Module-Lab 4: Probabilistic Machine Learning

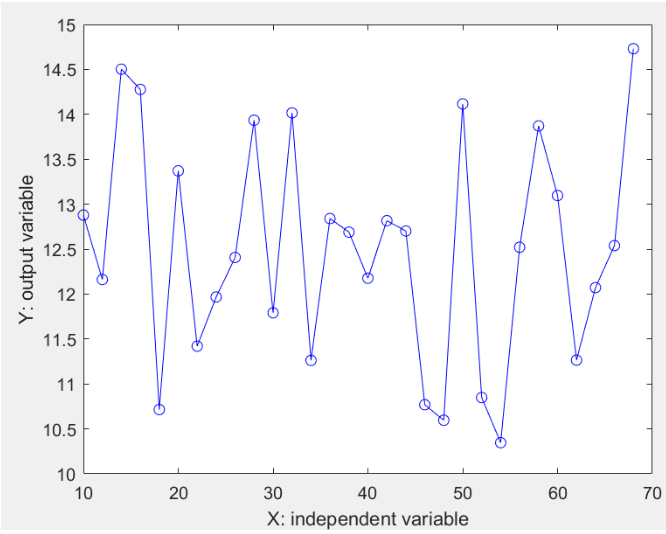
Exercise 1 This Exercise is part of the Lab/Practical session of the ACP ("Aprendizagem Computacional Probabilistica"). Please, access the materials through UCStudent, where you will find the folder **Lab4** containing the following files:
Lab4.pdf, DatasetRegressionLab4.txt, DatasetClassifiersLab4.txt

This exercise is about linear regression, in particular polynomial fitting, and we are given a data containing a training set and a test set. The former comprises 30 observed-points and in the test set we have 16 points. The input variable is denoted by $\mathbf{x}$ while the output variable (ie, the variable we wish to estimate) is denoted by $\mathbf{y}$. Let's consider the well-known polynomial function of the form

$$\mathrm{y}(\mathbf{x},\mathbf{w}) = \omega_0 + \omega_1 x + \omega_1 x^2 + \cdots + \omega_M x^M = \sum_{j=0}^{M} \omega_j x^j$$

The values of the parameters $\mathbf{w}$ (here, called coefficients) are determined by minimizing an error/cost function on the training data (ie, training points). To simplify the problem, the parameters have been already determined by using the training set and a minimization method (in MATLAB you can use the function `polyfit`)

a) Open/load the file DatasetRegressionLab4.txt and then generate a graph of the training points (eg, **plot(x,y)** in Matlab), as shown in the figure below



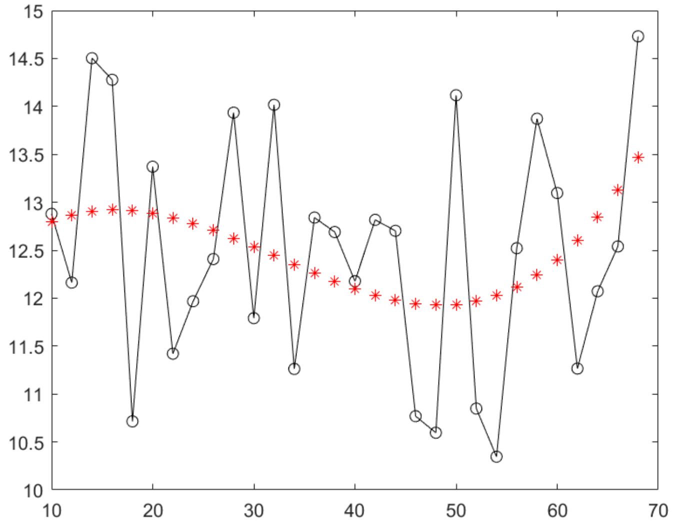
b) Calculate the corresponding estimated output $\mathrm{y}(\mathbf{x},\mathbf{w})$ on the TRAINING set for the following parameters (the values of M.order are increased staring by M=0,1,…,6)

