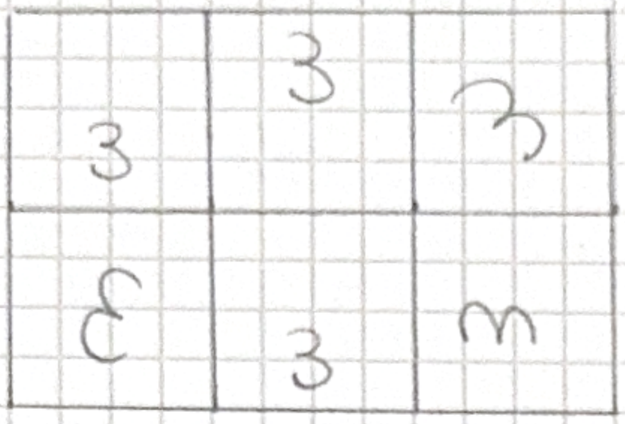
EX.6

1.

try to solve the problem of dimensionality reduction feature extraction and comprension visualization

2.



we have 36×36×3 RGB images the dimensionality reduction of dataset is 36×36×3 but the intrinsic parameter are 3 2 for translation one for rotation so with dimensionality reduction we try to $\mathbb{R}^{w \times h \times d} \rightarrow \mathbb{R}^3$

ML EXAMS 16/01/2023

EX. 1

1.1

if C=C1 and B=b1 then NO ①

if C=C1 and B=b2 then YES ②

if C=C2 and A=a1 then YES ③

if C=C2 and A=a2 and B=b1 then YES ④

if C=C2 and A=a2 and B=b2 then YES ⑤

if C=C2 and A=a3 then NO ⑥

if C=C3 then NO ⑦

1.2

S1 is comprintent because of ①

S2 is comprintent because of ④

S3 is comprintent because of ⑦

S4 is NOT comprint because of ②

1.3

accuracy $\frac{3}{4}$ = $\frac{\text{number of comprintent instances}}{\text{number of all instances}}$ = 0.75

EX. 2

2.1

LOGISTIC REGRESSION is a probabilistic discriminative model based on maximum likelihood

consider a dataset with two classes $D = \{(\tilde{x}_n, t_n)\}_{n=1}^{N}$ $t_n \in (0,1)$

likelihood function:

$$p(t|\tilde{w}) = \prod_{n=1}^{N} y_n^{t_n} (1-y_n)^{(1-t_n)}$$

with $y_n = p(C_1|\tilde{x}_n)$

where t_n is the value in the dataset corresponding to x_n

y_n: is the portion of the current model $\tilde{w}$ to x_n

cross-entropy error function (negative likelihood)

$$E(\tilde{w}) = -\ln(p(t|\tilde{w})) = -\sum_{n=1}^{N} [t_n \ln(y_n) + (1-t_n)\ln(1-y_n)]$$

what we are try to do with logist regression is to minimize $E(\tilde{w})$

$$w^* = \text{argmin}(E(\tilde{w}))$$

2.2

compute prediction with $\tilde{w}_1^T = (2, 0, -2)$ for the first point

$\tilde{w}_1^T x_i = 2 \cdot 0 + 0 \cdot 0 + (-2 \cdot 1) = -2$

the sigmoid of $-2 \simeq 0$

now for the second point

$\tilde{w}_1^T x_i = 2 \cdot 1 + 0 \cdot 2 + (-2 \cdot 3) = -4$

the sigmoid of $-4 \simeq 0$

now for the third function

$\tilde{w}_1^T x_i = 2 \cdot 4 + 0 \cdot 4 + (-2 \cdot 1) = 2$

the sigmoid of $2 \simeq 1$

so the result is $(0, 0, 1)$ that are the exct point of target

now we do the same for $\tilde{w}_2^T = (-2, -2, 0)$

$\tilde{w}_2^T x_i = -2 \cdot 0 + -2 \cdot 0 + 1 \cdot 4 = 4 \rightarrow 1$

$\tilde{w}_2^T x_i = -2 \cdot 1 + -2 \cdot 2 + 4 \cdot 3 = 6 \rightarrow 1$

$\tilde{w}_2^T x_i = -2 \cdot 4 + -2 \cdot 4 + 1 \cdot 4 = -12 \rightarrow 0$

$\tilde{w}_1^T$ fits better the solution

EX 3

3.1

the model of one state MDP is $< \{x_0\}, A, \delta, r >$ where

- $\{x_0\}$ = unique state
- A = finite set of action
- δ = transition function $f: X \times A \Rightarrow X$
- r = reward function reach

