EXAM 01/07/2019

EXERCISE N°1

POINT 1

The supervised learning problem is the problem to learn a function $h : X \to Y$ where :

- X be the domain set, which is the set of all possible objects to make predictions about, where a domain point $\vec{x} \in X$ is called instance and is usually represented by a vector of features
- Y be the label set that defines the set of all possible labels
- H be the hypothesis class

The function $\hat{h}$ that we need to pick from H must be the one with the lowest generalization error i.e. $L_d(\hat{h}) = E_{z\sim D}[l(\hat{h}, z)]$ where :

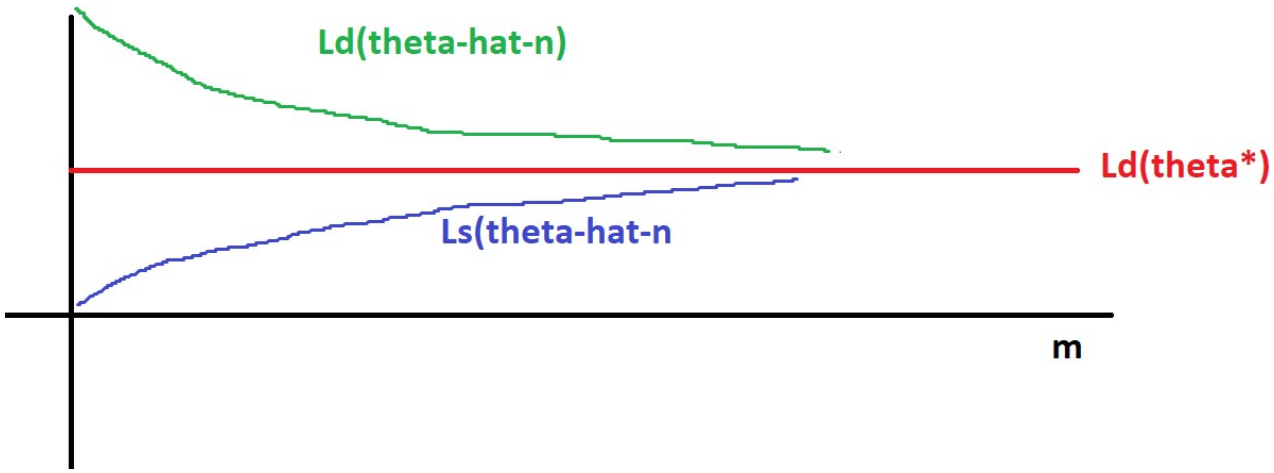
- D is the unknown probability distribution over Z from which $(x_i, y_i) \in S$ have been drawn (as independent samples).
- $l : HxZ \to R^+$ where $Z = XxY$ be the loss function namely a function that given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the

However because we do not know D ,under certain hypothesizes, a good estimate of $L_d(h)$ is given by the training error namely : $L_s(h) = \frac{1}{m}\sum_{i=1}^{m} l(h, (\overrightarrow{x_i} y_i))$ where $S = \left((\overrightarrow{x_1}, y_1) \dots (\overrightarrow{x_m}, y_m)\right)$ is the training set

