Bayes Theorem

αριθμός κλάσεων

$$P(\omega_i|x) = \frac{P(x|\omega_i)\cdot P(\omega_i)}{p(x)} \qquad p(x) \rightarrow \sum_{i=1}^{c} p(x|\omega_i)\cdot P(\omega_i)$$

- $P(\omega_i)$ το βρίσκεις απευθείας μέσα απ'τα training data
- $\sum_{i=1}^{c} P(\omega_i|x) = 1$
- BDR: ω_1 if $P(\omega_1|x) > P(\omega_2|x)$

(απλή περίπτωση:
a_1 = επιλέγω κλάση ω_1
a_2 = " " ω_2)

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

- Έχουμε "Loss" functions, π.χ. το $\lambda(a_2|\omega_1) = 5\lambda(a_1|\omega_2) = 50$€
  ενώ $\lambda(a_1|\omega_1) = \lambda(a_2|\omega_2) = 0$

  5 φορές μεγαλύτερο κόστος αν κάνω αυτό το λάθος

- Το ρίσκο του να επιλέξω την ενέργεια a_i δίνεται ως

$$R(a_i|x) = \sum_{j=1}^{c} \lambda(a_i|\omega_j)P(\omega_j|x)$$

→ επιλέγουμε την ενέργεια a_i που κάνει minimize αυτό (ονομάζεται Bayes Risk)

- Αντί να χρησιμοποιούμε τα ρίσκα, θα χρησιμοποιούμε τα discriminant functions, c συνολικά σε πλήθος, που μας λένε:

$$\text{επέλεξε } \omega_i \text{ αν } g_i(x) > g_j(x) \text{ για κάθε } i \neq j$$

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

Στην απλή μορφή: $g_i(x) = P(\omega_i|x)$

- Τι μορφή έχει το decision boundary?

- Θεωρούμε $p(x|\omega_i) \sim N(\mu_i, \Sigma_i)$ (Gaussian), π.χ. αν έχουμε 5 features
  θα έχουμε πίνακες $\mu_i = 5\times 1$ και $\Sigma_i = 5\times 5$

- Αν $\Sigma_i = \sigma^2 I$ (δηλαδή $Cov(feature_i, feature_j) = 0$ → ανεξάρτητα features μεταξύ τους)
  decision boundary: linear hyperplane με $g_i(x) = W_i x + b_i$

- Αν $\Sigma_i = \Sigma$ (δηλαδή $Cov(feature_i, feature_j) \neq 0$)
  decision boundary: πάλι linear hyperplane με $g_i(x) = W_i x + b_i$
  η διαφορά είναι ότι έχουμε ελλείψεις αντί για κύκλους στις ισοϋψείς καμπύλες (δες σχηματάκια)

- Αν Σ_i (πρακτικά σημαίνει ότι τα features πίσω από τις κλάσεις δεν ακολουθούν τις ίδιες κατανομές από κλάση σε κλάση, αλλά εδώ μας ενδιαφέρει το covariance matrix όχι μέσες τιμές)

decision boundary: Quadratic (καμπύλη)

> Όμως πως βρίσκουμε το $p(x|w_i)$?

Χρησιμοποίησε τα training data για να την βρεις

> Ναι αλλά πως?

Safe επιλογή είναι η Maximum Likelihood (ML) Estimation

Maximum Likelihood (Θεωρούμε ότι τα data ακολουθούν Gaussian)

1: Πάρε ένα τυχαίο θ → το θ περιλαμβάνει και το μ και το Σ (ή ένα από τα 2, π.χ. το μ για αρχή)

π.χ. θεωρούμε ότι τα data μας έχουν μέση τιμή 5 και διασπορά 16 (αν έχουμε 1 feature. Αν είχαμε παραπάνω αυτά θα ήταν πίνακες)

2. Υπολόγισε το likelihood (πιθανότητα αλλά όχι ακριβώς) τα data σου να έχουν αυτή την κατανομή με τις παραμέτρους θ

$$p(D|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

→ σ.π.π. Gaussian με $\mu = 5$, $\sigma^2 = 16$

sample (π.χ. $x = 2,7$)

3. Ξανακάνε τα ίδια για άλλο θ

4. Επέλεξε το θ που μας δίνει το μέγιστο $p(D|\theta)$ → $\hat{\theta}_{ML}$

(Note: αν θες να λύσεις το πρόβλημα χωρίς να παίρνεις τυχαία θ, καλύτερα να πάρεις το log-likelihood $l(\theta)$ στο βήμα 2 και να λύσεις ένα σύστημα 2 εξισώσεων (άγνωστοι το μ, σ^2), αν πάρεις την παράγωγο (gradient) του $l(\theta)$ ίση με μηδέν.)

Bayesian Estimation

Δεν θα μπω σε ανάλυση, αλλά οι σημαντικές διαφορές με το ML είναι:

- Βρίσκει την πλήρη κατανομή πιθανότητας του θ (δεν βρίσκει απλά μια τιμή όπως το ML)
- Λαμβάνει υπ'όψην την prior κατανομή $p(\theta)$, δηλαδή της κατανομής των παραμέτρων μ και σ^2 (άρα είναι ένας hyperparameter) και δίνει καλύτερα αποτελέσματα

Naive Bayes

Θεωρούμε ότι τα features είναι ανεξάρτητα μεταξύ τους

οπότε αν τα samples μας έχουν π.χ. 2 features $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, τότε:

$$p(x|w_i) = p(x_1|w_i) \cdot p(x_2|w_i)$$

μονοδιάστατες κατανομές

Με τις παραπάνω 2 μεθόδους κάναμε υποθέσεις για τις κατανομές των data. Τώρα θα δούμε μη-παραμετρικές μεθόδους.