optimal policy $\pi^*(x_0) = a_i$

3.2

is like before the solution for the problem is an optimal policy $\pi^*(x_0) = a_i$

and π^* is optimal if and only if any other policy π

$$V^{\pi^*}(x) \geq V^{\pi}(x) \quad \forall x$$

~~and else~~

and $V^{\pi}(x) = E(R_1 + \gamma R_2 + \gamma^2 R_3 + \dots)$ is cumulative discount reward where γ is discount factor

3.3

~~because~~ we have a stochastic behavior and unknown ~~reward~~ functions the algorithm is

non tengo conto del caso stocastico

1. for each $a_i \in A$
   - execute a_i and collect reward r_i
2. Optimal policy $\pi^*(x_0) = a_i$ with $i = \underset{i=1,\dots,|A|}{\operatorname{argmax}} r(i)$

~~we~~ need exctly $|A|$ iterations

1. initialize $\theta_{(0)}[i] \leftarrow 0$ and $c(i) \leftarrow 0$ for $i = 1, \dots, |A|$
2. for each time $t = 1, \dots, T$ until termination condition
   - choose an action a
   - execute $a_{(t)}$ and collect reward $r_{(t)}$
   - increment $c(\hat{i})$
   - update $\theta_{(t)}[\hat{i}] \leftarrow 1/c[\hat{i}] \left(r_{(t)} + (c[\hat{i}] - 1)\, \theta_{(t-1)}[\hat{i}] \right)$
3. optimal policy $\pi^*(x_0) = a_i$ with $i = \underset{i=1,\dots,|A|}{\operatorname{argmax}} \theta_{(T)}[i]$

for choose the action we can use an algorithm like ε-greedy where $0 \leq \varepsilon \leq 1$

$\varepsilon \rightarrow$ random action

$1 - \varepsilon \rightarrow$ best action

EX 4

4.1

paramo = $7 \times 7 \times 3 \times 3 + 5 = 740$

4.2

$$w_{out} = 128 - \frac{7}{1} + 2 \cdot 1 + 1 = 124$$

$$h_{out} = 128 - \frac{7}{1} + 2 \cdot 1 + 1 = 124$$

4.3

feature map 64 consider ~~width=36 height=36~~

kernel dimension 5×5 ~~width=32~~ images in input 32×32×3

~~feature~~

paramo : $5 \times 5 \times 3 \times 64 + 64$

EX 5

5.1

SVM aims to maximum margin for letter scores by going more in detail a margin is defined as: $\frac{|y(x_n)|}{\|w\|}$ so the goal of SVM is: given data and an hyperplane

$$\min_{n=1,\dots,N} \frac{|y(x_n)|}{\|w\|} = \dots = \frac{1}{\|w\|} \min_{n=1,\dots,N} [t_n(w^T x_n + x_0)]$$

so our goal is

$$w^*, w_0^* = \arg\max_{w, w_0} \frac{1}{\|w\|} \min_{n=1,\dots,N} [t_n(w^T x_n + x_0)]$$

in the canonical representation of the problem the maximum margin hyperplane can be set as an optimization problem

$$w^*, w_0^* = \arg\max \frac{1}{\|w\|} = \arg\min \frac{1}{2} \|w\|^2$$

this can be solved by using lagrange multiplier:

$$w^* = \sum_{n=1}^{N} a_n^* t_n x_n$$

considering also support vector $SV = \{x_k \in D \mid t_k y(x_k) = 1\}$ we obtain:

$$w_0 = \frac{1}{|SV|} \sum_{x_k \in SV} \left(t_k - \sum_{x_j \in S} a_j^* t_j x_k^T x_j \right)$$