POINT 2

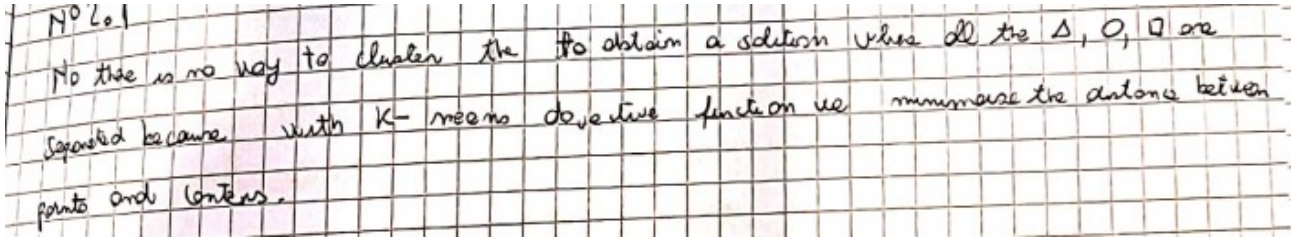
One way to pursue the objective explained in point 1, is ,as already introduce in point 1 as well, to use the so called Empirical Risk Minimization (ERM for short). So given a training set $S = \left((\overrightarrow{x_1}, y_1) \dots (\overrightarrow{x_m}, y_m)\right)$ with $\overrightarrow{x_i} \in X, y_i \in Y \; \forall i = 1, .., m$ to find $\hat{h}$ that has the lowest generalization error we can find the hypothesis that minimizes the training error namely : $L_s(h) = \frac{1}{m}\sum_{i=1}^{m} l(h, (\overrightarrow{x_i} y_i))$ where $l : HxZ \to R^+$ is the loss function .
Notice that this paradigm works only under certain hypothesizes regarding the hypothesis class H.

POINT 3

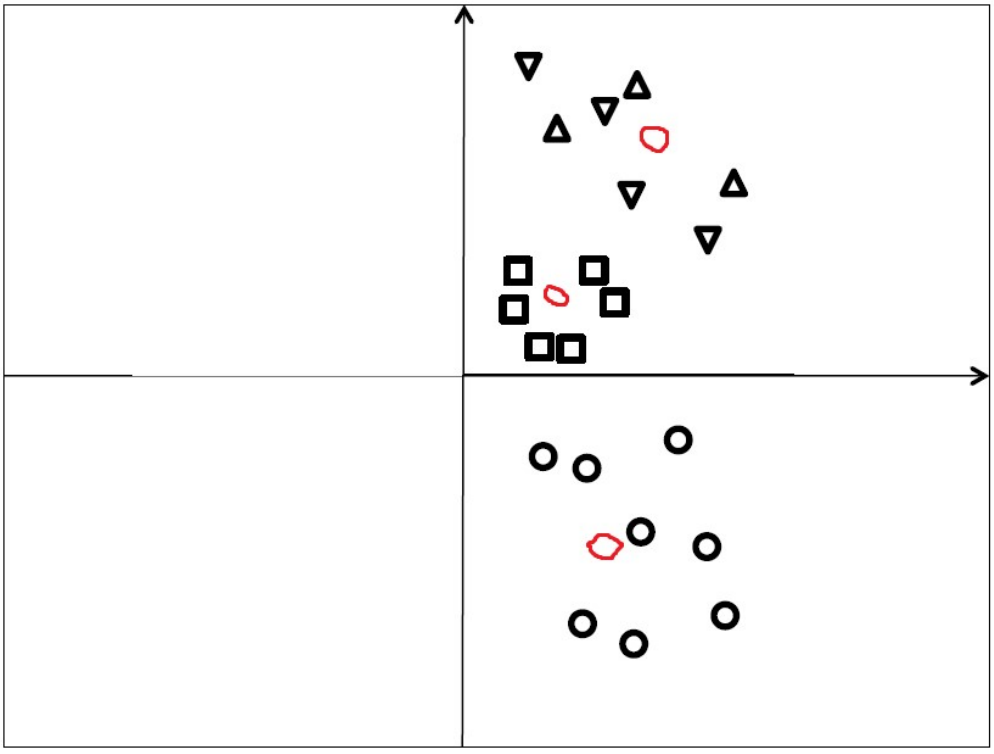


EXERCISE N°2

POINT 1


N°2.1
No there is no way to cluster the to obtain a solution where all the Δ, O, □ are separated because with K- means objective function we minimize the distance between points and centers.

