INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Introduction to Machine Learning — 2021/2022
Supervised Learning

> These exercises should be solved using Python notebooks (Jupyter) due to the ability to generate a report integrated with the code. It is assumed you are proficient with programming. All answers must be justified and the results discussed and compared to the appropriate baselines.
> Each exercise is scored 1 point. Max score of the assignment is 4 points. Optional exercises help achieving the max score by complementing the errors or mistakes in the mandatory exercises.
> **Deadline:** December 6th, 2021

In the following exercises the objective is to program algorithms that, given examples and an expected output learn to mimic the behavior present in the data.

Exercise 1

The "network" in Fig. 1 represents a perceptron with two inputs that can also be described by the following equations:

$$o = f(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2) \tag{1}$$

$$f(x) = \begin{cases} 1, & \text{if} \quad x > 0 \\ -1, & \text{if} \quad x \leq 0 \end{cases} \tag{2}$$

1. Choose one of the binary operations (AND or OR) and build a vector with all combination of two bit patterns: $\{\{-1,-1\},\{-1,1\},\{1,-1\},\{1,1\}\}$ and in another vector (called desired response, d) the corresponding result of the operation: OR $\{-1,1,1,1\}$ or AND $\{-1,-1,-1,1\}$. Notice that 0, 1 can also be used instead of -1, 1 with the due adaptations, as seen in the class.
2. Initialize w_0, w_1, and w_2 to small random values and, for each input pattern calculate the output (o).

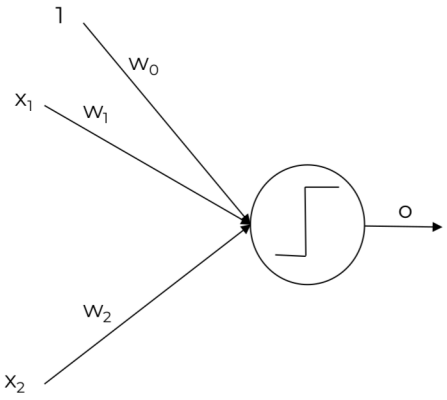

Figure 1: Perceptron

3. Calculate the difference / error (e) between o and the desired response (d) for each output.

4. Add to the update term for w_1 (Δw_1) and w_2 (Δw_2) according to

$$\Delta w_0 = \Delta w_0 + \alpha \cdot (d - o) \tag{3}$$
$$\Delta w_1 = \Delta w_1 + \alpha \cdot x_1 \cdot (d - o) \tag{4}$$
$$\Delta w_2 = \Delta w_2 + \alpha \cdot x_2 \cdot (d - o) \tag{5}$$

   where $\alpha = 10E - 4$.

5. After all examples are presented (an epoch), update w_1 and w_2 according to

$$w_0 = w_0 + \Delta w_0 \tag{6}$$
$$w_1 = w_1 + \Delta w_1 \tag{7}$$
$$w_2 = w_2 + \Delta w_2 \tag{8}$$

   so that in the next iteration the error will diminish.

a) How many epochs (iterations through the whole set) did it take to get all examples right? Repeat the experiment 30 times with different random values for the initial weights and present the average and standard deviation of the number of epochs it took to converge.

b) Try with different values for α (at least four) and measure the number of iterations to reach zero error. Repeat each experiment 30 times and compare in a graphic the average and standard deviation of the number of epochs it took to converge.

Exercise 2

Implement a k-NN classifier that is specifically suited for the dataset in (`https://archive.ics.uci.edu/ml/datasets/iris`).

Given a dataset containing labelled examples (a *a training set*) and a new example, the classifier should calculate the euclidean distance from the new example to all the elements of the "training set", choose the k closest elements of the "training set" and output this example classification as the class of the majority of the k closest "training set" elements (the k-Nearest Neighbors).

Split the dataset randomly in two subsets (70% / 30%). Use the bigger subset as the "training set" and the smaller as the *test set*, the new examples never seen by the classifier. Run all test examples through the classifier and calculate the number of correct predictions over the total number of examples of the test set. Compare the scores of k-NN classifiers for $k = 3$, 7, and 11. Repeat 10 times, with different dataset splits for each value of k and plot the results in a graphic to allow easy comparison. Why should k always be an odd number?

Exercise 3

Using the dataset from the previous exercise, split the dataset in two, according to the values of the first column (all examples where this column's value is larger than the column's average to one subset and all examples where the values are smaller to another subset). Use *Iris-setosa* as your target value ($p+$ are the examples classified as *Iris-setosa* and $p-$ the remaining ones) and calculate the entropy of the 3 datasets (the complete dataset, and the two subsets).

Remember that a set's (S) entropy is calculated by:

$$\text{entropy}(S) = -(p+) \times \log_2(p+) - (p-) \times \log_2(p-) \tag{9}$$

Calculate the gain of the split of S by feature a:

$$\text{gain}(S, a) = \text{entropy}(S) - \frac{\sum_v (|S_v| \times \text{entropy}(S_v))}{|S|} \tag{10}$$

where $|S|$ is the number of elements in S and S_v is each of the subsets of S when partitioned by the value of a.

What is the value of $\text{gain}(S, a)$? What does it mean in terms of your ability to classify the elements of S before and after the split?

Do the same for all features of your set. Which is the feature with greatest gain? How can you improve your chances of guessing a random examples' class using this information?

Explain how to build a decision tree with this information.

Exercise 4

Using the dataset from the previous exercises, implement a Naive Bayes classifier. As in exercise 2, split the dataset randomly in two subsets (70% / 30%). Use the bigger subset as the *training set* and the smaller as the *test set*. Run all test examples through the classifier and calculate the number of correct predictions over the total number of examples of the test set. Repeat 10 times, with different dataset splits and plot the results in a graphic to allow easy comparison. Comment the results. How does this classifier compares to the k-NN classifier?