ML EXAMS 18/12/2023

EX. 1

if B = b1 and A = a1 then [-] ①

if B = b1 and A = a2 and C = c1 then [+] ②

if B = b1 and A = a2 and C = c2 then [+] ③

if B = b1 and A = a2 and C = c3 then [-] ④

if B = b1 and A = a3 then [+] ⑤

if B = b2 and C = c1 then [-] ⑥

if B = b2 and C = c2 then [-] ⑦

if B = b2 and C = c3 then [+] ⑧

1.2

An hypothesis h is consistent with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for each training sample $(x, c(x))$ in D

$$\text{consistent}(x, c) \Leftrightarrow h(x) = c(x) \quad \forall x \in D$$

1.3

S1 is consistent because of ①

S2 is consistent because of ③

S3 is consistent because of ⑧

S4 is NOT consistent because of ⑦

EX. 2

2.1

$h_{MAP} = \arg\max P(h|D) = \arg\max \frac{P(D|h)P(h)}{P(D)}$ if we assume $P(h_i) = P(h_j)$

we obtain $h_{ML} = \arg\max_{h \in H} P(D|h)$

2.2

Naive bayesian classifier uses conditional independence to approximate the solution more in detail

$$P(X, Y | Z) = P(X | Y, Z) P(Y | Z) = P(X|Z) P(Y|Z)$$

assuming a target function $f: X \to V$ where each x is describe by attribute $\langle a_1, \dots, a_n \rangle$ we compute

$$\arg\max P(v_j | x, D) = \arg\max P(v_j | a_1, \dots, a_n, D)$$

with Bayesian assumption

$$P(a_1, \dots, a_n | v_j, D) = \prod_{\text{valj}} P(a_i | v_i, D)$$

we obtain $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \; P(v_j|D) \prod_i P(a_i|v_j, D)$

2.3

Naive bayes classifier is based on two assumptions

① conditional independence

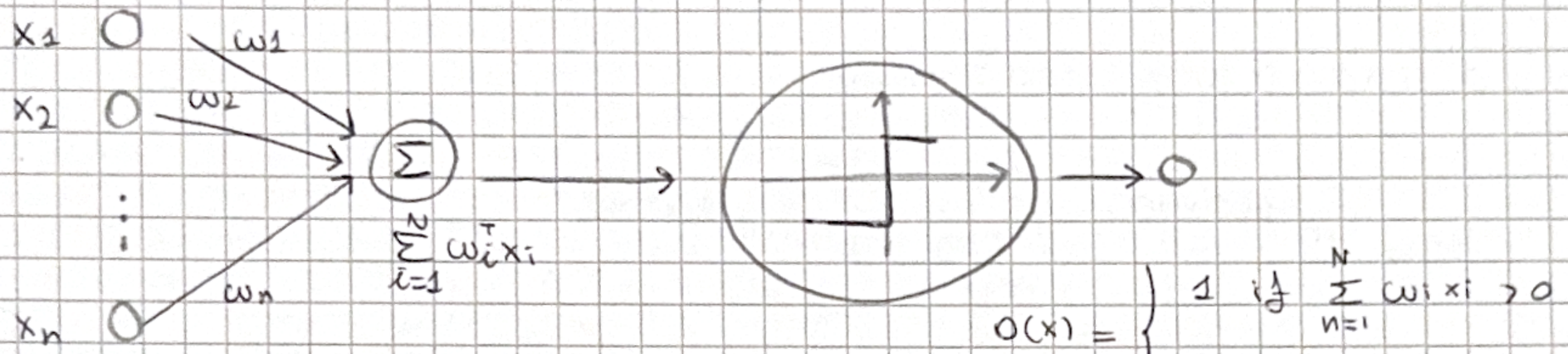
$$P(X,Y|Z) = P(X|Y,Z)P(Y|Z) = P(X|Z)P(Y|Z)$$

② Bayesian assumption

$$P(a_1, \dots, a_n|v_j, D) = \prod_i P(a_i|v_j, D)$$

EX. 3

3.1



we want minimize the squared error function

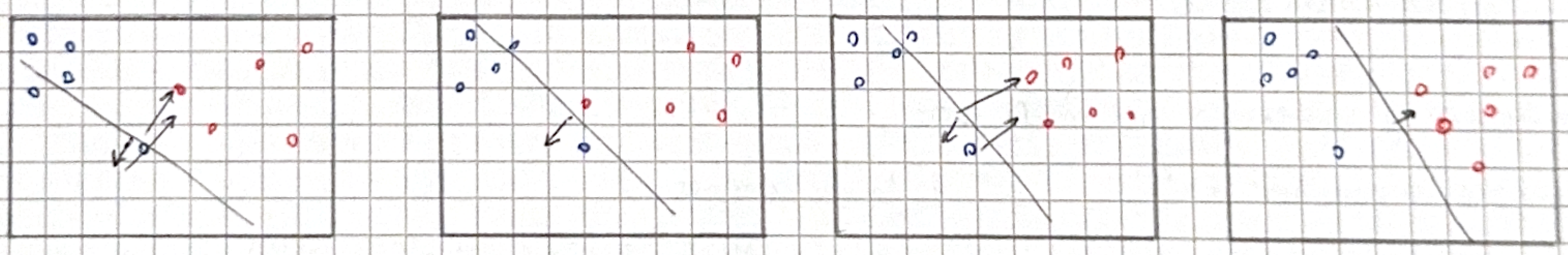
& assuming $E(w) = \frac{1}{2} \sum_{n=1}^{N} (t_n - o(n))^2 = \frac{1}{2} \sum_{n=1}^{N} (t_n - w_i^T x_n)^2$

$$\frac{\partial E}{\partial w} = \sum_{n=1}^{N} (t_n - w^T x_n)^{(2)} (-x_{i,n})$$

