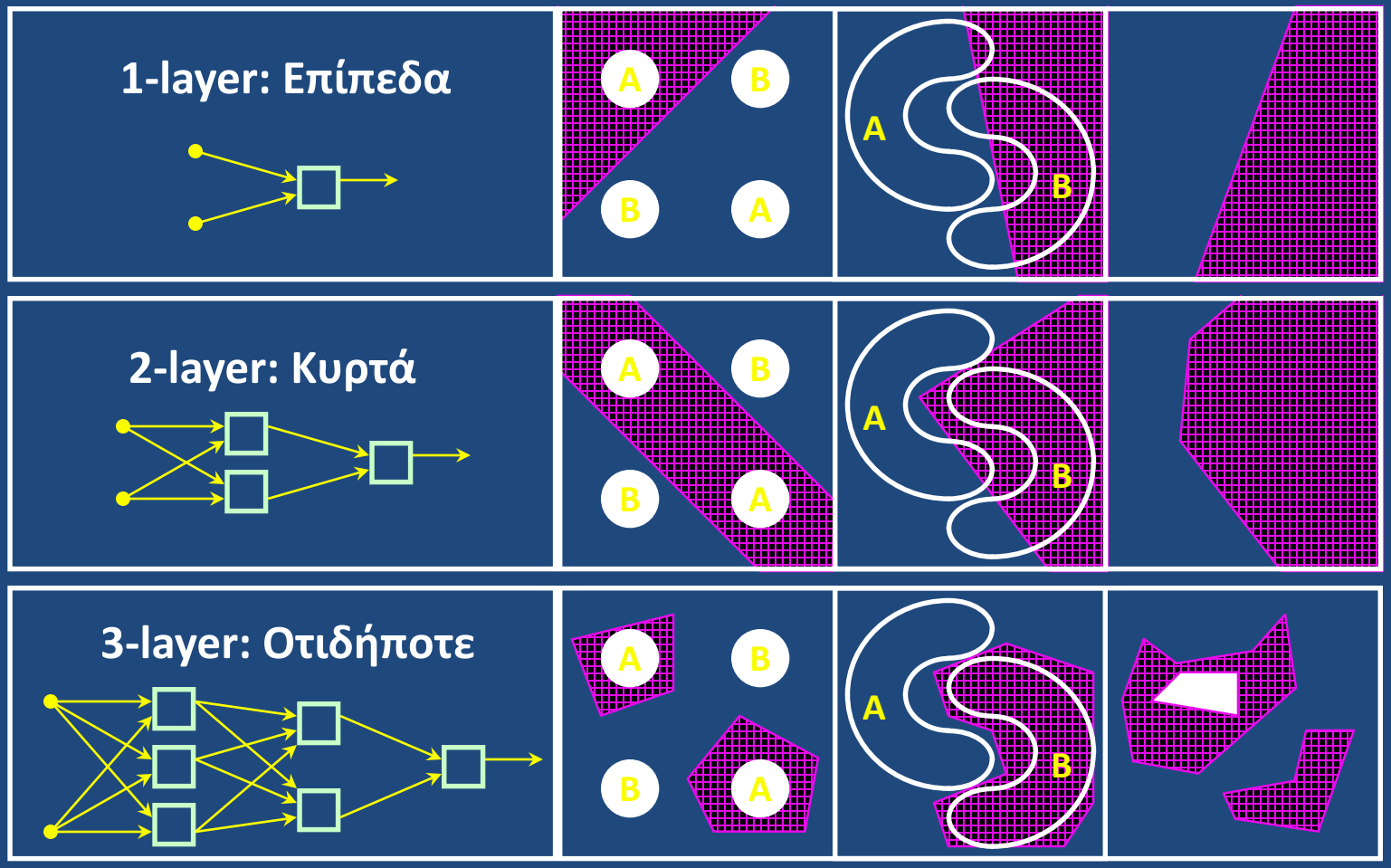
Αντιληπτήρες πολλαπλών επιπέδων (MLP)

- Ένα από τα πιο συνηθισμένα Νευρωνικά Δίκτυα
- Δίκτυο πρόσθιας τροφοδότησης (feed forward)
- Γενίκευση του (μονοστρωματικού) perceptron
- Εποπτευόμενη (supervised) μάθηση
- Αρχιτεκτονική:
  - 1 επίπεδο εισόδου
  - 1 ή περισσότερα «κρυμμένα» επίπεδα
  - 1 επίπεδο εξόδου
- Εκπαίδευση με τον αλγόριθμο ανάστροφης διάδοσης σφάλματος (Back Propagation)

MLP

- Ένα multi level perceptron έχει τα εξής χαρακτηριστικά:
  - Κάθε κρυμμένος νευρώνας περιέχει μια ΜΗ-ΓΡΑΜΜΙΚΗ συνάρτηση ενεργοποίησης (activation function)
  - Η συνάρτηση αυτή είναι ΔΙΑΦΟΡΙΣΙΜΗ
  - Τα κρυμμένα επίπεδα προσδίδουν στο δίκτυο την δυνατότητα να «μάθει» πολύπλοκα πρότυπα
  - Δεν επιτρέπονται συνδέσεις μεταξύ επιπέδων που δεν ανήκουν σε διαδοχικά επίπεδα
  - Δεν επιτρέπονται συνδέσεις μεταξύ νευρώνων του ίδιου επιπέδου

Γιατί πολλαπλά επίπεδα; Όρια αποφάσεων MLP



Μάθηση στα ΝΔ: *Κανόνας Δέλτα*

Σφάλμα:

$$E = \frac{1}{N}\sum_{t=1}^{N}(F(x_t;W) - y_t)^2$$

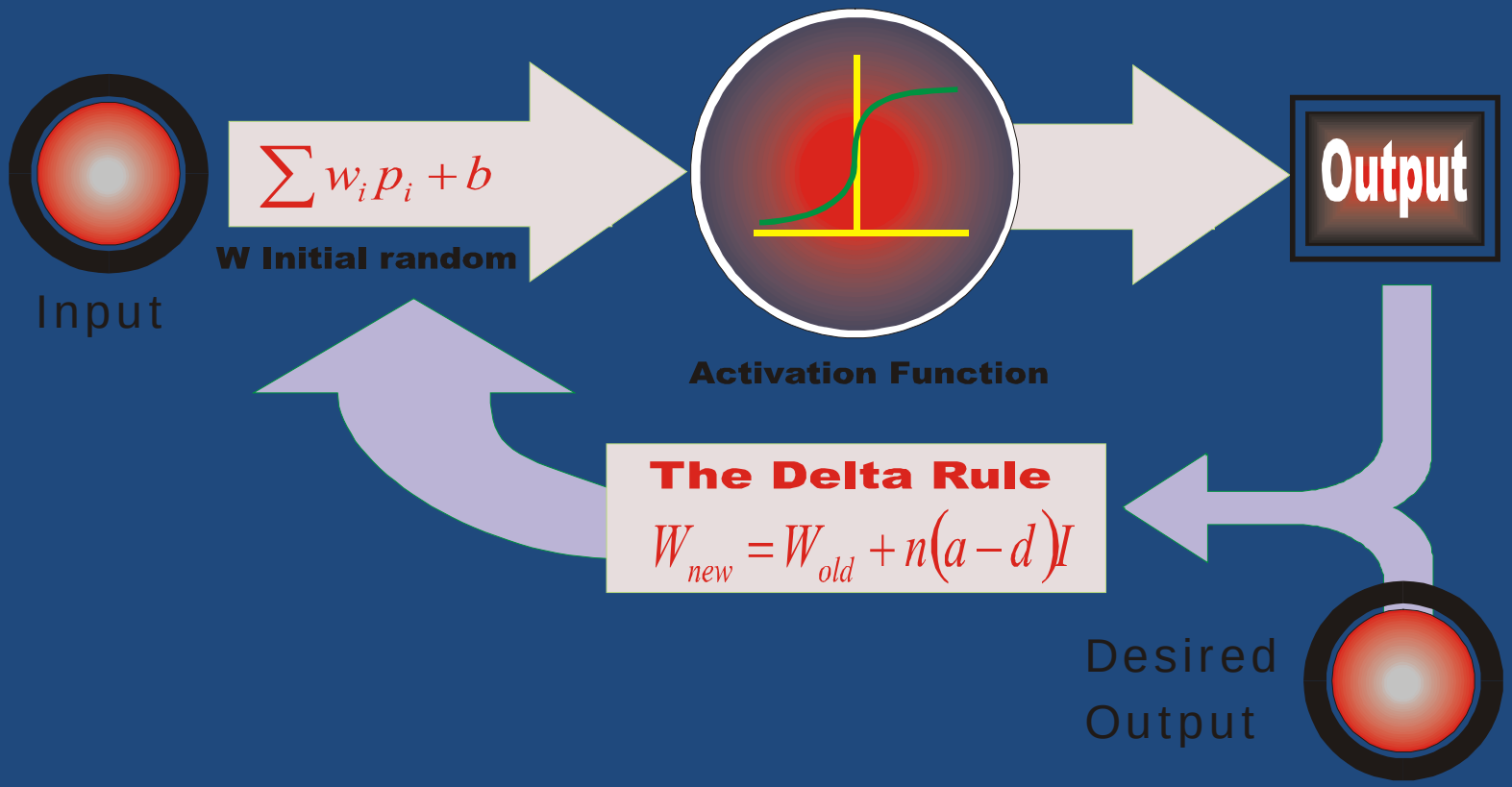
Κανόνας διόρθωσης βαρών:

$$\Delta w_i^j = -c \cdot \frac{\partial E}{\partial w_i^j}(W)$$

$$w_i^{j,new} = w_i^j + \Delta w_i^j$$

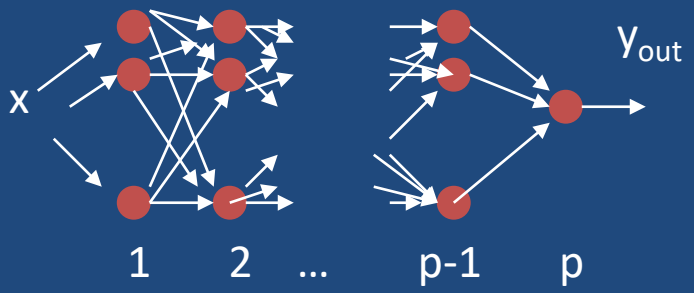
c είναι κάποια (σταθερή συνήθως) παράμετρος «μάθησης»