• Parzen Windows

Χώνεις τα δεδομένα σου μέσα σε "slots" (hypercubes), αλλά αντί να προσθέτεις 1 μέσα στο hypercube, βάζεις μια Gaussian με κέντρο το σημείο που προσθέτεις. Αν τις προσθέσεις φτιάχνεις μια smooth κατανομή

• k-NN

Η απλή έκδοση k-NN classifier απλά βλέπει τους k πιο κοντινούς γείτονες απ'το training sample, και επιλέγουμε την κλάση με τους περισσότερους γείτονες.

Η περίπλοκη έκδοση βρίσκει k γείτονες από την κάθε κλάση, και χρησιμοποιώντας τις ακτίνες r_1, r_2 (δηλ. τους όγκους V_1, V_2) μπορούμε να κάνουμε μια εκτίμηση:

training samples στην κλάση w_1

$$\text{assign } x \text{ το } w_1 \text{ if } \quad \frac{V_2}{V_1} > \frac{N_1}{N_2} \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(w_2)}{P(w_1)}$$

> Αν οι κλάσεις είναι γραμμικά διαχωρίσιμες, τι μπορώ να κάνω?

Θα χρησιμοποιήσεις linear discriminant functions της μορφής $y(x) = w^T x + w_0$

όπου αν $y(x) \geq 0 \to \omega_1$ και αν $y(x) < 0 \to \omega_2$

- Το w είναι κάθετο στο όριο απόφασης
- Το w_0 ορίζει την απόσταση από την αρχή των αξόνων
- Το $y(x)$ ορίζει την απόσταση του σημείου x από το decision boundary

> Πως βρίσκουμε τα w και w_0?

Least Squares

- Βρίσκεις το $W_{LS} = (X^T X)^{-1} X^T \cdot T$

$$X \in \mathbb{R}^{N \times (d+1)}, \quad T \in \mathbb{R}^{N \times c}, \quad W_{LS} \in \mathbb{R}^{d+1 \times c}$$

δεν είναι πιθανότητες btw

- Βρίσκεις το $\vec{y}(x) = W_{LS} \cdot X = \begin{bmatrix} y_1(x) \\ \vdots \\ y_c(x) \end{bmatrix}$ και επιλέγεις την κλάση i με το $\max y_i(x)$

Δεν είναι γρήγορος αλγόριθμος και ούτε πολύ καλός :(

προβλήματα masking $c>2$ και επηρεάζεται από outliers

Perceptron

Για να βρεις το w πρέπει να εφαρμόσεις (Stohastic) Gradient Descent

- SGD: $w^{i+1} = w^i - \eta \nabla E_n(w^i) = w^i + \eta x_n t_n$

τυχαίο w για $i=0$ misclassified

(Για να τσεκάρεις αν είναι misclassified θα ελέγξεις το $w^T \cdot x_n \cdot t_n \leq 0$)

π.χ. αν σου δίνει 4 2D σημεία, βάζεις στην 1η στήλη 1 σε όλα (διαστάσεις (3×1)), και μετά εφαρμόζεις τον παραπάνω τύπο για κάθε misclassified σημείο. Αφού περάσεις απ' όλα τα σημεία μια φορά (epoch 1), συνεχίζεις στο epoch 2 κτλ. μέχρι να είναι όλα σωστά classified

- SGD με minibatch: το $x_n t_n$ δίνει ένα 1×3 διάνυσμα (για κάθε σημείο), οπότε αν έχεις ένα minibatch με 2 σημεία x_1, x_2 θα κάνεις $w^{i+1} = w^i + \eta(x_1 t_1 + x_2 t_2)$ → misclassified

- GD: κάνεις $w^{i+1} = w^i + \eta(\underbrace{x_1 t_1 + x_2 t_2 + \dots + x_n t_n}_{\text{όλα τα misclassified}})$

Least Mean Squares (LMS)

Έχουμε $w^{i+1} = w^i + \eta(t_n - (w^i)^t x) x^t$, η διαφορά εδώ είναι ότι τρέχουμε αυτόν τον τύπο για κάθε σημείο ανεξάρτητα απ' το αν είναι misclassified ή όχι. Σχετικά με το "η":

- Αν "η" πολύ μεγάλο, μπορεί να μην φτάσει κοντά στην optimal λύση
- Αν "η" μικρό, τότε θα συγκλίνει αργά

Στον perceptron πάντα συγκλίνουμε ανεξάρτητα το $\eta > 0$ (αν είναι linearly separable οι κλάσεις)

Logistic Regression

Έχουμε $w^{i+1} = w^i - \eta(y_n - t_n) x_n$ όπου:

- $y_n = \sigma(w^t \cdot x_n) = \dfrac{1}{1 + e^{-w^t x_n}}$
- $t_n = \{\overset{\omega_2}{0}, \overset{\omega_1}{1}\}$

οπότε κάνουμε την ίδια διαδικασία (χρησιμοποιούμε όλα τα σημεία για πολλά epochs).

Είναι κυρτή η συνάρτηση κόστους άρα δεν έχουμε local minimum

Για να κάνουμε classification επιλέγουμε ω_1 αν $\sigma(w^t \cdot x) \geq \frac{1}{2}$