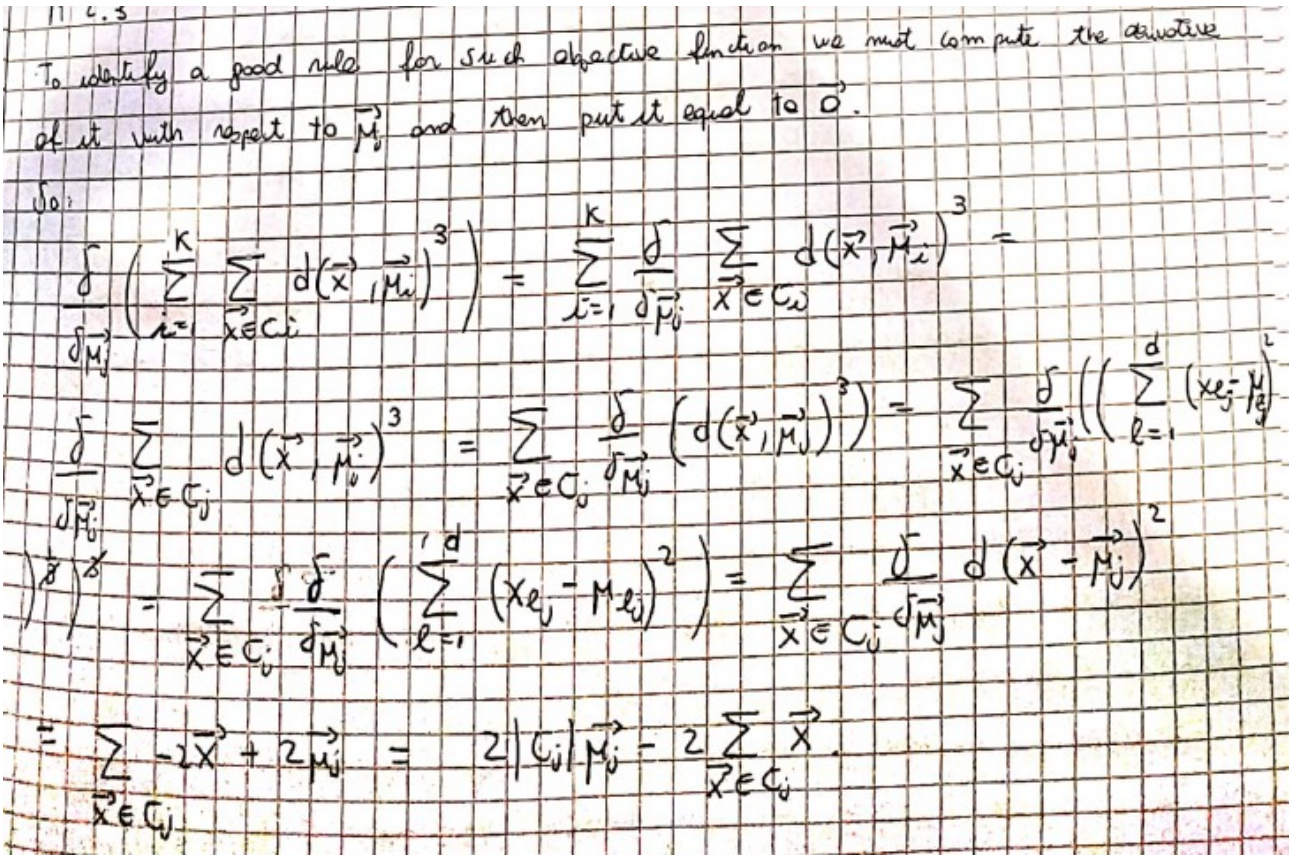
POINT 2

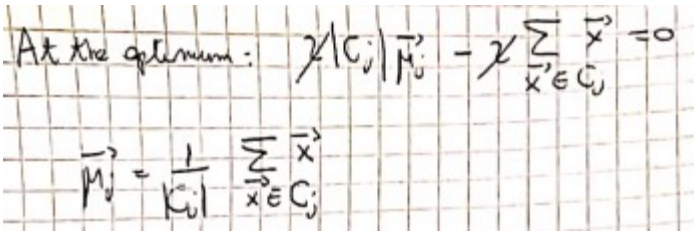


Rule to obtain such clustering :