Διαδικασία


$\sum w_i p_i + b$
W Initial random
Input
Activation Function
Output
The Delta Rule
$W_{new} = W_{old} + n(a-d)I$
Desired
Output

Ο υπολογισμός των βαρών όμως γίνεται δύσκολος...

MLP με p επίπεδα (τα ενδιάμεσα λέγονται κρυφά)



$$y_k^1 = \frac{1}{1+e^{-w^{1kT}x - a_k^1}}, k = 1,...,M_1$$

$$y^1 = (y_1^1,...,y_{M_1}^1)^T$$

$$y_k^2 = \frac{1}{1+e^{-w^{2kT}y^1 - a_k^2}}, k = 1,...,M_2$$

$$y^2 = (y_1^2,...,y_{M_2}^2)^T$$

...

$$y_{out} = F(x;W) = w^{pT} y^{p-1}$$

Δεδομένα: $(x^1, y_1),(x^2, y_2),...,(x^N, y_N)$

Σφάλμα: $E(t) = (y(t)_{out} - y_t)^2 = (F(x^t;W) - y_t)^2$

Ο κανόνας είναι κάπως πιο δύσκολος...

Υπολογισμός με ολική βελτιστοποίηση

- Αρκεί ακόμα και ένα επίπεδο για να προσομοιώσουμε μη-γραμμικές συναρτήσεις.
- Στη περίπτωση αυτή το πρόβλημα υπολογισμού βαρών μπορεί να λυθεί με βελτιστοποίηση
- Πχ. Για τη συνάρτηση:

$$y_{out} = F(x;W) = \sum_{k=1}^{M} w_k^2 \cdot \frac{1}{1+e^{-w^{1,kT}x-a_k}}$$

Βελτιστοποίηση (βαθμονόμηση…)

Βάρη για το επίπεδο εξόδου (ένας νευρώνας):

$$w_i^2(t+1) = w_i^2(t) - c \cdot \frac{\partial E(t)}{\partial w_i^2}$$

$$\frac{\partial E(t)}{\partial w_i^2} = 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{1}{1 + e^{-w^{1,iT} x^t - a_i}}$$

Βάρη για το κρυφό επίπεδο (πολλοί νευρώνες)

$$w_j^{1,i}(t+1) = w_j^{1,i}(t) - c \cdot \frac{\partial E(t)}{\partial w_j^{1,i}}$$

$$\frac{\partial E(t)}{\partial w_j^{1,i}} = 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{\partial}{\partial w_j^{1,i}} \left(\frac{1}{1 + e^{-w^{1,iT} x^t - a_i}} \right)$$

$$\frac{\partial}{\partial w_j^{1,i}} \left(\frac{1}{1 + e^{-w^{1,iT} x^t - a_i}} \right) = \frac{e^{-w^{1,iT} x^t - a_i}}{\left(1 + e^{-w^{1,iT} x^t - a_i}\right)^2} \cdot \frac{\partial}{\partial w_j^{1,i}} \left(- w^{1,iT} x^t - a_i \right)$$

$$\frac{\partial}{\partial w_j^{1,i}} \left(- w^{1,iT} x^t - a_i \right) = -x_j^t$$

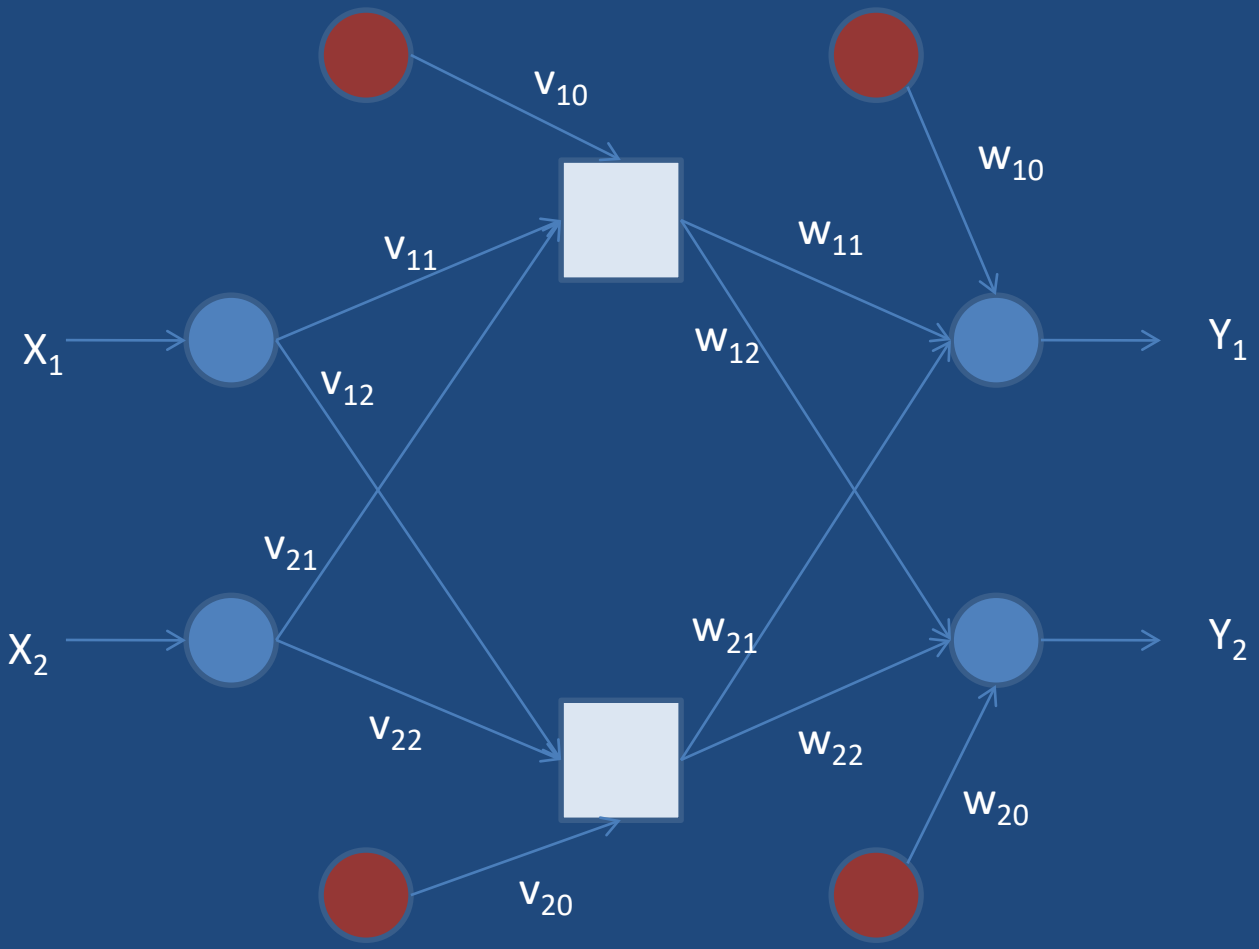
$$w_j^{1,i}(t+1) = w_j^{1,i}(t) - c \cdot 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{e^{-w^{1,iT} x^t - a_i}}{\left(1 + e^{-w^{1,iT} x^t - a_i}\right)^2} \cdot (-x_j^t)$$

Ή βέβαια με γενετικούς!

