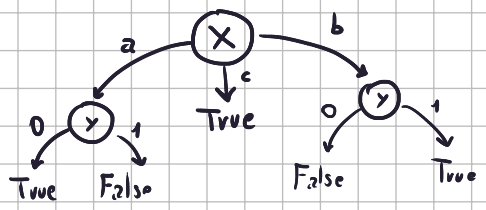
EXERCISE 1

Consider a classification problem $f : X \times Y \times Z \to \{T, F\}$, with $X = \{a, b, c\}, Y = \{0, 1\}, Z = \{u, v, w\}$ and data set D in the table on the right

X	Y	Z	f
a	1	v	F
b	1	w	T
c	0	u	T
b	0	w	F
a	0	w	T
c	1	u	T

1. Draw a decision tree consistent with the data set.
2. Write a new sample for this problem (not in the dataset) and classify this new sample according to the decision tree provided above.
3. Describe the problem of overfitting in decision trees and **formally** provide a possible solution: i.e., write the pseudo-code of a suitable algorithm to deal with overfitting in decision trees, explaining input, output and all the steps.



$\Rightarrow$ Tve : $(y=0 \wedge x=a) \vee (y=1 \wedge x=b) \vee x=c$

A new sample $(a, 1, v)$ is classified as False.

If the tree is too big, it may overfits (in the worst case we have 1 leaf for each sample). To overcome this, we can *prune* the tree:

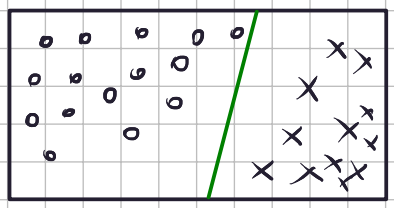
```
Pruning (T: tree, D: val. set) {
    do until termination condition {
        T' = T
        select random node n
        T': Substitute n with most common label in the subtree
        if (accuracy_D(T') > accuracy_D(T)) {
            T = T'
        }
    }
    return T
}
```

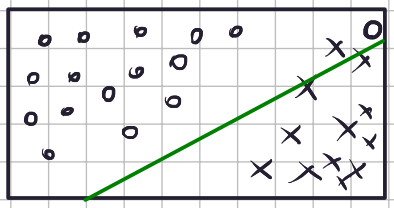
EXERCISE 2

1. Describe the Least squares method for linear classification. Formally define the error function that is used to solve the problem and the method used to compute a solution.
2. Draw a 2D data set for binary classification and qualitatively draw a possible solution that can be obtained by Least square.
3. Add to the above data set, some examples from one class coming from a different distribution (i.e., outliers far from the current points), in such a way that the Least square solution will be significantly different from the one shown in the previous point. Draw a possible solution for Least square in this new data set.

In the LSM we define $E(w) = \frac{1}{2} \sum_{i=1}^{N} \| t_i - \xi(x_i) \|^2 = \frac{1}{2} \text{trace}\{(T - WX)^T (T - WX)\}$.
(labels, parameters, design matrix)

A solution is given by $W = X^\dagger T$. ($\dagger$ pseudo inverse)





EXERCISE 3

Assume you are given a dataset D for regression of a function $f : \mathbb{R}^3 \to \mathbb{R}$.

1. Provide the mathematical formulation of a suitable model for the problem.
2. Describe a suitable solution method, illustrating an error function and a solution method.

We consider a Function $\phi : \mathbb{R}^3 \to \mathbb{R}^d$ and $d+1$ parameters $w \in \mathbb{R}^d$, $w_0 \in \mathbb{R}$ with the model : $\xi(x) = w^T \phi(x)$. We consider the error Function $E(w) = \frac{1}{2} \sum_{i=1}^{N} (t_i - \xi(x_i))^2$, we consider the gradient $\nabla E = \left(\frac{\partial E}{\partial w_j}\right)$ and apply : $w \leftarrow w - \gamma \nabla E$ until a satisfying solution is Found, where $\gamma \in \mathbb{R}^+$. We can add a reg. term to reduce overFilling : $E(w) = \frac{1}{2} \sum_{i=1}^{N} (t_i - \xi(x_i))^2 + \lambda \|w\|^2$, $\lambda \in \mathbb{R}^+$

