ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

Τεχνητή Νοημοσύνη Ι

Τεχνητά Νευρωνικά Δίκτυα

Διδάσκων: Τσούρος Δήμος

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

Τεχνητά Νευρωνικά Δίκτυα

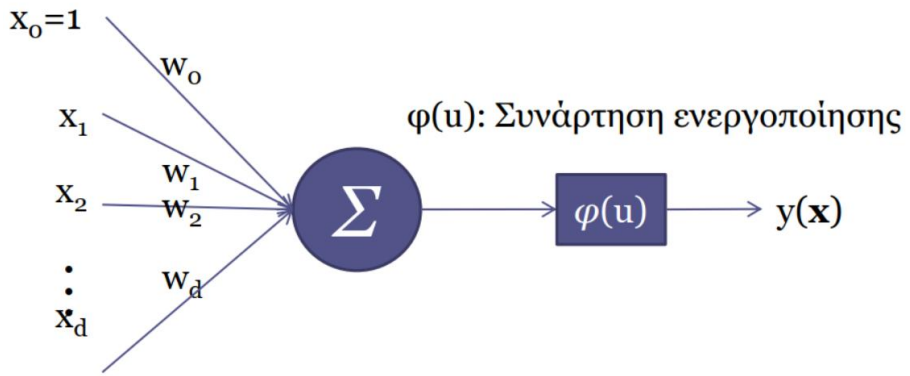
Τεχνητά Νευρωνικά Δίκτυα

Στόχος μαθήματος

- να κατανοήσουμε γιατί ο perceptron δεν επαρκεί,
- να δούμε πως μπορούμε να χρησιμοποιήσουμε παραπάνω από έναν νευρώνες, φτιάχνοντας δίκτυα νευρώνων (*νευρωνικά δίκτυα*),
- να δούμε της μαθηματικές ιδέες πίσω από τα κρυφά επίπεδα νευρώνων και την μη γραμμικότητα,
- να μάθουμε πώς εκπαιδεύονται τα νευρωνικά δίκτυα (Loss + Backpropagation).

Νευρώνας perceptron

Ο perceptron είναι η πιο βασική μορφή νευρώνα:



x: διάνυσμα εισόδου
w: διάνυσμα βαρών

- Χρησιμοποιεί την βηματική συνάρτηση ως συνάρτηση ενεργοποίησης:

$$y = step(\sum_{i=1}^{n} w_i x_i + b),$$

με

$$step(f(x)) = \begin{cases} 1, f(x) > 0 \\ 0, f(x) \leq 0 \end{cases}$$

  - Υπενθύμιση: Ουσιαστικά χρησιμοποιούμε μία μοναδιαία είσοδο ($x_0 = 1$), τις οποίας το βάρος (w_0) ουσιαστικά αναπαριστά το *b*

Νευρώνας perceptron

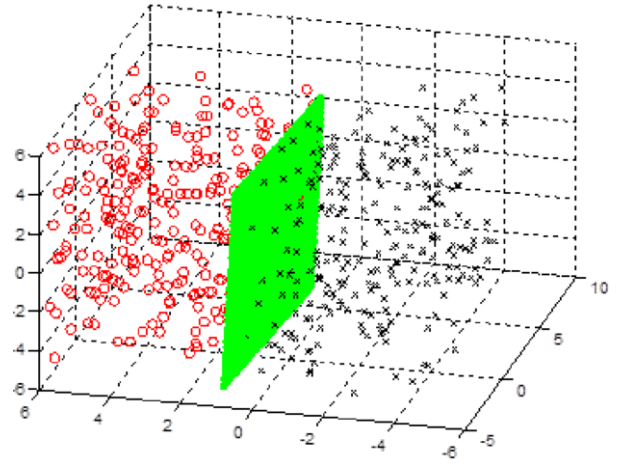
Ο perceptron ουσιαστικά μαθαίνει μια συνάρτηση