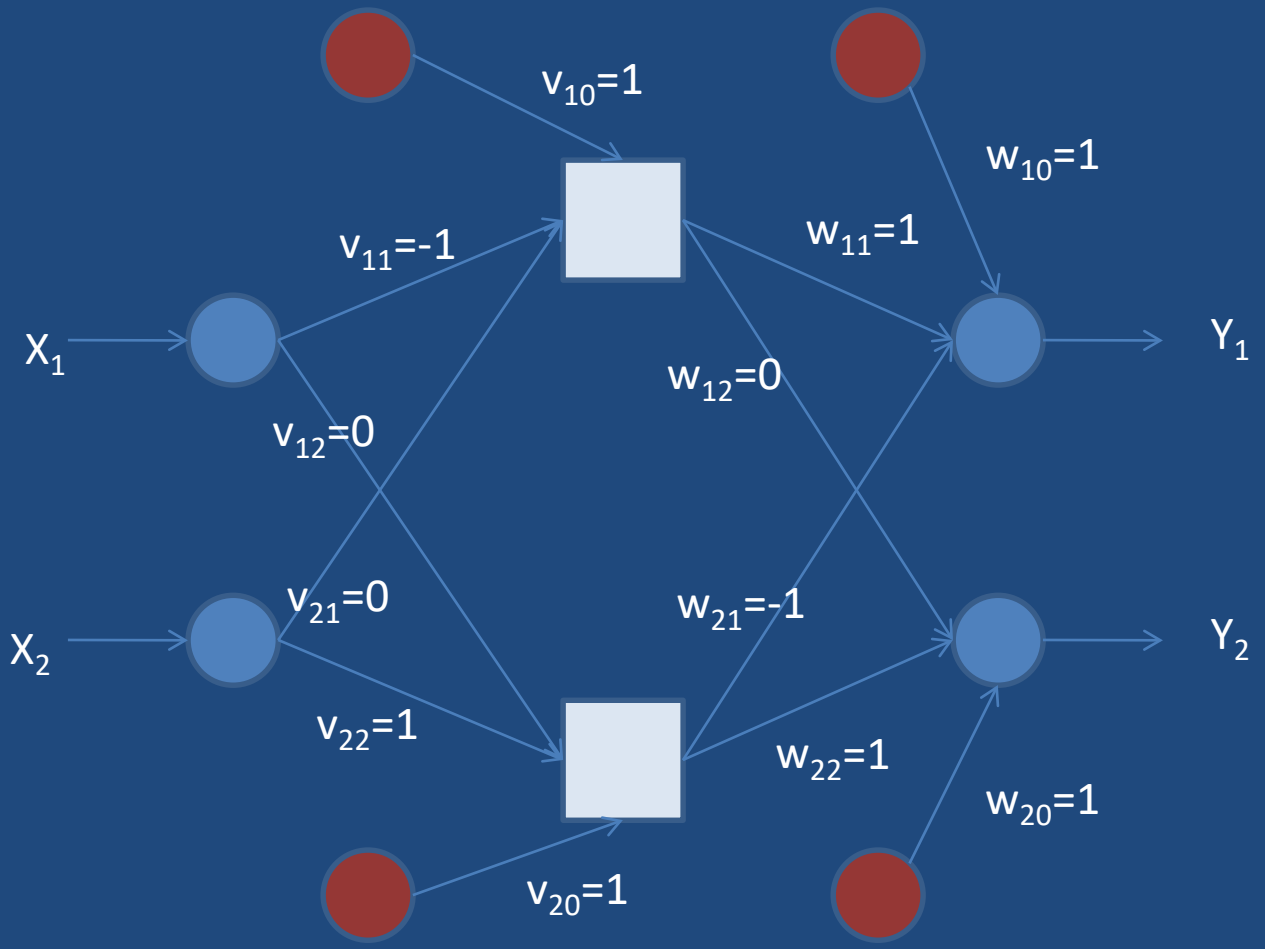
Εναλλακτικά μάθηση «προς τα πίσω» (backpropagation)

- Αυτή είναι η πιο γενική περίπτωση (και πιο συνηθισμένη για τα MLP) και βασίζεται σε επίλυση σε βήματα/επαναλήψεις.
- Υπολογίζουμε πρώτα τις αλλαγές των βαρών στον νευρώνα **εξόδου**
- Υπολογίζουμε τις υπόλοιπες αλλαγές **προς τα πίσω** ξεκινώντας από το επίπεδο p-1, και ‘περνώντας’ προς τα πίσω το τοπικό σφάλμα.

