

Bayes Theorem

$$P(w_i|x) = \frac{P(x|w_i) \cdot P(w_i)}{\underbrace{P(x)}_{\text{apόθηση κλασεων}}} \rightarrow \sum_{i=1}^c p(x|w_i) \cdot P(w_i)$$

- $P(w_i)$ ου βρίσκεται ανευθείς μέσα από τα training data

$$\sum_{i=1}^c P(w_i|x) = 1$$

- BDR: w_1 if $P(w_1|x) > P(w_2|x)$

(ανήγη nepiñzwoñ:
 $a_1 = \text{επιλεγόμενη κλάση } w_1$
 $a_2 = \text{'' '' '' } w_2$)

Tώρα αντί να επιλέγουμε κλάσεις, θα επιλέγουμε ενέργειες (a_i)

- Έχουμε "loss" functions, π.χ. ου $\lambda(a_2|w_1) = 5 \lambda(a_1|w_2) = 50 \epsilon$

$$\text{ενώ } \lambda(a_1|w_1) = \lambda(a_2|w_2) = 0$$

μεγαλύτερο κόσος αν
κάνω αυτό το λάθος

- Το ρισκό ζου να επιλέξω την ενέργεια a_i δινεται ως

$$R(a_i|x) = \sum_{j=1}^c \lambda(a_i|w_j) P(w_j|x) \rightarrow \text{που καλει minimize αυτό} \\ (\text{ονομάζεται Bayes Risk})$$

- Αντί να χρησιμοποιούμε τα ρισκά, θα χρησιμοποιούμε τα discriminant functions, σε συνδίκα σε λήθος, που γιατί:

επιλέξει w_i αν $g_i(x) > g_j(x)$ για κάθε $i \neq j$

Υπάρχουν ανείρες επιλογές $g_i(x)$ (δεν είναι μοναδικά)

$$\sum \text{την ανήγη μορφή: } g_i(x) = P(w_i|x)$$

- Τι μορφή έχει το decision boundary?

Gaussian

- Θεωρούμε $p(x|w_i) \sim N(\mu_i, \Sigma_i)$, π.χ. αν έχουμε 5 features

θα έχουμε nivales $\mu_i = 5 \times 1$ και $\Sigma_i = 5 \times 5$

- Αν $\Sigma_i = \sigma^2 I$ (δηλαδή $\text{Cov}(\text{feature}_i, \text{feature}_j) = 0 \rightarrow$ ανεξάρτητα features μεταξύ τους)

decision boundary: linear hyperplane με $g_i(x) = w_i x + b_i$

- Αν $\Sigma_i \neq \Sigma$ (δηλαδή $\text{Cov}(\text{feature}_i, \text{feature}_j) \neq 0$)

decision boundary: ήτοι linear hyperplane με $g_i(x) = w_i x + b_i$

η διαφορά είναι ότι έχουμε ελλείψεις αντί για κύκλους στις ισούγιεις καμπύλες (δες σχηματάρια)

- Αν Σ_i (ηρακτικά σημαίνει ότι τα features πιο ώρια αντί τις κλάσεις δεν ακολουθούν τις ίδιες κανονίες αντί κλάση σε κλάση, αλλά εδώ μας ενδιαφέρει το covariance matrix ήτοι μέσες τιμές)

decision boundary: Quadratic (καμπύλη)

Hard Margin SVM

Av oi κλασεις ειναι linearly separable, μπορουμε να μεχιστωνοησουμε zo margin μεταξυ zwv support vectors (za σημεια nou ειναι υπεύθυνα για zo οριο ανδφασης).