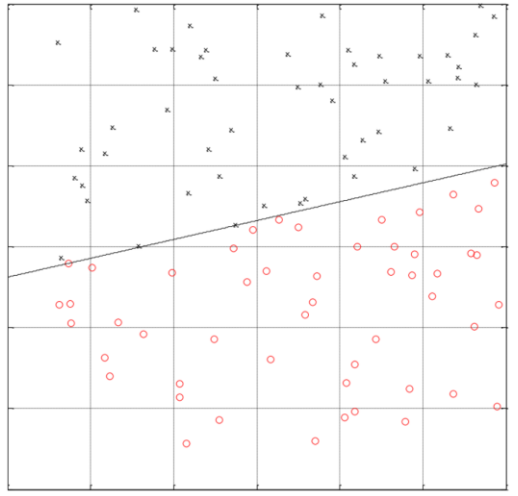
$$\mathrm{h1} = \sum_{i=1}^{n} w_i\, x_i + b$$

Και δημιουργεί ένα **υπερεπίπεδο διαχωρισμού** στο χώρο εισόδων με βάση τη συνάρτηση:

$$\sum_{i=1}^{n} w_i\, x_i + b = 0$$

- Όλα τα σημεία που βρίσκονται πάνω ή κάτω από αυτήν την ευθεία/υπερεπίπεδο κατηγοριοποιούνται αντίστοιχα στην κατηγορία 0 ή 1.

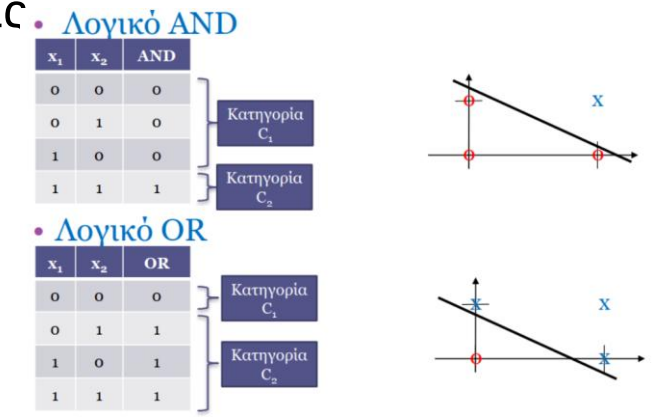




Νευρώνας perceptron

- Ο perceptron μπορεί να αναπαραστήσει πολλές συναρτήσεις που βρίσκουμε σε πολλά προβλήματα.

- Π.χ. τις συναρτήσεις AND και OR
  - Αν βρέχει ή (OR) είμαι τραυματίας, δεν θα πάω να παίξω ποδόσφαιρο
  - Αν έχει σταθερό εισόδημα και (AND) δεν χρωστάει, θα του δώσουμε δάνειο.



Δεν μπορεί όμως να αναπαραστήσει όλες τις πιθανές συναρτήσεις:

- Μόνο γραμμικές! Ακριβώς γιατί μαθαίνει γραμμική συνάρτηση $\mathrm{h1} = \sum_{i=1}^{n} w_i\, x_i + b$
- Μη γραμμικές συναρτήσεις δεν μπορούν να μαθευτούν.
  - Π.χ. συνάρτηση XOR: (0,0)→0,(0,1)→1,(1,0)→1,(1,1)→0

1. (0,0)→0 : $b \leq 0$
2. (0,1)→1: $w_2 + b > 0 \Rightarrow w_2 > -b$
3. (1,0)→1: $w_1 + b > 0 \Rightarrow w_1 > -b$
4. (1,1)→0: $w_1 + w_2 + b \leq 0 \Rightarrow w_1 + w_2 \leq -b$

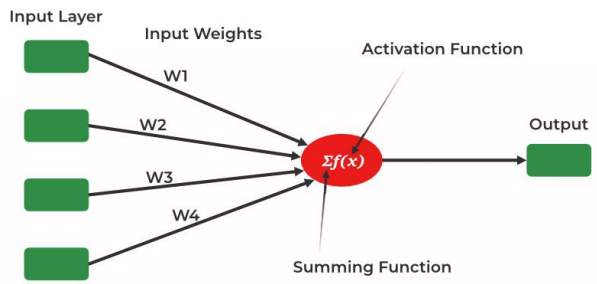
- Συνδυάζοντας τα 2, 3: $w_1 + w_2 > -2b$
- Όμως το 4: $w_1 + w_2 \leq -b$ **!αντίφαση!**