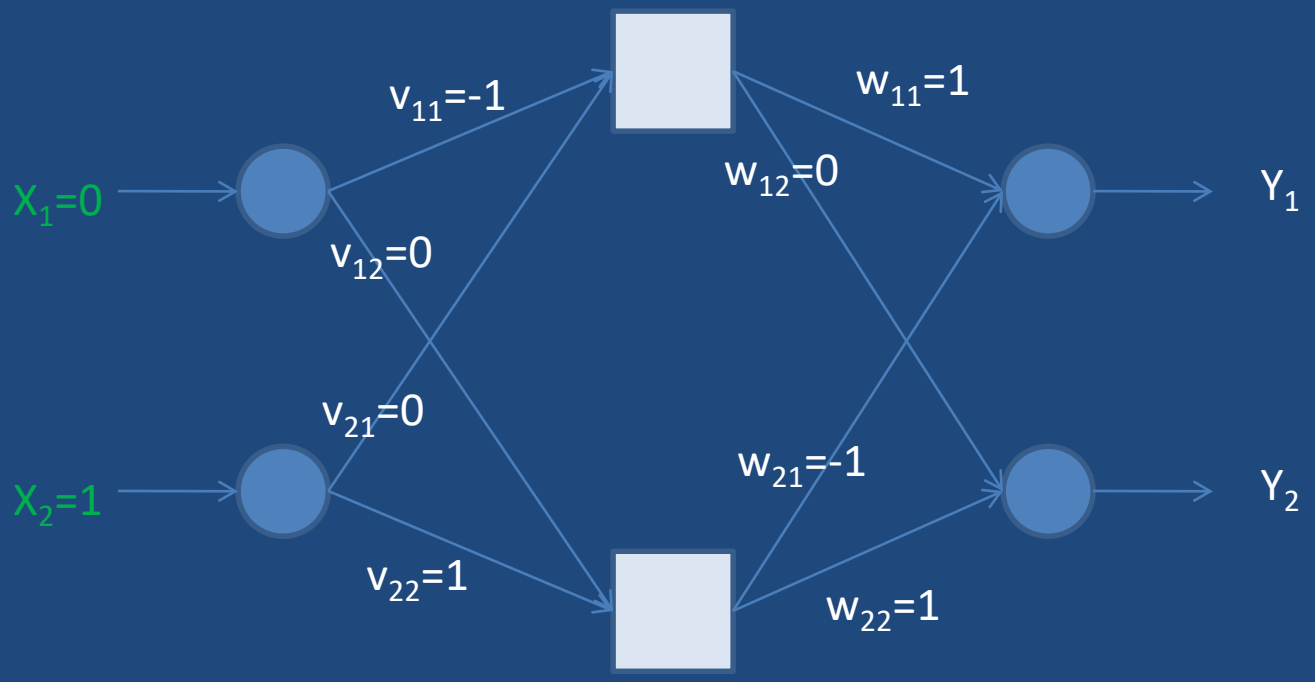
Παράδειγμα



Παράδειγμα

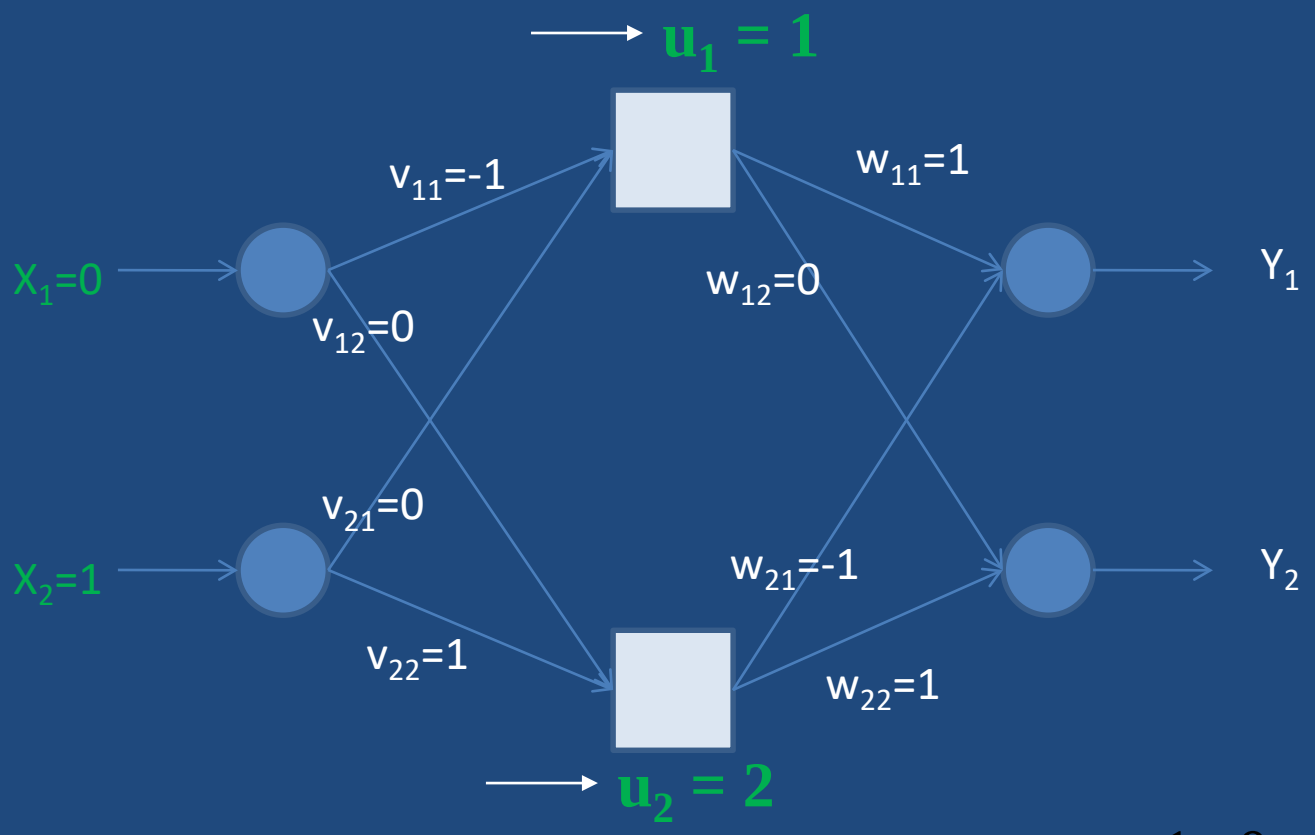

$v_{10}=1$
$w_{10}=1$
$v_{11}=-1$
$w_{11}=1$
X_1
Y_1
$w_{12}=0$
$v_{12}=0$
$v_{21}=0$
$w_{21}=-1$
X_2
Y_2
$v_{22}=1$
$w_{22}=1$
$w_{20}=1$
$v_{20}=1$

Παράδειγμα



Είσοδος: $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ Επιθυμητή Έξοδος: $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Παράδειγμα

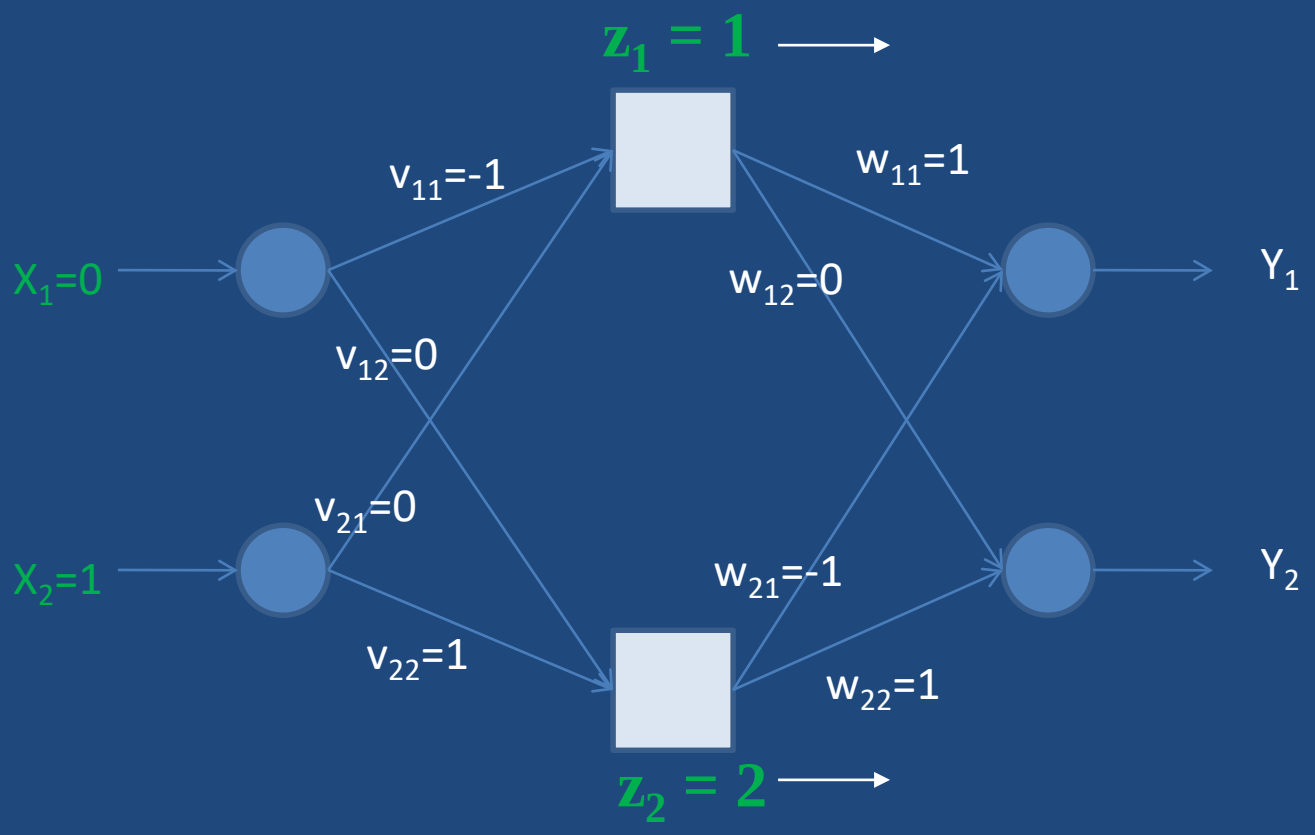


- Πέρασμα κατά την **ΕΥΘΕΙΑ** φορα
- Υπολογίζουμε τις ενεργοποιήσεις 1ου επιπέδου

$$u_1 = -1 \times 0 + 0 \times 1 + 1 = 1$$

$$u_2 = 0 \times 0 + 1 \times 1 + 1 = 2$$

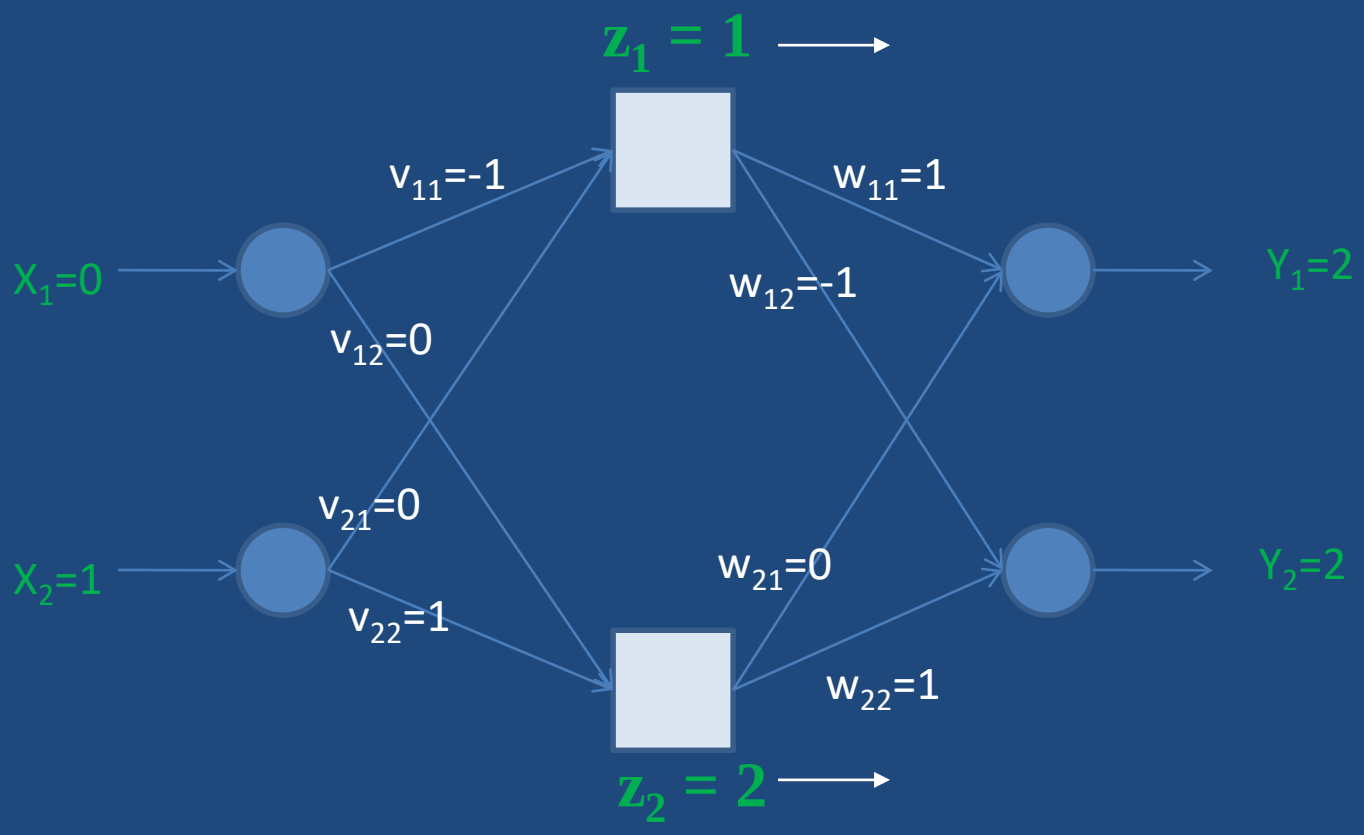
Παράδειγμα



Υπολογίζουμε τις εξόδους του πρώτου επιπέδου, περνώντας τις ενεργοποιήσεις που υπολογίσαμε προηγουμένως στη **συνάρτηση ενεργοποίησης**:

Εδώ (για λόγους απλότητας):
z1 = u1
z2 = u2

Παράδειγμα

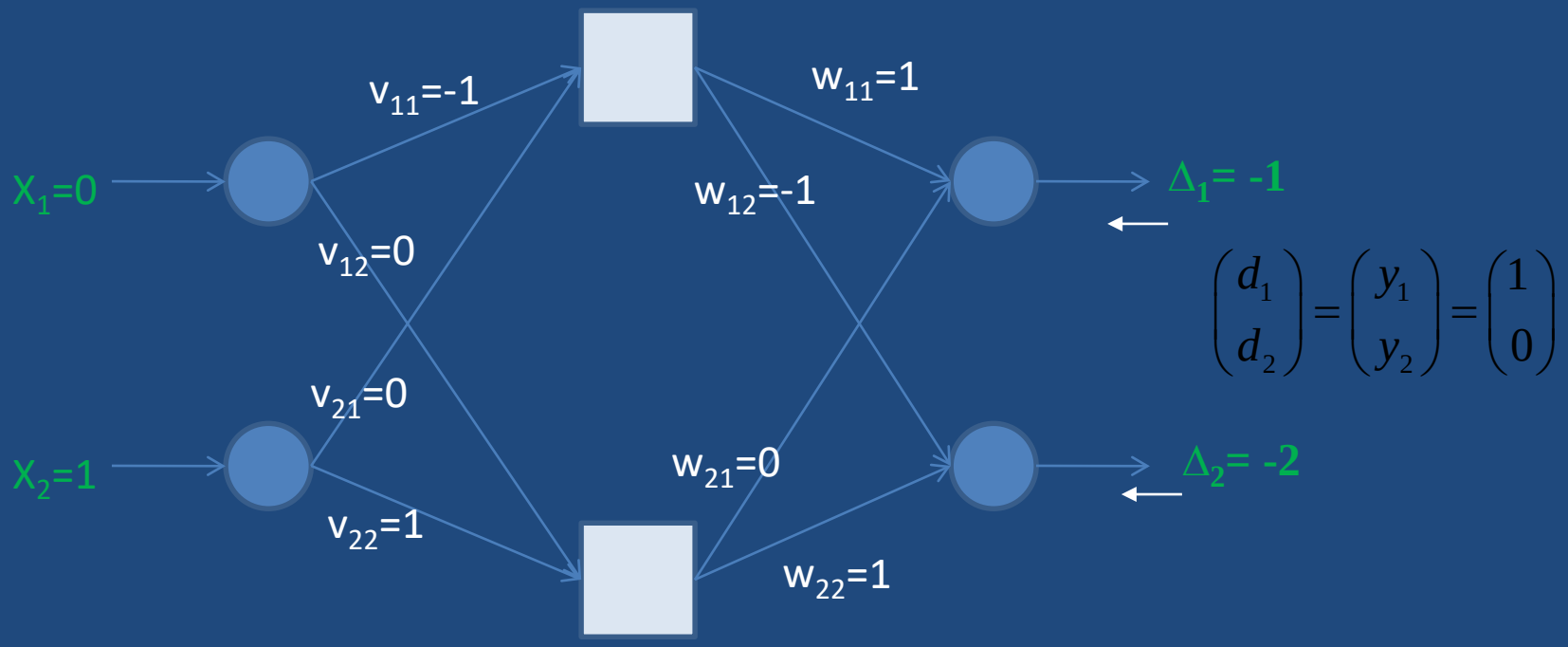


- Υπολογίζουμε τις εξόδους του 2ου layer:

$$y_1 = a_1 = 1\times 1 + 0\times 2 + 1 = 2$$

$$y_2 = a_2 = -1\times 1 + 1\times 2 + 1 = 2$$

Παράδειγμα



$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Διόρθωση Βαρών κατά cΔ = c(σωστό – απάντηση).

Δ1 = 1-2 = -1, Δ2 = 0-2 = -2

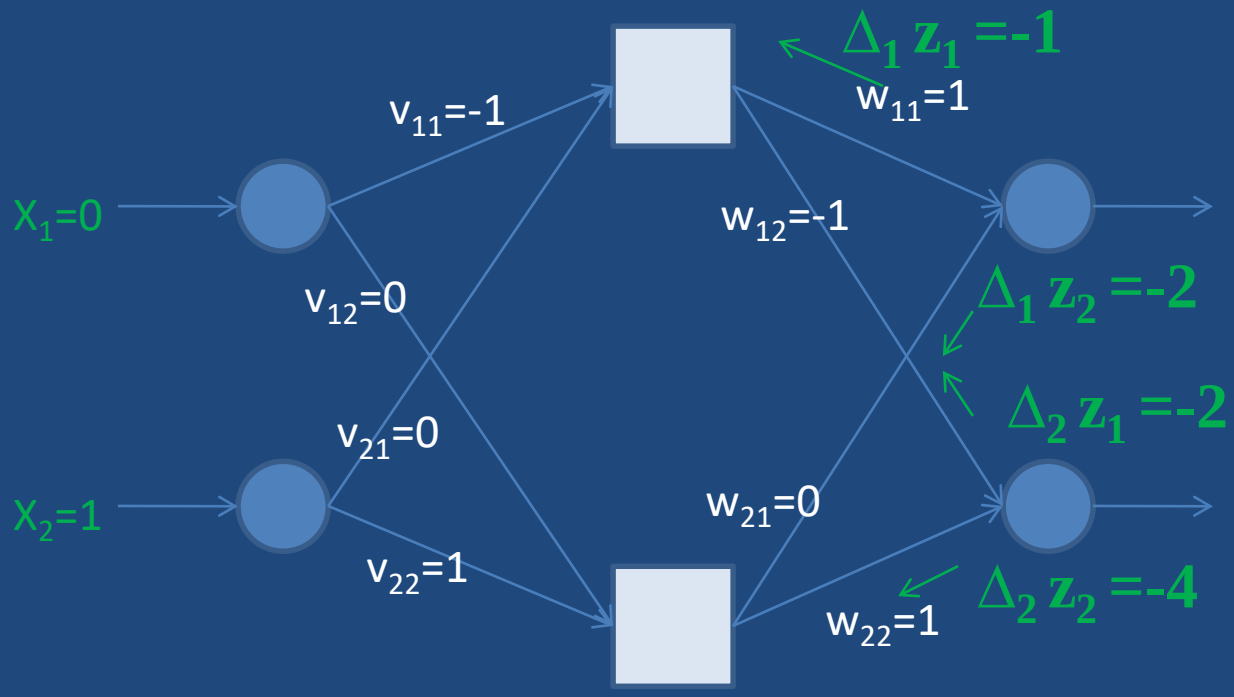
$$w_{ij}(t+1) - w_{ij}(t) = \eta \Delta_i(t) z_j(t)$$

Παράδειγμα

για c=0.1

z1=1
z2=2

Δ1=-1
Δ2=-2

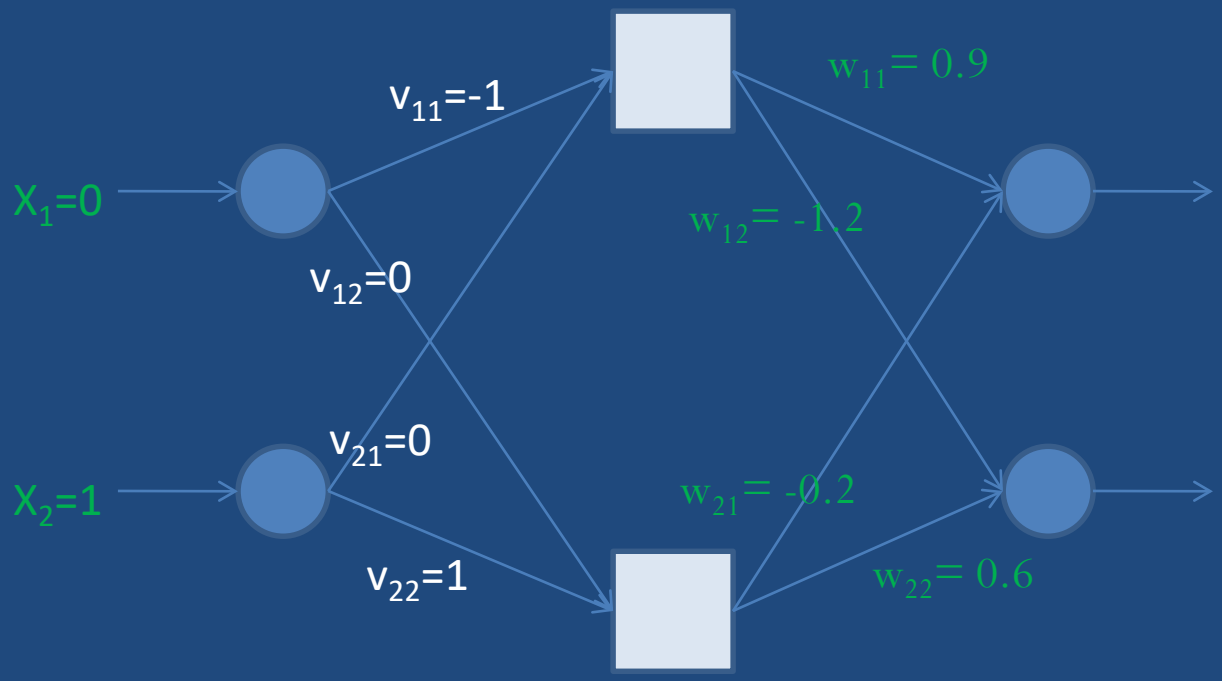


- Υπολογίζουμε τις αλλαγές στα βάρη για το 2ο επίπεδο, με βάση τον κανόνα:

$$w_{ij}(t+1) - w_{ij}(t) = \eta \Delta_i(t) z_j(t)$$

Παράδειγμα

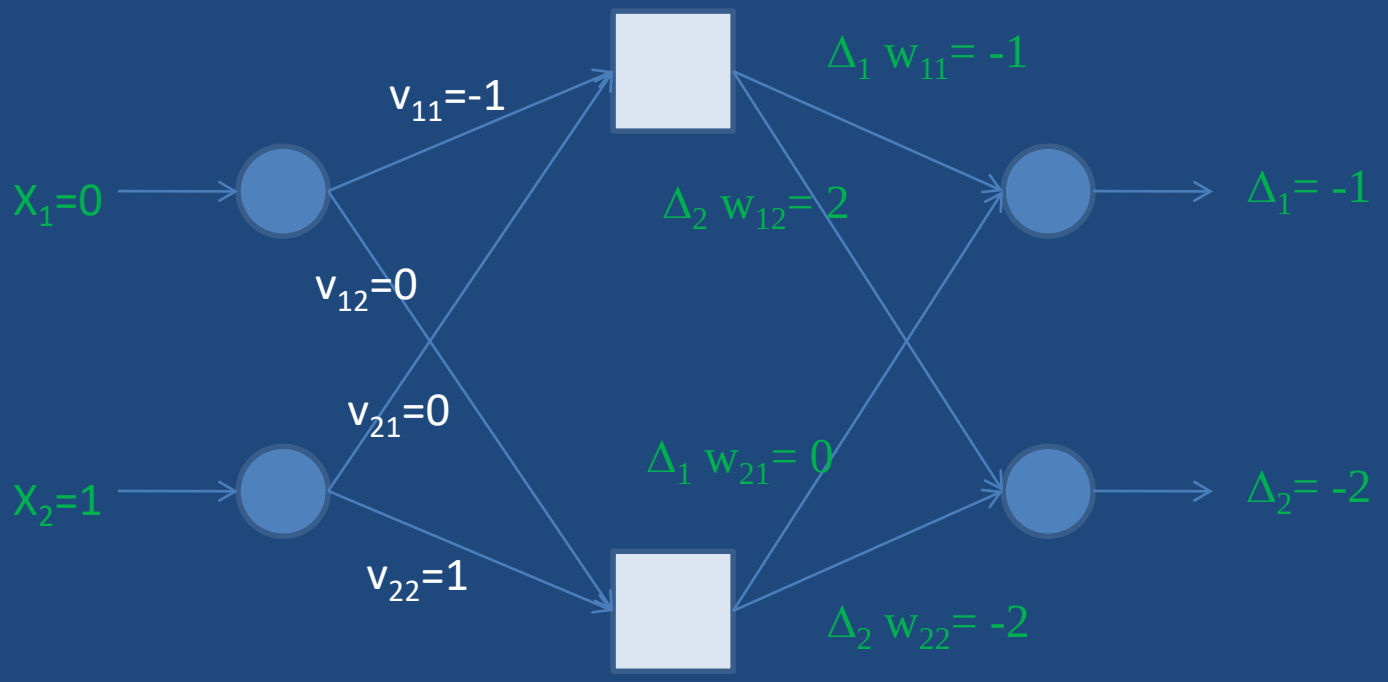
για c=0.1



Οι αλλαγές στα βάρη είναι:

w11 (t+1) = w11 (t) + 0.1 * (-1) = 0.9

Παράδειγμα

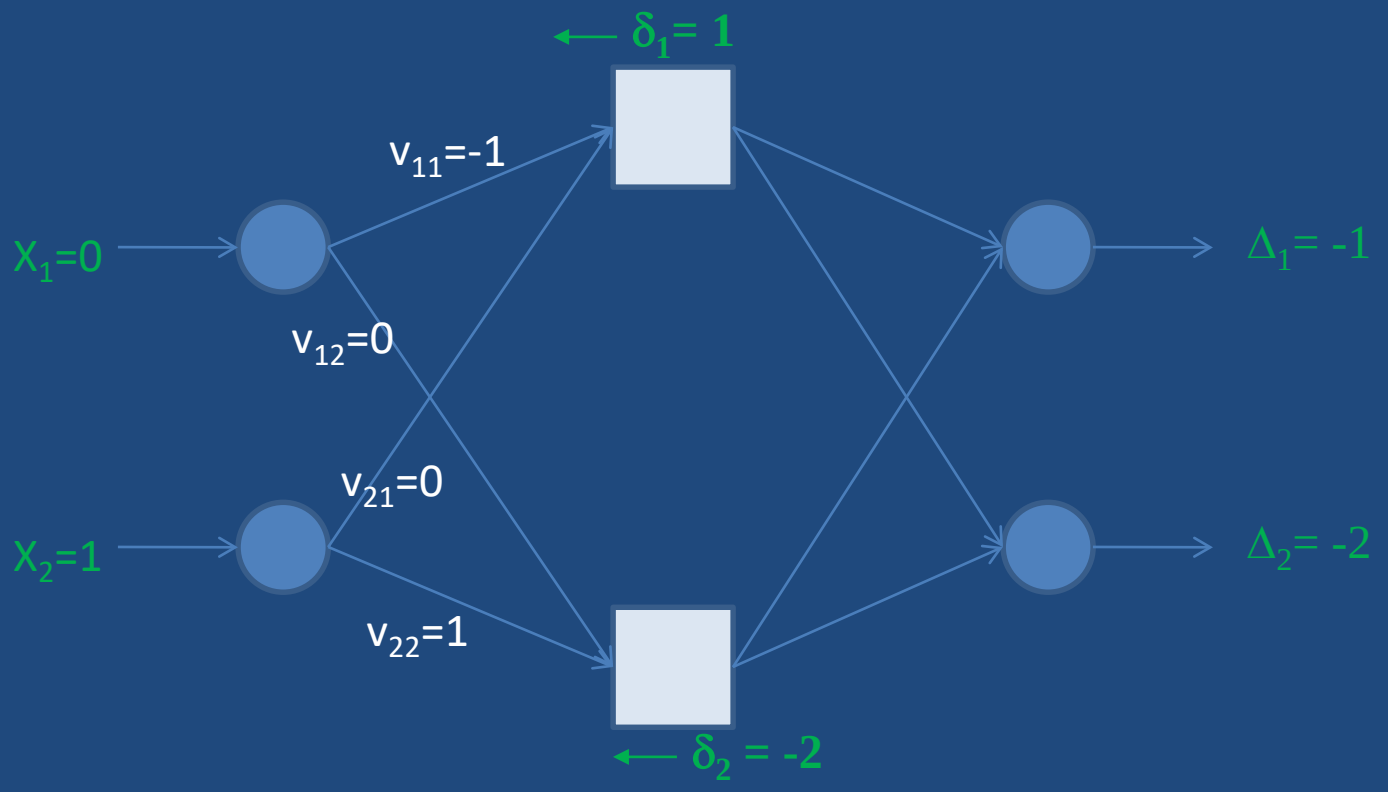


Υπολογίζουμε τα δ : $\delta_i(t) = g'(u_i(t)) \sum_k \Delta_k(t) w_{ki}$

$$\delta_1 = -1 + 2 = 1$$

$$\delta_2 = 0 - 2 = -2$$

Παράδειγμα

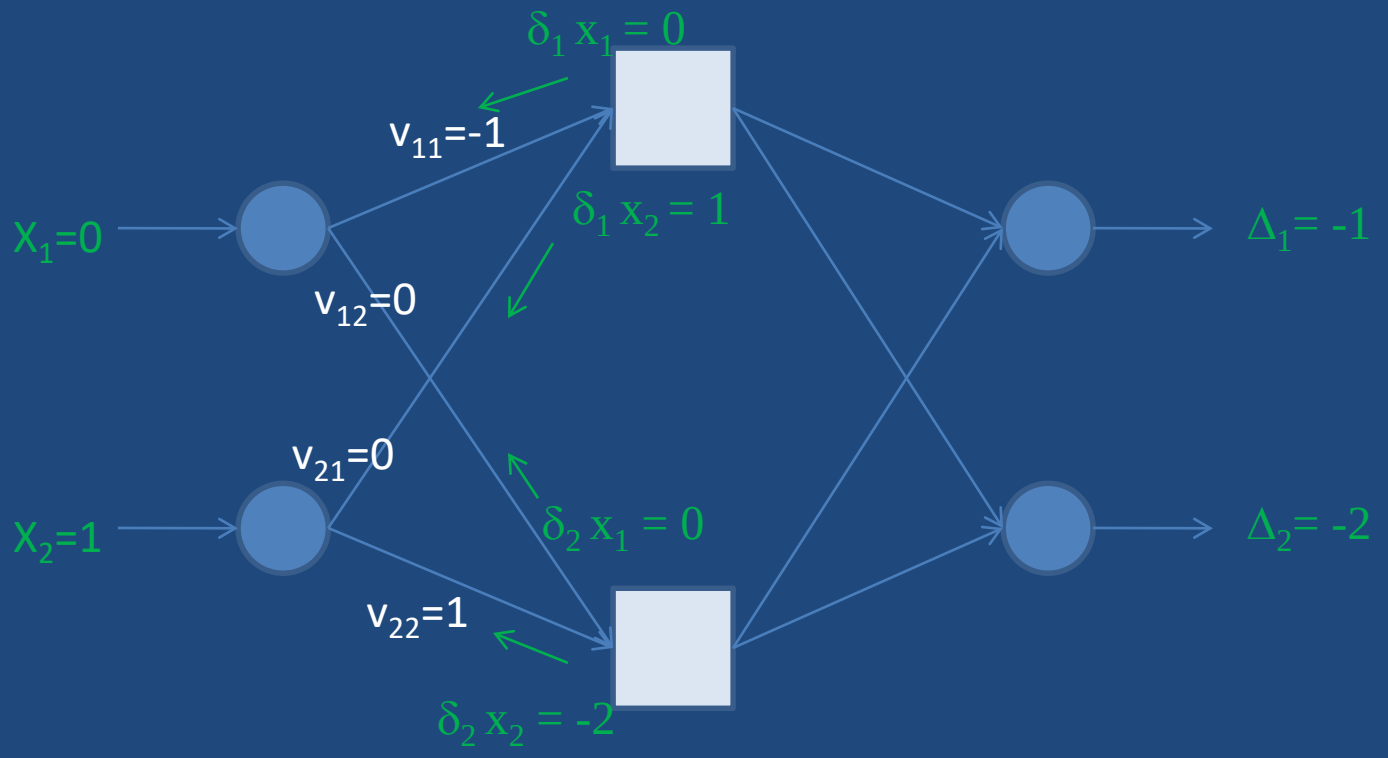


Τα Δ διαδίδονται προς την αντίθετη κατεύθυνση:

$$\delta_1 = -1 + 2 = 1$$

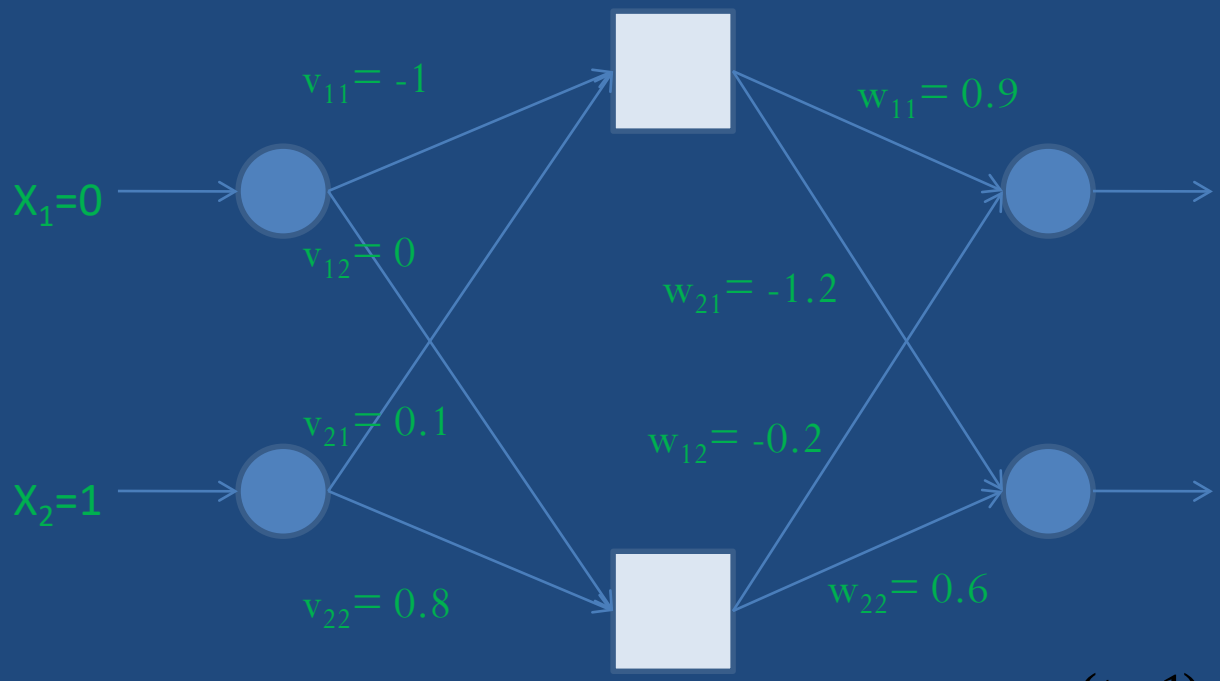
$$\delta_2 = 0 - 2 = -2$$

Παράδειγμα



Και πολλαπλασιάζοντας με τις εισόδους: $v_{ij}(t+1) - v_{ij}(t) = \eta \delta_i(t) x_j(t)$