Tiwpia arji για $y_n(x) \geq 0$ kai $y_n(x) < 0$, kai zafoumpe $y(x) \geq 1$ kai $y(x) \leq -1$

- Για za support vectors: $y(x) = 1 \text{ } \& \text{ } -1$

Meza zo training (όzav zεσκάρουμε), zεσκάρουμε κλασσικά $y(x) \geq 0$ $y(x) < 0$

- Mεγεθος margin: $\frac{1}{\|w\|}$

• Για va βρεις zo w nperei kavovika va napelis Lagrangian function, KKT conditions και. Tedika katalinjiesis va βρεις éva Lagrange Multiplier μ_n για κάθε σημειο x_n . $\mu_n > 0 \rightarrow x_n$ support vector $\mu_n = 0 \rightarrow x_n$ μακριà apo zo decision boundary

$$\text{Apa } w = \sum_{n=1}^{N_s} \mu_n t_n x_n$$

- Για va βρεις zo w_0 naipvelis $t_n (w^t x_n + w_0) = 1$ (για kanolo support vector x_n)

Soft Margin SVM

Tiwpia exoume $y(x) \geq 1 - \xi_n$ kai $y(x) \leq -1 + \xi_n$

κλαση +1

καλύτερα

$\xi_n y(x_n) \geq 1 - \xi_n$

$\xi_n = 0$ ózav correctly classified (ézow ñ návw ozo margin)

To ξ_n naipveli zifies $0 < \xi_n \leq 1$ " " " (evrás zo margin)
 $\rightarrow \xi_n > 1$ ózav misclassified

$C \rightarrow \infty$ = hard margin

Exoupe kai éva hyperparameter C órou $C = 0$ = infinite margin

• Av C μεγάλο (n.x. 1000) \rightarrow katalinjoumpe σε mikro margin

• Av C mikro (n.x. 0,1) \rightarrow " " " μεγάλο "

Kernel Trick

Av oi κλασεις δεν ειναι γραμμικα διαχωρισιμες, μπορεις na probarais za σημεia σε μεγαλύτερη διάσταση.

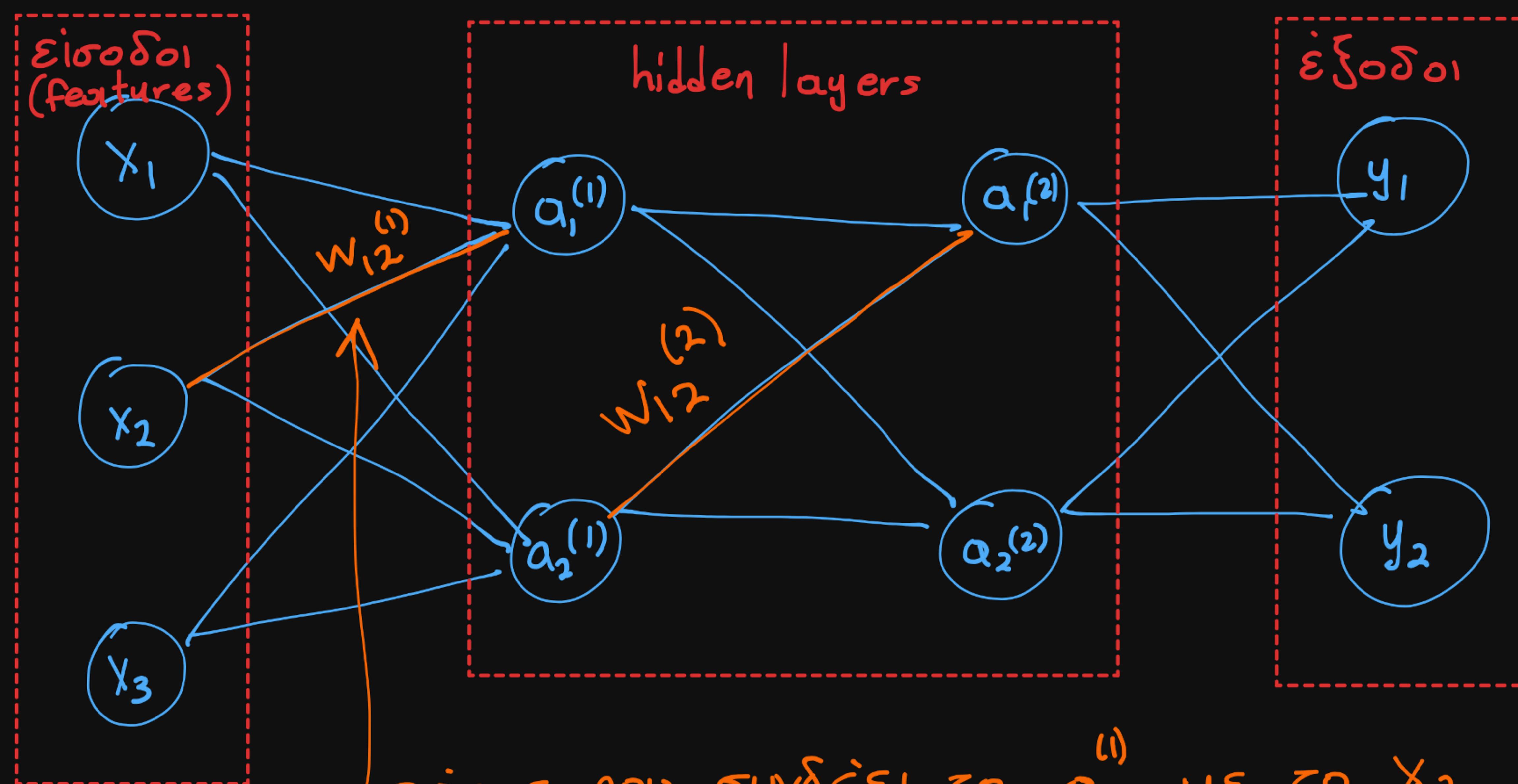
Me zo kernel trick zo aozio ειναι òzi δεν probarais za σημεia σην pragmatikotza, γiazi ειναι unologisiotika δùsokolo (φαρζάσου àneires διαστάσεις).