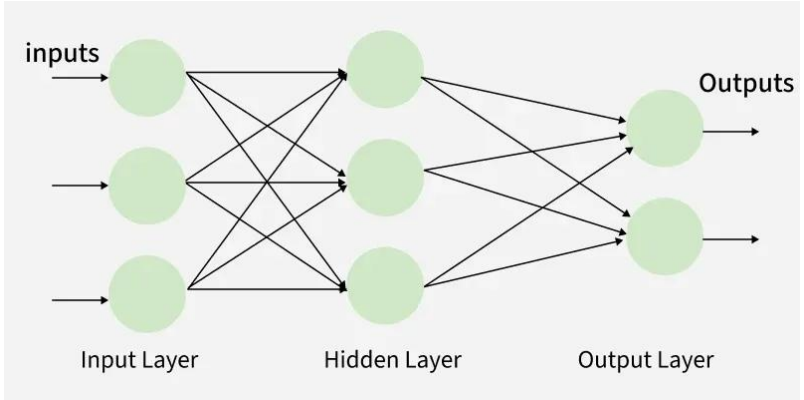
Πολυεπίπεδα νευρωνικά δίκτυα Multi-layer Perceptron

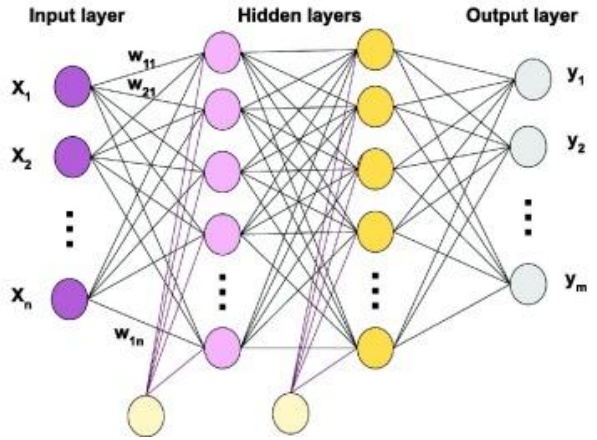
- Ο νευρώνας perceptron έχει:
  - Είσοδο: x
  - Έξοδο: y

Βασική ιδέα νευρωνικών δικτύων:

- Χρησιμοποίησε παραπάνω νευρώνες και ένωσέ τους μεταξύ τους. Χρήση **κρυφών επιπέδων** πριν την έξοδο.
  - για να έχεις περισσότερα βάρη να μάθεις, αναπαριστώντας πιο περίπλοκες συναρτήσεις
- Μπορούμε να έχουμε πολλαπλούς νευρώνες σε κάθε επίπεδο, και πολλαπλά επίπεδα
  - **Πλήρης διασύνδεση** μεταξύ των νευρώνων δύο διαδοχικών επιπέδων. Συνήθως δεν επιτρέπονται συνδέσεις μεταξύ νευρώνων που ανήκουν σε επίπεδα που δεν είναι διαδοχικά.







Αρκεί να βάλουμε παραπάνω νευρώνες;

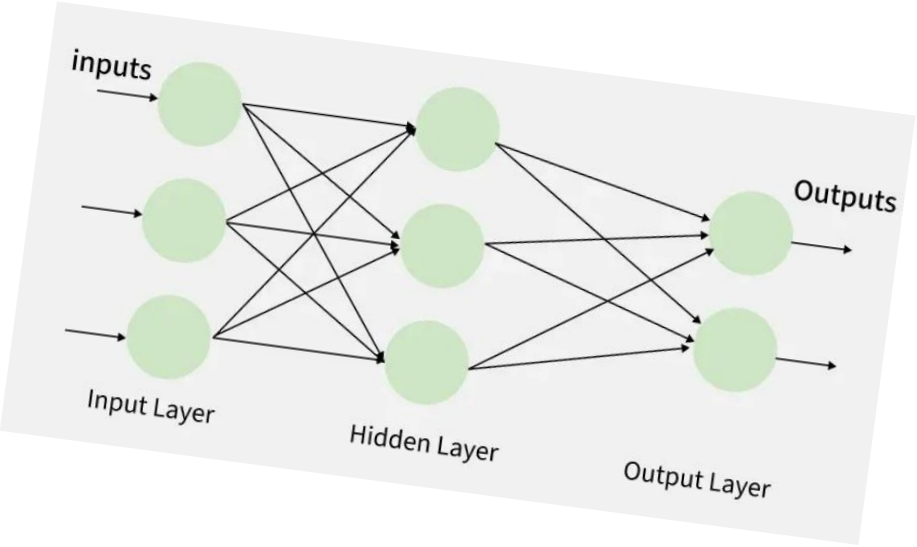
Σκεφτείτε:

ένα κρυφό επίπεδο με πολλούς γραμμικούς νευρώνες:

- $h_i = \sum_{j=1}^{n} w_{ij}\, x_j + b_i$ *για κάθε κρυφο νευρωνα* i

Και έξοδο που τους συνδυάζει γραμμικά:

- $y = \sum_{i=1}^{m} v_i\, h_i + c$



Αντικαθιστώντας τα h_i στην έξοδο:

$$y = \sum_{i=1}^{m} v_i\, h_i + c = \sum_{i=1}^{m} v_i\,(\sum_{j=1}^{n} w_{ij}\, x_j + b_i) + c = \sum_{j=1}^{n}(\sum_{i=1}^{m} v_i\, w_{ij})x_j + (\sum_{i=1}^{m} v_i\, b_i + c) = \sum_{j=1}^{n} W_j'\, x_j + b'$$

Το αποτέλεσμα **είναι πάλι μια γραμμική συνάρτηση** των εισόδων!

- **Οποιοσδήποτε αριθμός νευρώνων ή επιπέδων δεν προσθέτει εκφραστική δύναμη** — το δίκτυο παραμένει γραμμικό.

Χρήση συναρτήσεων ενεργοποίησης

Σε ένα **multi-layer perceptron (MLP)**, κάθε νευρώνας σε κάθε **κρυφό επίπεδο** παίρνει εισόδους $x_1, x_2, \ldots, x_n$ και παράγει:

$$h_i = \varphi\left(\sum_{j=1}^{n} w_{ij}\, x_j + b_i\right)$$

- Δεν είναι δηλαδή μια απλή γραμμική συνάρτηση, αλλά χρησιμοποιεί μια (μη γραμμική) **συνάρτηση ενεργοποίησης** ϕ
  - Όπως στον απλό perceptron χρησιμοποιούσαμε την βηματική (step) συνάρτηση για να παράγουμε έξοδο.
- Παραδείγματα μη γραμμικών ϕ:
  - Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$
  - ReLU: $\mathrm{ReLU}\,(z) = \max(0, \mathrm{z})$

Αυτή η «μη γραμμική μετατροπή» στο κρυφό επίπεδο μετατρέπει **γραμμικούς συνδυασμούς εισόδων** σε **μη γραμμικά χαρακτηριστικά**.

Τι αλλάζει με μη γραμμική ενεργοποίηση

- Επιλέγοντας μη γραμμική φ(π.χ. sigmoid, tanh, ReLU):

$$h_i = \varphi(\sum_j w_{ij}\, x_j + b_i)$$

- Τώρα η έξοδος είναι:

$$y = \sum_i v_i \;\varphi(\sum_j w_{ij}\, x_j + b_i) + c$$

- **Σημείωση:** η σύνθεση μη γραμμικών συναρτήσεων + γραμμικός συνδυασμός τους στην έξοδο επιτρέπει πολύπλοκες μη γραμμικές σχέσεις μεταξύ εισόδων και εξόδου.

Παράδειγμα: 1-hidden layer για x^2

- Με ReLU μπορούμε να *προσεγγίσουμε την* x^2
- Η μέθοδος αυτή ονομάζεται piecewise-linear approximation
  - **άθροισμα κομματιών γραμμικών συναρτήσεων** :

$$x^2 \approx \sum_{j=1}^{k} v_j \;\mathrm{ReLU}(x - t_j)$$

Επίπεδα εισόδου και εξόδου

- Το **επίπεδο εισόδου** ουσιαστικά απλά προωθεί το επαυξημένο διάνυσμα εισόδου x στο επόμενο επίπεδο
  - Δεν κάνει **υπολογισμούς**, δεν έχει **βάρη**, δεν έχει **ενεργοποίηση**.