$Let\ \vec{x} \in X\ then$:
$If\ (x_1 < 0\ and\ x_2 > 0) then\ \overrightarrow{x'} = [|x_1|, -|x_2|]$
$else\ if\ (x_1 < 0\ and\ x_2 < 0) then\ \overrightarrow{x'} = [|x_1|, |x_2|]$
$else\ then\ leave\ as\ it\ is$

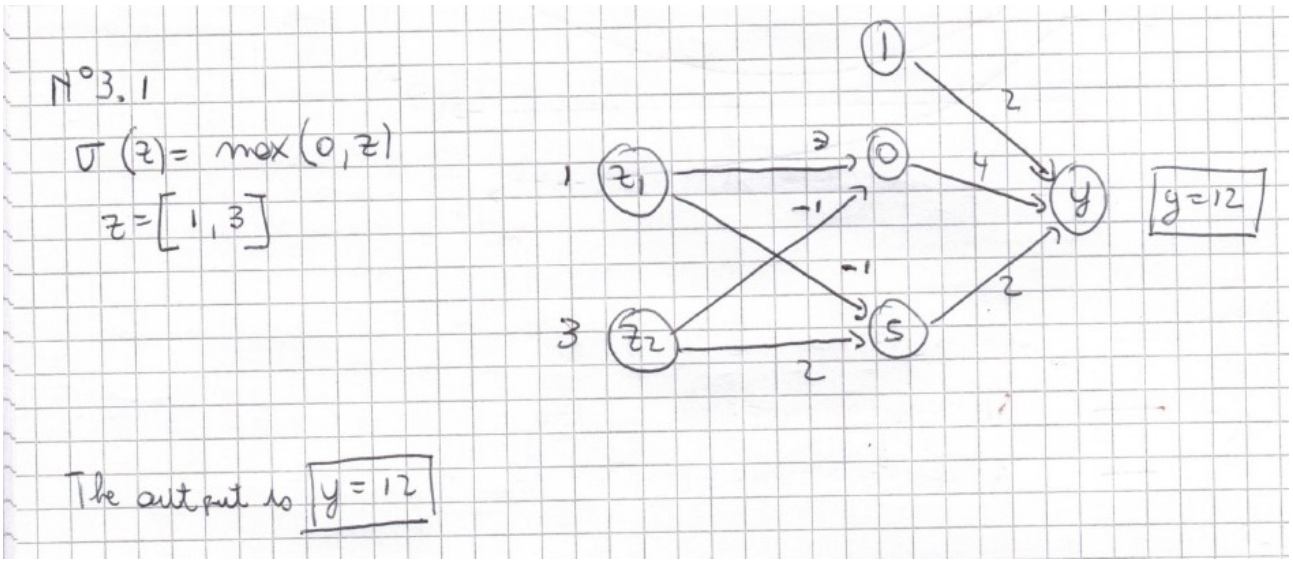
POINT 3





EXERCISE N°3

POINT 1



POINT 2

The back propagation algorithm is the algorithm to train a neural network based on stochastic gradient descent. The main structure of the algorithm is the following :

Input : Training data $(\overrightarrow{x_1}, y_1) \dots (\overrightarrow{x_m}, y_m)$ and a neural network with no weights

Output : a neural network with weights $w_{ij}^{(t)}\ \forall i, j, t$

And it works as follows :

- Randomly initialize the weights $w_{i,j}^{(t)}\ \forall i, j, t$
- Until convergence is reached then repeat the following steps:
    1. Pick a point $\overrightarrow{x_k}, y_k$ randomly from the training data
    2. Apply the forward propagation algorithm for such data and so compute $v_{t,j} \forall j, t$
    3. Compute the sensitivity vectors to make the updates of the weights namely : $\delta_j^{(t)} \forall j, t$
    4. Update the weights with the following rule : $w_{i,j}^{(t+1)} = w_{i,j}^{(t)} - \eta v_{t-1,j} \delta_j^{(t)} \forall i, j, t$
- If convergence is reached then return all the weights $w_{i,j}^{(t)}\ \forall i, j, t$