EXERCISE 4

1. Give a short explanation of the *Kernel trick* (kernel substitution). What are the necessary conditions for applying the kernel trick?
2. Consider a linear model for regularized binary classification, i.e. $\mathbf{y}(\mathbf{x};\mathbf{w}) = \mathbf{w}^T\mathbf{x}$ with regularization $\|\mathbf{w}\|^2$. Provide an example of applying the kernel trick on this problem. In detail:
   - provide the mathematical formulation of the model in its original form (before applying the kernel trick);
   - explain why it is possible to apply the kernel trick and provide the "kernelized" formulation of the model.

   Note: If needed, consider the identity $(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda I)^{-1}$.

If in a machine learning algorithm, some vectors x appears only in form of dot product: $x_i^T \cdot x_j$, we can substitute that with $K(x_i, x_j)$, that is the kernel function that measures the similarity between two vectors.

In binary class. the error function is

$$E(w) = \sum_{i=1}^{N} (t_i - w^T x_i) + \lambda \|w\|.$$ With this, the solution is given by:

$$y(x) = \sum_{i=1}^{N} \alpha_i x_i^T \cdot x \quad \text{where} \quad \alpha = (\alpha_1 \cdots \alpha_N)^T \text{ is given by } (X^T X + \lambda I)^{-1} X^T.$$

we can apply the kernel trick and consider the model: $y(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x)$

EXERCISE 5

Consider the following Convolutional Neural Network (CNN) acting on images of dimension $96 \times 96 \times 3$:

1. What kind of problem and which dimension of the problem it is able to solve?
2. Compute the number of *trainable* parameters of the first block (coinv1,relu1,pool1) and of the block from the output of do1 to the output of do2.
3. Explain what is the role of the dropout layer in a CNN and in particular the meaning of the dropout rate.

conv1	Conv2D 3×3 kernel, 8 feature maps with padding 1 and stride 1
relu1	ReLU activation function
pool1	2×2 max pooling with stride 2
conv2	Conv2D 3×3 kernel and 5 feature maps with padding 1 and stride 1
relu2	ReLU activation function
conv3	Conv2D 3×3 kernel and 3 feature maps with padding 1 and stride 1
relu3	ReLU activation function
pool3	2×2 max pooling with stride 2
flatten	flatten operation
fc1	30 units
relu4	ReLU activation function
do1	dropout with rate 0.5
fc2	10 units
relu5	ReLU activation function
do2	dropout with rate 0.5
fc3	10 units
output	softmax

From the input and output layers we get that is a model that classifies RGB images of 96×96 px in 10 classes.

The number of parameter of layer 1 is: $\underset{\text{Features}}{8} \times (\underset{\text{kernels}}{3 \times 3 \times 3}) + \underset{\text{bias}}{8} = 224$

From do1 to do2 we pass from 30 units to 10 so there is a 10×30 matrix with 10 bias terms: 310 parameters.

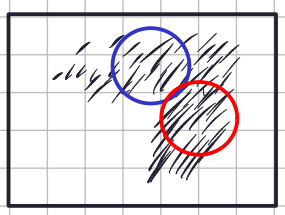
The dropout is a method to reduce overfitting and the rate gives the probability to ignore some blocks of the layer.

EXERCISE 6

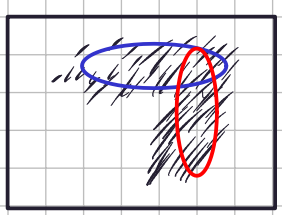
1. Define the unsupervised learning problem. Illustrate the concepts through a 2D data set ($|D| > 10$) and describe the concept of expected solutions for this problem.
2. Describe the difference between K-Means and Expectation-Maximization. Illustrate the differences on the above data set (or another one, if needed). In particular, show an example in which the solution provided by EM is *"better"* than the one provided by K-means.

in unsupervised learning we don't have the labels in the dataset so we try to assume that the samples are classified in K (decided by the experts of the domain) classes and we try to divide the samples in this classes.

Both KMeans and EM consider the GMM but Kmeans tries to estimate only the means, when EM also the covariance matrices and at priori probabilities.



Kmeans sol



EM sol