Επίπεδο εξόδου:

- Το επίπεδο εξόδου μετατρέπει την τελευταία αναπαράσταση του δικτύου σε **τελική πρόβλεψη**.
- Για **ταξινόμηση** πολλών κατηγοριών (multi-class classification)**:**
  - Χρησιμοποιούμε 1 νευρώνα για κάθε κατηγορία (k νευρώνες για k κατηγορίες)
- Ο κάθε νευρώνας παράγει τιμές (scores) μέσα από τις πράξεις του.
  - Παράδειγμα εξόδου MLP για 3 κατηγορίες : $z = [2.3, 1.1, -0.5]$
- Η **«προβλεπόμενη» κατηγορία** είναι αυτή με την **μεγαλύτερη τιμή**!
- **ΠΡΟΣΟΧΗ**! Έτσι δεν ξέρουμε **πόσο σίγουρο** είναι το MLP για την πρόβλεψη
- **Λύση**: Μετατρέπουμε τις τιμές εξόδου των νευρώνων σε πιθανότητες
  - Εφαρμόζουμε πρώτα **softmax** στις τιμές

Επίπεδα εισόδου και εξόδου

Επίπεδο εξόδου: παράγει την **τελική πρόβλεψη**.

- Για **ταξινόμηση** k κατηγοριών χρησιμοποιούμε k νευρώνες.
- Ο κάθε νευρώνας παράγει τιμές (scores) μέσα από τις πράξεις του.
  - Παράδειγμα για 3 κατηγορίες: z = [2.3,1.1,−0.5]
- Μετατρέπουμε τις τιμές εξόδου των νευρώνων σε πιθανότητες
  - Εφαρμόζουμε **softmax** στις τιμές
- Γιατί softmax;
  - **Πάντα θετικές τιμές:** εύκολη κανονικοποίηση σε πιθανότητες.
  - Μεγαλύτερα scores γίνονται εκθετικά μεγαλύτερα**: ενισχύουν την πιο πιθανή κατηγορία**.
  - **Διατηρούν σχετικές διαφορές μεταξύ scores.**

- $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ Παράδειγμα: $e^{2.3} \approx 9.97$, $e^{1.1} \approx 3.00$, $e^{-0.5} \approx 0.61$ Πιθανότητες: $P_1 = 9.97/13.58 = 0.73$, $P_2 = 3.00/13.58 = 0.22$, $P_3 = 0.61/13.58 = 0.045$

Πολυεπίπεδα νευρωνικά δίκτυα
Multi-layer Perceptron

Χρησιμοποίησε παραπάνω νευρώνες και ένωσέ τους μεταξύ τους.

- Χρήση **κρυφών επιπέδων** πριν την έξοδο.
  - για να έχεις περισσότερα βάρη να μάθεις, αναπαριστώντας πιο περίπλοκες συναρτήσεις
- Μπορούμε να έχουμε πολλαπλούς νευρώνες σε κάθε επίπεδο, και πολλαπλά επίπεδα
- Χρησιμοποιούμε **μη γραμμικές συναρτήσεις ενεργοποίησης** σε κάθε νευρώνα

Έτσι, τα MLP μπορούν να αναπαραστήσουν μη γραμμικές συναρτήσεις

- με τους **γραμμικούς συνδυασμούς** των συναρτήσεων των νευρώνων!
- Περισσότεροι νευρώνες και περισσότερα επίπεδα: αναπαράσταση πιο πολύπλοκων συναρτήσεων!!

ΠΡΟΣΟΧΗ!

Χρήση λογικού αριθμού νευρώνων και επιπέδων:

- Υπολογιστικά **χρονοβόρα** τα μεγάλα δίκτυα
- Κίνδυνος **overfitting**

Πολυεπίπεδα νευρωνικά δίκτυα
Multi-layer Perceptron

Έτσι, τα MLP μπορούν να αναπαραστήσουν μη γραμμικές συναρτήσεις

- με τους **γραμμικούς συνδυασμούς** των συναρτήσεων των νευρώνων!
- Περισσότεροι νευρώνες και περισσότερα επίπεδα: αναπαράσταση πιο πολύπλοκων συναρτήσεων!!

ΠΡΟΣΟΧΗ!