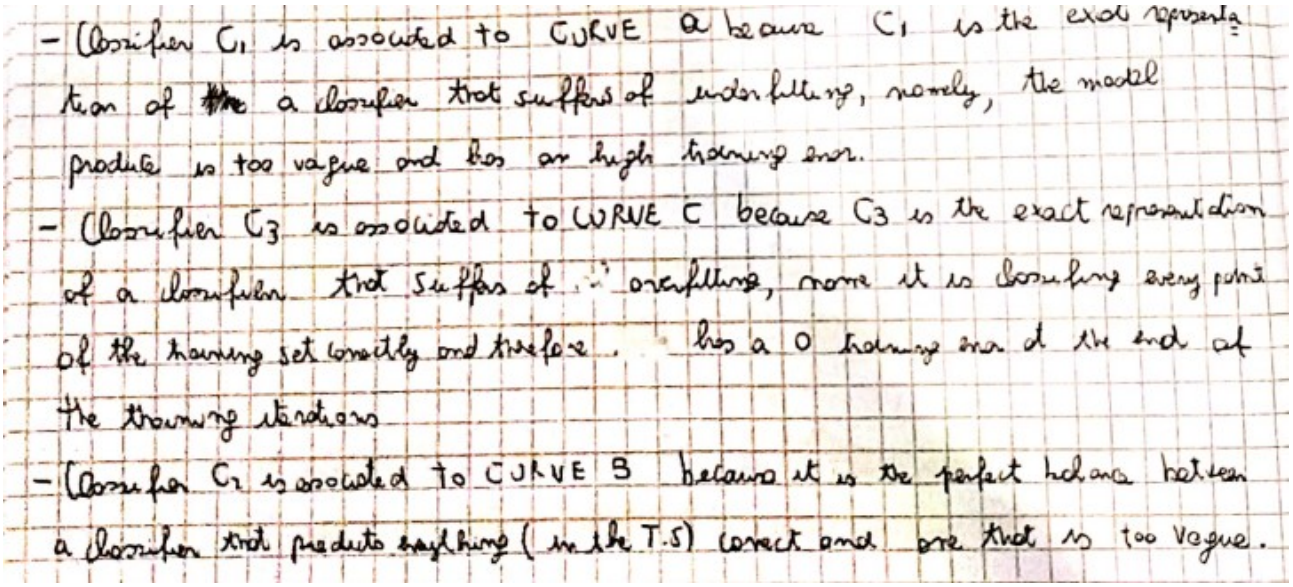
POINT 3

The main reasons why the ReLU function is used are the followings :

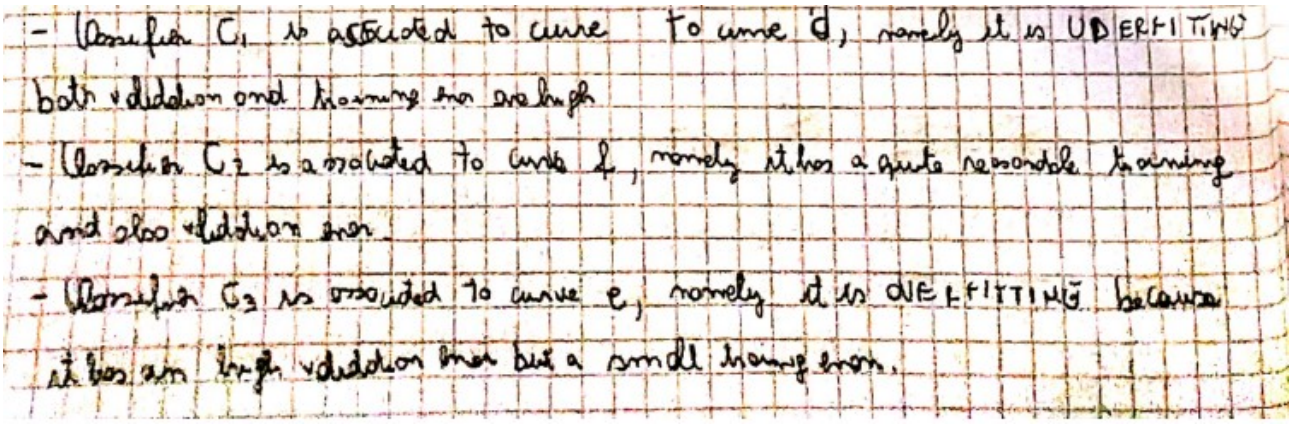
- It is simpler to compute
- It is as powerful as others
- It does not suffer of the vanishing gradient problem which is common problem in deep learning when using activation functions like the sigmoid or the hyperbolic tangent. Such problem consists of having a roughly 0 value as value of the delta terms to update the weights, on deep levels when applying the back propagation algorithm, and therefore such value will propagates to the shallow levels and so weights are not updated.