For M=0, ω_0 = 12.49
For M=1, ω_1 = -0.0063835 $\quad\quad$ ω_0 = 12.739

and so on …
Notice that the values of $\mathbf{w}$ for increasing values of M are given in the file ***DatasetRegressionLab4.txt***

Hint:
For M=0, the first values of the estimated output are
12.49 12.49 12.49 12.49 12.49 12.49 12.49 …
For M=1, the first values of the estimated output are
12.675 12.662 12.65 12.637 12.624 12.611 12.599…
For M=2, the first values of the estimated output are
13.316 13.171 13.035 12.909 12.792 12.684 12.586…

c) In the same graph, plot the **training** points (ie, **x** vs **y**) and the *estimated.outputs* (ie, **x** vs $\mathrm{y}(\mathbf{x},\mathbf{w})$) determined by the seven models ie, for M=0,1,…,6. For example, for M=3 the figure shows the training points (in black) and the estimated output (in red).



d) Calculate, for all the seven models, the sum of the squares of the errors (SSE), given by

$$\mathrm{E}(\mathbf{w}) = \sum_{n=1}^{N} \{\, y(x_n,\mathbf{w}) - t_n\}^2$$

here, t_n correspond to the observed data on the **training set** and N=30.
For example, for M=1 the SSE is 44.775

e) Calculate for all the models (ie, for M=0,…,6), the Root-mean-square (RMS) error that is defined by

$$E_{RMS} = \sqrt{E(\boldsymbol{w})/N}$$

f) Finally, plot in the same graph 2 curves of the E_{RMS} as function of the order M. The first and the second curve should be, respectively
(i) the training RMS error
(ii) the testing RMS error (now, N=16)

Eventually, it might seem like a strange graph thus, we can plot **log**(RMS.test) instead to facilitate the visualization of both RMS errors in the same graph.

Exercise 2 This Exercise is related to the first Lab session. This time we will make use of the data set in the file ***DatasetClassifiersLab4.txt***

As in Lab1, each line in the data set corresponds to an examples/entity/object thus, the dataset contains 1400 examples. Students can use Matlab or Python to coding.

In this exercise the observed values comprise the outputs of a given supervised classifier (eg, a linear Discriminant); hereafter denoted by **Y1**. The labels/**ground-truth** is given by "0" (the negative class) or "1" (correspond to the positives). Therefore, we have to consider as **positives** all the examples/objects labelled by 1 (ie, digit = 1) while label = 0 represents the negatives.

a) To be able to make decisions, a classifier depends on a threshold. So, considering as thresholds the value **zero** (eg, thr = 0.0), develop a code to compute:

- The number of examples per class
- The number of true positives (TP) per class
- The number of true negatives (TN) per class
- The number of false positives (FP) per class
- The number of false positives (FN) per class
- The respective rates ie, TPrate, TNrate, FPrate, FNrate

b) By developing new piece of code or functions, calculate the following performance measures (a.k.a. performance metrics):

- Accuracy
- Balanced accuracy
- F1/F-score
- Precision
- Recall

Note: because this is a 2-class problem, it is common to assume that the classifier's prediction is 'positive' when the output is greater than a *threshold (thr)*. If **Y1** ≥ 0.0 than we say the classifier decision is a 'positive' otherwise the example is 'negative'.

c) Now, the goal is to develop code to compute some of the common functions in machine learning for the output variable (**Y1**), namely (we might need to see the lecture#3's slides)
(i) Logistic sigmoid
(ii) Softplus
(iii) ReLU

d) Finally, develop a code that might be useful in implementing a 4-fold cross-validation strategy. So, the goal is to implement a code that "split" the data set in 4 groups (with the same size) and then identify the indices of the examples/outputs (belonging to **Y1**) that would correspond to the 1st validation set, then the 2nd… and so on. For example:
- the first validation set can be the first 350 examples (indices=0,1,…,349) then, the second valid.set will be the examples denoted by the indices 350,351,…,699.
- On the other hand, the first training set will comprise the examples from 350 to 1400… and so on.