Κίνδυνος **overfitting:**

- Αναπαράσταση πιο περίπλοκης συνάρτησης από ότι χρειάζεται
- Κίνδυνος για μάθηση και θορύβου
- Κακή **γενίκευση** σε νέα δεδομένα

Εκπαίδευση νευρωνικών δικτύων

Εκπαιδεύοντας ένα σύστημα θέλουμε να μειώσουμε το σφάλμα που κάνει!

- Ορίζουμε **loss (σφάλμα)** για κάθε δείγμα με βάση το τι θέλουμε να μειώσουμε.
- Πιο πλατιά χρησιμοποιημένο: **Mean Squared Error (MSE)**:

$$L = \frac{1}{N}\sum_{k=1}^{N}\left(y_i^{pred} - y_i^{true}\right)^2$$

Ο τελικός στόχος: **ελαχιστοποίηση του συνολικού σφάλματος**:

- Θέλουμε να ελαχιστοποιήσουμε το loss ως προς όλα τα **βάρη** και **bias**:

$$\min_{w,b} L$$

Gradient Descent: Η βασική ιδέα

Πώς αλλάζουμε τα βάρη για να μειώσουμε το σφάλμα

Θέλουμε να ελαχιστοποιήσουμε το loss ως προς όλα τα **βάρη** και **bias**:

$$\min_{w,b} L$$

Πρέπει να βρούμε τις καλύτερες τιμές στα βάρη

- Κάθε βάρος έχει τη δική του συνεισφορά στο loss

Για να ξέρουμε **πώς να αλλάξουμε κάθε βάρος** ώστε να μειωθεί το loss, πρέπει να υπολογίσουμε το **gradient** για αυτό το βάρος:

- Προς ποια κατεύθυνση να το αλλάξουμε (μείωση ή αύξηση)

Gradient Descent:

- Βρες την κλίση της συνάρτησης για κάθε βάρος!
- Άλλαξέ το προς τα εκεί.

Εκπαίδευση νευρωνικών δικτύων: **Gradient Descent**

Θέλουμε να ελαχιστοποιήσουμε το loss ως προς όλα τα **βάρη** και **bias**:

$$\min_{w,b} L$$

Στον νευρώνα perceptron χρησιμοποιήσαμε μια απλή μέθοδο αλλαγής των τιμών των βαρών,

- γιατί η είσοδος ήταν απευθείας συνδεδεμένη με την έξοδο!
- Τώρα θέλουμε να το κάνουμε σε **ολόκληρο το δίκτυο** — όχι μόνο για ένα νευρώνα.
  - Τι κάνουμε με τα βάρη στα κρυφά επίπεδα;

Εκπαίδευση νευρωνικών δικτύων: **Gradient Descent**

Θέλουμε να βρούμε την κλίση κάθε βάρους

- Αν δηλαδή αύξηση/μείωση του **βάρους** οδηγεί σε αύξηση/μείωση του **loss**
- Αλλάζουμε το βάρος προς την κατεύθυνση **μεγαλύτερης μείωσης του loss**
- Αυτό το βρίσκουμε με χρήση μερικών παραγώγων του σφάλματος και του βάρους:

$$w_{ij} \leftarrow w_{ij} - n\frac{\partial \mathrm{L}}{\partial w_{ij}}$$

- n: ρυθμός μάθησης
- Λόγω της **ανάγκης παραγώγισης**, δεν μπορούμε να χρησιμοποιήσουμε οποιαδήποτε συνάρτηση ενεργοποίησης
  - **Μόνο παραγωγίσημες (διαφορίσιμες)**
  - Πχ όχι την βηματική

Εκπαίδευση νευρωνικών δικτύων: **Backpropagation**

Χρήση της μεθόδου **backpropagation** (οπισθοδιάδοση)

- **Διάδοσε** το **σφάλμα** προς τους πίσω νευρώνες αναλογικά
- Μας δείχνει **πώς να αλλάξουμε ΌΛΑ τα βάρη** για να μειώσουμε τα λάθη.

Βήματα:

1. *Forward pass*: υπολογισμός εξόδου y^{pred}
2. Υπολογισμός *loss*
3. *Backward pass*: υπολογισμός gradients
   - (αλυσιδοτά, επίπεδο το επίπεδο)
4. Ενημέρωση βαρών: $w \leftarrow w - n\frac{\partial L}{\partial w}$

Πολυεπίπεδα νευρωνικά δίκτυα
Multi-layer Perceptron

Χρησιμοποίησε παραπάνω νευρώνες και ένωσέ τους μεταξύ τους.