EXERCISE N°4

POINT 1



POINT 2



POINT 3

One strategy to reduce overfitting issue is to use cross-validation to choose the classifier C_i and then, once C_i is chosen we train C_i over all the data. Therefore recalling i, ϑ, we can apply the cross validation method.

To be more precise, cross validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have.
To do so, we split the dataset into k different folds of size m/k (this quantity is supposed to be integer). Then for each value of the parameter ϑ we find the best model for all the possible k-1 folds that we can select. In addition to that, we compute the average error of each parameter as the average of the errors on the folds left out, and when we have repeated such procedure for all the values of the parameters, then we select the optimal one by choosing the one that minimizes the error computed for each parameter. To conclude, we train our model on all the dataset with the parameter found.
The pseudo code is the following :

Input : $S = \left((\overrightarrow{x_1}, y_1) \dots (\overrightarrow{x_m}, y_m)\right)$; set of parameters θ; integer k; learning algorithm A

Split S into $S_1, \dots, S_k$

foreach ϑ ∈ θ
  for i = 1 ... k
    $h_{i,\vartheta} = A(S \backslash S_i; \vartheta)$
  $error(\vartheta) = \frac{1}{k}\sum_{i=1}^{k} L_{S_i}(h_{i,\vartheta})$
Outuput : $\vartheta^* = argmin_{\vartheta}(error(\vartheta))$
$h_{\vartheta^*} = A(S; \vartheta^*)$

Assuming that we have enough data another strategy is to use validation to choose C_i and then, once C_i is chosen, we use all the data to train C_i.

To be more precise validation consists in a way to select a value of a parameter ϑ by understanding what is the best value that such parameter can assume with the data we have.
To do so we split out dataset into 2 parts training set and validation set. Then, for each value of the parameter that we have, we find the best model for every possible values of the parameter on the training set. Then among such models that correspond to a value of the parameter we select the one that minimizes the validation error. The model obtain then is trained on the entire dataset.
Notice that training error is computed as : $L_s(h) = \frac{1}{m}\sum_{i=1}^{m} l(h, (\overrightarrow{x_i} y_i))$ where S is the training set, namely, $S = \left((\overrightarrow{x_1}, y_1) \dots (\overrightarrow{x_m}, y_m)\right)$, $l : HxZ \rightarrow R^+$ is the loss function where $Z = XxY$,that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.
The validation error instead is computed as : $L_V(h) = \frac{1}{mv}\sum_{i=1}^{mv} l(h, (\overrightarrow{x_i} y_i))$ where V is the validation set, namely, $V = \left((\overrightarrow{x_1}, y_1) \dots (\overrightarrow{x_{mv}}, y_{mv})\right)$, $l : HxZ \rightarrow R^+$ is the loss function where $Z = XxY$,that, given an hypothesis provides a measure of how much we lose by predicting the value $h(\vec{x})$ for $\vec{x}$ instead of the correct value y.