Algorithm will be with this update rule of weights

$w_i \leftarrow w_i + \Delta w_i$ where $\Delta w_i = -\eta \frac{\partial E}{\partial w} = +\eta \sum_{n=1}^{N} (t_n - w^T x_n)(x_{i,n})$

3.2



EX 4.

4.1

Consider a linear model $y(x,w) = w^T x$ and a dataset $D = \{(x_n, t_n)\}_{n=1}^{N}$ &

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (w^T x_n - t_n)^2 + \lambda w^T w$$

$$E(w) = (t - w^T x)^T (t - w^T x)^2 + \lambda \|w\|^2$$

where $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix}$ and $t_2 = \begin{bmatrix} t_1 \\ \vdots \\ t_n \end{bmatrix}$

optimal solution $\nabla E(w) = 0$

$$y(x, w^*) = \sum_{n=1}^{N} a_n w_n^T x \quad \text{where}$$

$$a_n = (XX^T + \lambda I_N)^{-1} t \quad \text{and} \quad XX^T = \text{gram matrix} = \begin{bmatrix} x_1^T x_1 & \cdots & x_1^T x_M \\ \vdots & \ddots & \vdots \\ x_N^T x_1 & \cdots & x_N^T x_M \end{bmatrix}$$

for kernelized version we use kernel trick where we substitute inner product $x^T x$ with kernel function $k(x', x)$

$$\text{gram matrix} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_M) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_M) \end{bmatrix}$$

so we will have $y(x, a) = \sum_{n=1}^{N} a_n k(x', x)$ and

4.2

$f : \mathbb{R}^4 \to C$ with $|C| = 3$ and D with 100 sample

$X \in \mathbb{R}^{100 \times 4}$ $K \in \mathbb{R}^{100 \times 100}$ and $t \in \mathbb{R}^{100 \times 3}$

EX. 5

5.1

the role of backpropagation is to compute the gradient in fact in not a phase of learning but just compute the gradient in an efficient way different is for stochastic gradient descent that is an learnig algorithm that allow to model to find the best values of weights used for solve the task

5.2

STOCHASTIC GRADIENT DESCENT

Require learning rate $\eta > 0$

Require initial values of $\theta^{(1)}$

$k \leftarrow 1$

while stopping criterion not meet DO:

sample a subset $\{x^{(1)}, \ldots, x^{(m)}\}$ from dataset

Compute the gradient estimate $g \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}, \theta^{(k)}), \epsilon^{(i)})$

update $\theta^{(k+1)} \leftarrow \theta^{(k)} - \eta g$

$k \leftarrow k + 1$

end while

$\eta > 0 \to$ always $\eta \ll 0$ can happen still never converge

$$P(C_3|X) = \frac{P(X|C_3)P(C_3)}{P(X)} = \frac{P(X|C_3)P(C_3)}{P(X|C_1)P(C_1)+P(X|C_2)P(C_2)+P(X|C_3)P(C_3)}$$

Considering a parametric mode $P(X|C_i) = N(x; \mu_i; \Sigma)$

the number of parameter will be

Σ = size of covariance matrix $d\frac{(d+1)}{2}$ because is symmetric

μ = dimension of 3 (numb of class)

now considering:

$$a = \ln\left(\frac{P(X|C_1)P(C_1)}{P(X|C_2)P(C_2)+P(X|C_3)P(C_3)}\right) = \ln\left(\frac{N(x;\mu_1;\Sigma)P(C_1)}{N(x;\mu_2;\Sigma)P(C_2)+N(x;\mu_3;\Sigma)P(C_3)}\right)$$

$$= w^T x + w_0$$

thus

$$P(C_1|X) = \sigma(w^T x + w_0)$$

now because $P(C_1) = \pi_1$ $P(C_2) = \pi_2$ $P(C_3) = 1 - (\pi_1 + \pi_2)$

we need to estimate $\pi, \mu_1, \mu_2, \Sigma$

EX. 6

6.1

parama = ((100 · 50) + 50) + ((50 · 10) + 10) = 10060 trainable parama

(100 · 50) → weights; + 50 → bias; (50 · 10) → weights; + 10 → bias

6.2

because we are dealing with multiclass classification problem
output units → softmax activation function

$$y_i = \text{softmax}(a^{(i)}) = \frac{\exp(a^{(i)})}{\sum_j \exp(a_j)}$$

6.3

Loss function categorical cross-entropy

$$J_i(\theta) = E_{x,y,p}[-\ln(\text{softmax}(a^{(i)}))]$$

where $a^{(i)} = w_i^T h + b_i$

I choose this loss function because we are dealing with categorical multiclass classification problem