- Χρήση **κρυφών επιπέδων** πριν την έξοδο.
  - για να έχεις περισσότερα βάρη να μάθεις, αναπαριστώντας πιο περίπλοκες συναρτήσεις
- Μπορούμε να έχουμε πολλαπλούς νευρώνες σε κάθε επίπεδο, και πολλαπλά επίπεδα
- Χρησιμοποιούμε **μη γραμμικές συναρτήσεις ενεργοποίησης** σε κάθε νευρώνα
  - Αναγκαστικά **διαφορίσιμες**

Έτσι, τα MLP μπορούν να αναπαραστήσουν μη γραμμικές συναρτήσεις

- με τους **γραμμικούς συνδυασμούς** των συναρτήσεων των νευρώνων!
- Περισσότεροι νευρώνες και περισσότερα επίπεδα: αναπαράσταση πιο πολύπλοκων συναρτήσεων!!
- ΠΡΟΣΟΧΗ στο **overfitting**

Εκπαίδευση με **gradient descent** (με τη μέθοδο **backpropagation**)

- Αναγκαστηκά παραγωγίσιμες συναρτήσεις ενεργοποίησης και παραγωγίσιμο σφάλμα.

Διαφορετικές αρχιτεκτονικές νευρωνικών δικτύων

Τα MLP μπορούν να προσεγγίσουν οποιαδήποτε συνάρτηση και να πετύχουν ακριβή πρόβλεψη

- Αλλά **δεν εκμεταλλεύεται δομή στα δεδομένα**.
- Πολλά δεδομένα έχουν **χωρική** (εικόνες), **ακολουθιακή** (κείμενο), ή **χρονική** (σειρές χρόνου) σχέση.
- Για να «μάθει» τέτοιες συναρτήσεις, ένα MLP θα χρειάζεται **πάρα πολλούς νευρώνες**

Χρειαζόμαστε άλλες ***αρχιτεκτονικές*** νευρωνικών δικτύων

Η αρχιτεκτονική ενός νευρωνικού δικτύου είναι η δομή του:

- το πώς **οργανώνονται** τα επίπεδα,
- πώς **συνδέονται** μεταξύ τους και
- τι είδους **υπολογισμούς** κάνουν.

Για να εκμεταλλευτούμε τη δομή των δεδομένων σε πολλές περιπτώσεις χρειαζόμαστε άλλες αρχιτεκτονικές, που:

- μειώνουν αριθμό **παραμέτρων**
- εκμεταλλεύονται επαναληπτικά **patterns**
- κ.α.

Διαφορετικές αρχιτεκτονικές νευρωνικών δικτύων

Όλες οι αρχιτεκτονικές βασίζονται στις βασικές αρχές που είδαμε:

- Νευρώνες που κάνουν υπολογισμούς
- Νευρώνες που συνδέονται μεταξύ τους, και βάρη στις συνδέσεις
- Εκπαίδευση με gradient descent
- Συναρτήσεις ενεργοποίησης
- κλπ

Οι βασικοί τύποι αρχιτεκτονικών που χρησιμοποιούνται σήμερα**:**

- **Convolutional Neural Networks (CNNs)** → για εικόνες, βίντεο κλπ
- **Recurrent Neural Networks (RNNs)** → για ακολουθίες / χρόνο κλπ
- **Transformers** → για κείμενο, ακολουθίες, εικόνες, όλα (η αρχιτεκτονική του ChatGPT)
- **Graph Neural Networks (GNNs)** → για γραφήματα (social networks κ.α.)
- **Autoencoders & VAEs** → για συμπίεση, παραγωγή δεδομένων κ.α.
- **Κ.α.**

Επόμενο μάθημα: Εργαστήριο

- Εισαγωγή στην χρήση τεχνικών μηχανικής μάθησης και νευρωνικών δικτύων με **python**
- Εφαρμογή διαφορετικών τεχνικών σε διαφορετικά γνωστά σύνολα δεδομένων
- Εξαγωγή συμπερασμάτων από αυτά
- Παρουσίαση εργασίας **bonus**