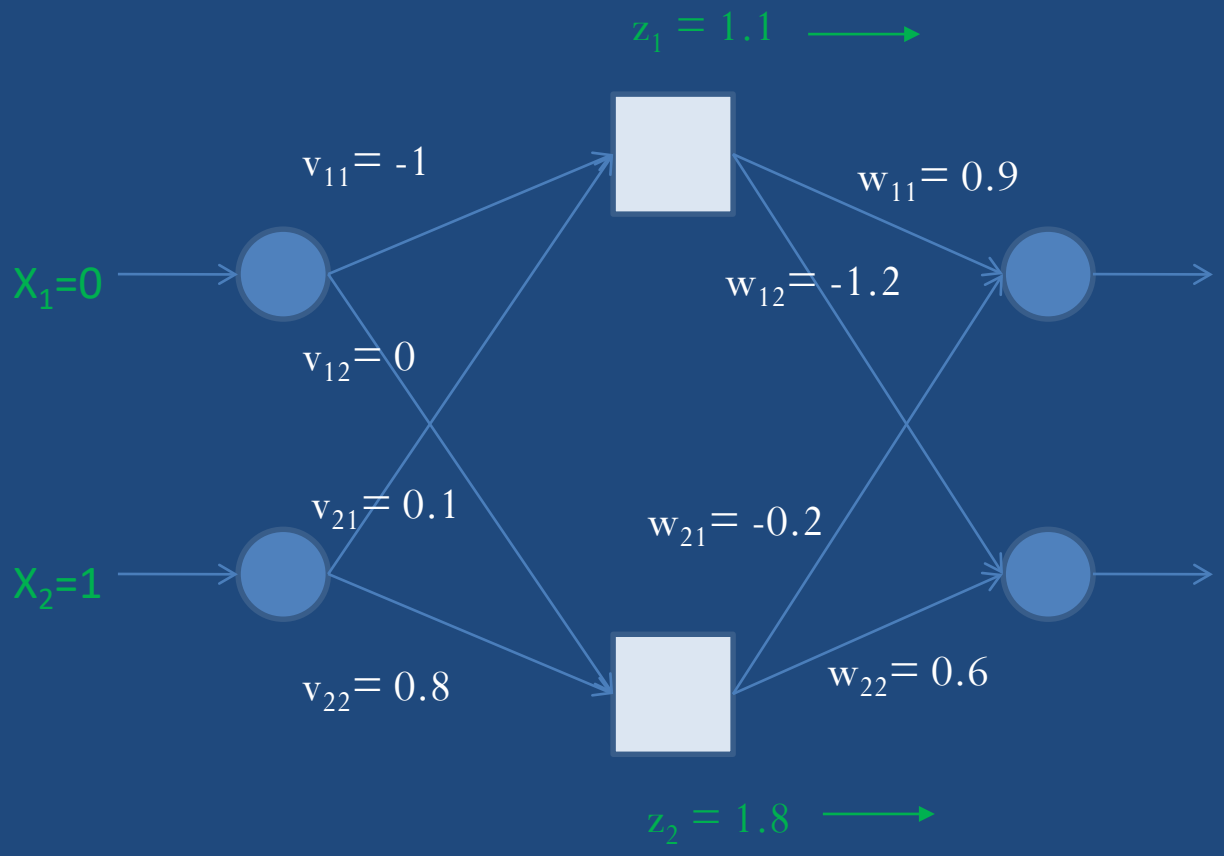
Παράδειγμα



$$v_{ij}(t+1) - v_{ij}(t) = \eta \delta_i(t) x_j(t)$$

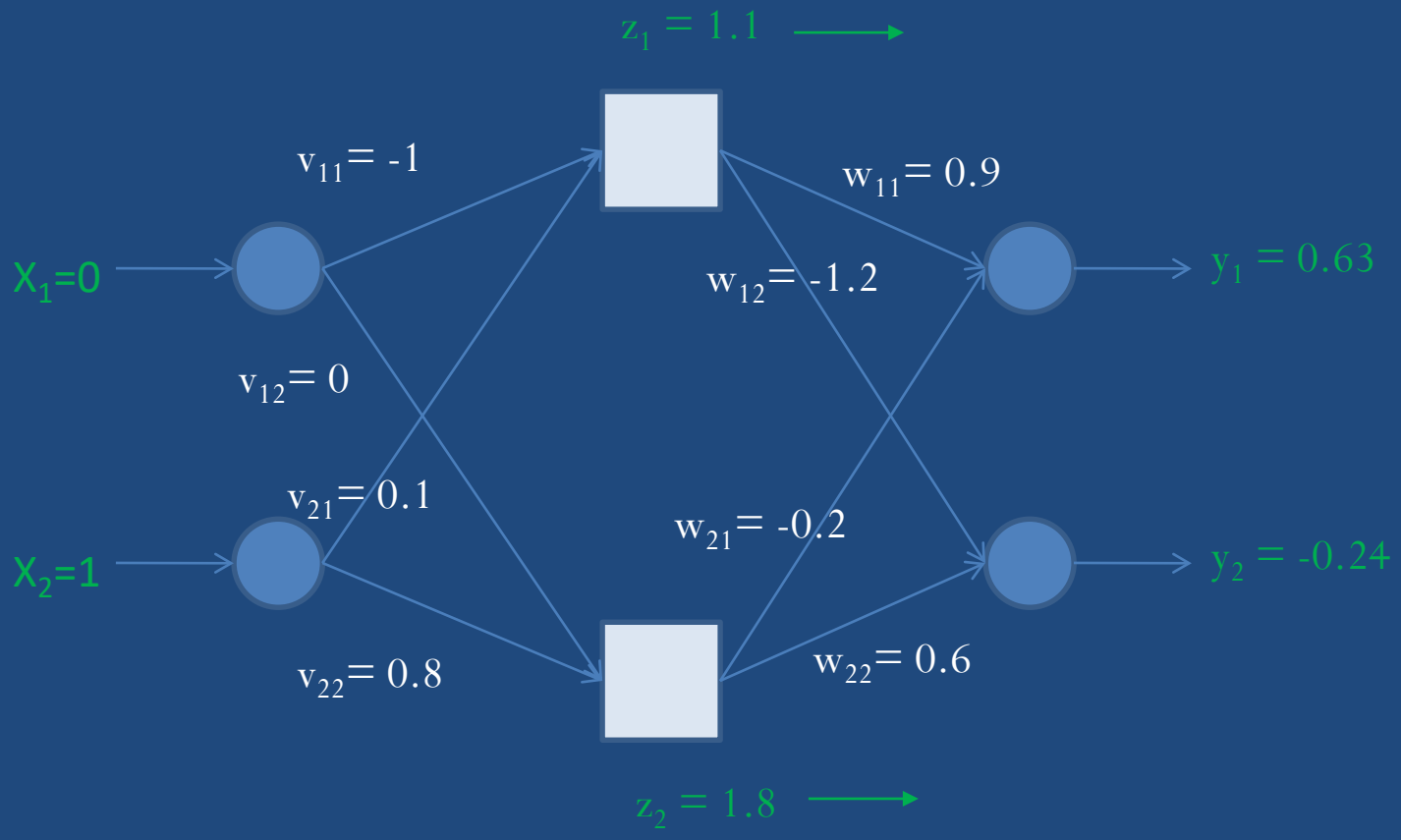
Τέλος, αλλάζουμε τα βάρη.

Παράδειγμα



Επαναλαμβάνουμε την διαδικασία χρησιμοποιώντας το ίδιο πρότυπο

Παράδειγμα



Τώρα, οι έξοδοι βρίσκονται πιο κοντά στις επιθυμητές: $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Γενική Διαδικασία

- Επιλογή της αρχιτεκτονικής του δικτύου (κρυμμένα επίπεδα, αριθμός νευρώνων, συνάρτηση μάθησης, συνάρτηση ενεργοποίησης):
- Δυστυχώς δεν υπάρχει ‘σωστή’ απάντηση σ αυτό (trial and error + Ocham’s razor…)
- Αρχικοποίηση του δικτύου (βάρη, παράμετροι)
- Έλεγχος και προεπεξεργασία δεδομένων
- Παρουσίαση του συνόλου δεδομένων
- Εκπαίδευση του δικτύου
  - Επαναλήψεις εώς κάποιο καθορισμένο σημείο με βάση προκαθορισμένα κριτήρια
  - Παρουσίαση του συνόλου ελέγχου
  - Αξιολόγηση αποτελεσμάτων και αν χρειαστεί, αλλαγή παραμέτρων και/ή αρχιτεκτονικής και εκ νέου εκπαίδευση του δικτύου
- Τι **πρόβλημα** προβλέπετε να έχουμε με αυτή την διαδικασία μάθησης;
- Τι **πρόβλημα** θα έχουμε πάντα;

Matlab

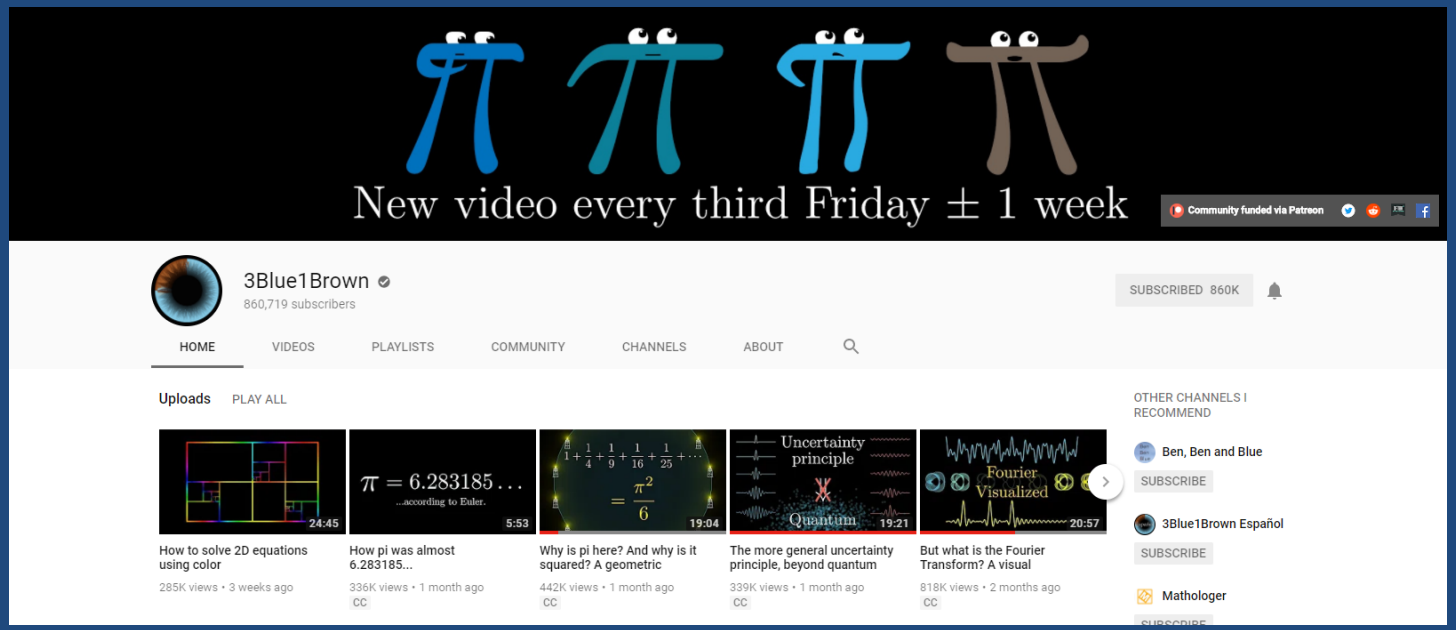
1. nnstart
2. nftool
    1. (load test data – save net – see weights (net.lw))
    2. Create training data (inputs, outputs), train an ANN
    3. To run outputs=net(inputs)
3. ntstool
    1. Load test data (pollution mortality), train
    2. Evaluate for pollutionInput
    3. Command line: *outputs = net2(pollutionInputs)*
4. nprtool:
    1. wines (check **confusion matrix**).
    2. Φτιάξτε ένα νέο vineyard and add to set
    3. Θυμηθείτε: [a;b] [a b]

Αναφορές

Η παρουσίαση χρησιμοποίησε υλικό, σχήματα και παραδείγματα από αντίστοιχες παρουσιάσεις των συναδέλφων:

- Α. Αργυρίου. Εργαστήριο Φυσικής της Ατμόσφαιρας, Παν/μιο Πάτρας
- P. Andras, University of Newcastle
- Ν. Σίμου και Β. Σπύρου, Εργαστήριο Εικόνας, Βίντεο και Πολυμέσων, ΕΜΠ
- Γ. Παπαδουράκη, ΤΕΙ Κρήτης

Αξίζει επίσης να δείτε αυτό:



https://www.youtube.com/watch?v=aircAruvnKk&t=1021s