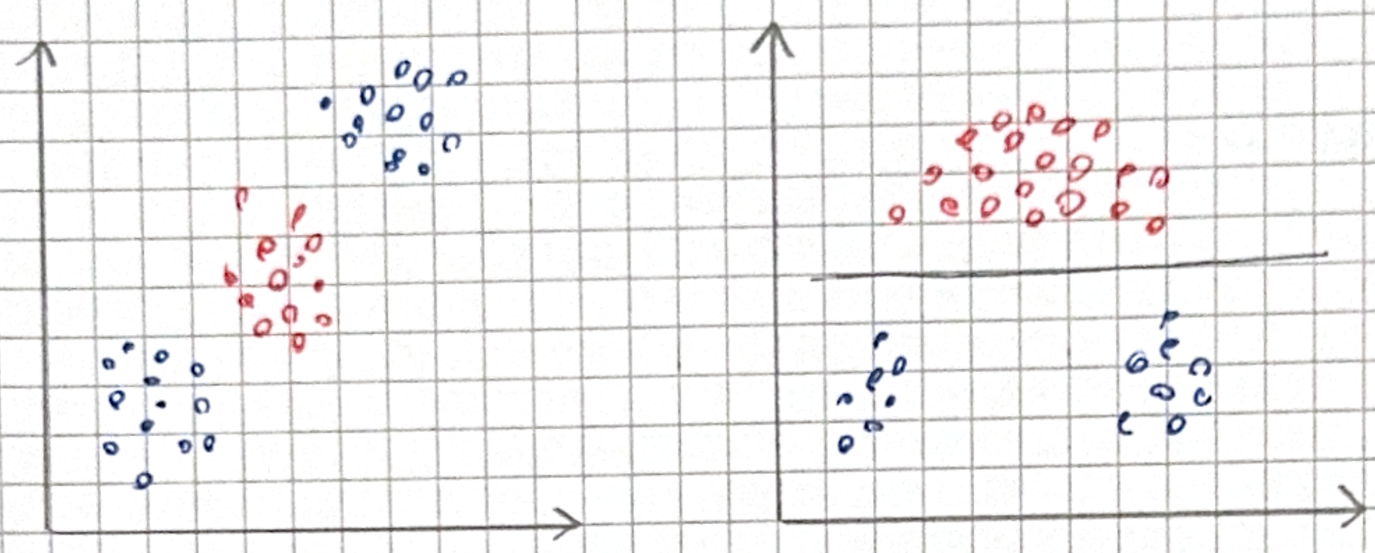
(2)

if we consider non linear SVM we can use kernel trick that mean instead normal inner product $x^T x$ with kernel $K(x, x')$
so we will have (consider polinomial kernel)

$$w_0^* = \frac{1}{|SV|} \sum_{x_k \in SV} \left(t_k - \sum_{x_j \in S} \alpha_j^* t_j K(x_i, x_j) \right)$$

$$y(x, w^*) = \text{sign}\left(w_0 + \sum_{n=1}^{N} a_n^* (\beta x^T x + \gamma)^d \right)$$

S 2



EX. 6

6.2

VOTING

given a dataset D

1 use D to train a set of models $y_m(x)$
for $m = 1, \ldots, M$

2. make predictions with

$$y_{VOTING}(x) = \sum_{m=1}^{M} w_m y_m(x)$$

$$y_{VOTING}(x) = \underset{C}{\text{argmax}} \sum_{m=1}^{M} w_m \mathbb{I}(y_m(x) = C)$$

$$\mathbb{I}(e) = \begin{cases} 1 & \text{if } e = \text{true} \\ 0 & \text{otherwise} \end{cases}$$

6.1

BAGGING

given a dataset D

~~1 use D to train a set of models~~

1. generate M bootstrap $D_1 \ldots D_M$
2. use each of ~~models~~ dataset to train model
3. make prediction with voting scheme

$$y_{BAGGING}(x) = \frac{1}{M} \sum_{m=1}^{M} y_m(x)$$

voting use all dataset bagging generate m bootstrap from dataset.