Unologijoumpe zo esωzepiko givómeno zo low-dimensional xwpo, zo xrelofisoumoumpe σε κánoia (kernel) συáρtηση, kai ειναι σαν na unologisoumpe zo es. givómeno zo high-dimensional xwpo

- Klassika kernels: Polynomial, RBF, Tanh

Neural Networks (NN)

Εμεις εδω ασχολουμασε με Multi Layer Perceptron NN ην ανεκοιφεται ως εξις:



βάρος που συνδέει το $a_1^{(1)}$ με το x_2
 $w_{1,2}$

το ξύει το παρακάτω:

$$a_1^{(1)} = f(w^T \cdot x + b_1^{(1)})$$

↑ ↑ ↑
 $w = 3 \times 1$ $w^T = 1 \times 3$ scalar
activation function

Πιο αντί μπορούμε να το γράψουμε ως:

$$a^{(1)} = f(w^T x + b^{(1)})$$

↑ ↑ ↑
 2×1 $w = 3 \times 2$ 3×1 2×1

η μπορεις να θεωρησεις εξ' αρχης $W = 2 \times 3$
οποτε θα έχεις $a^{(1)} = f(W \cdot x + b^{(1)})$

To activation function μας επιτρέπει να μην έχουμε linear decision boundaries. Κλασσικές επιλογές: sigmoid, ReLU, tanh

Με ReLU προτιμάμε την αύξηση του depth (βάσου της παρανάνω hidden layers) πάντα την αύξηση του width (αριθμός hidden units ανά layer)
 $a_1, a_2 \in \mathbb{R}$

Ο συνολικός αριθμός των ηδαμέζων του NN είναι όταν τα w και όταν τα b (στο δικό μας παράδειγμα $3 \cdot 2 + 2 + 2 \cdot 2 + 2 + 2 \cdot 2 + 2 = 20$)

Για να βρούμε αυτες τις ηδαμέζων, εφαρμόζουμε SGD αφού οι συναρτήσεις κώστους δεν είναι κυρτές (έχουν local minimum που θέλουμε να αποφύγουμε)

Bootstrapping

Χρησιμοποιείται για την δημιουργία B dataset χρησιμοποιώντας ένα αρχικό dataset \rightarrow γράβας N συστοιχεία ανά αυτό με ενανθεση.

Feature