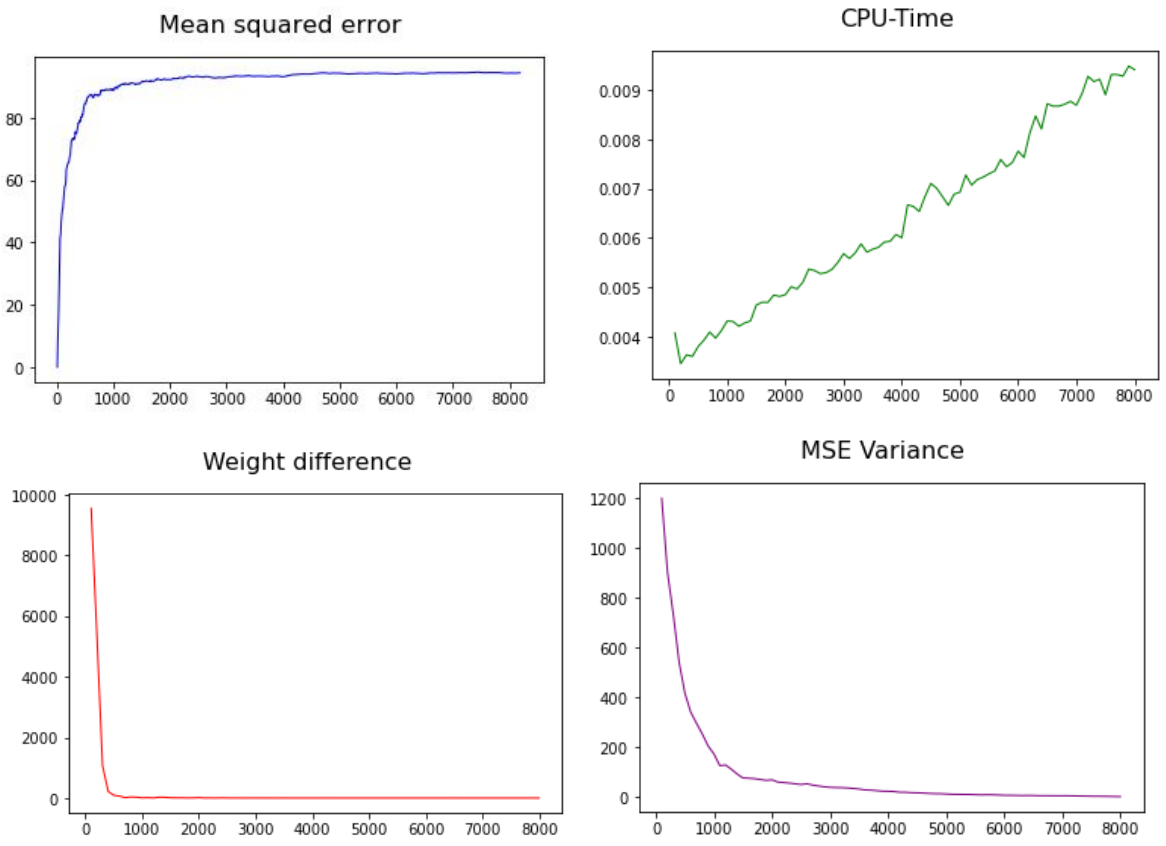
# Programming Exercise 1

Arnau Colom & David Moreno

## Regression Task




```
Dataset used: cpusmall_scale.txt
      Number of classes: 2
      Number of samples: 8192
      Number of data features: 12
      SPACE_SIZE = 100  #incremental step
```

***1. Plot the approximation error (mean accuracy) on the training set as a function of the number of samples N (i.e. data points in the training set).***

The left figure above represents the Mean Squared Error (smoothed) of the training set. Briefly explained, it graphically looks like a logarithmic function because, with very few samples, the *h* (weight vector) is really “adapted” or “personalized” for those training inputs. Nevertheless, the more data points we use to train the model (and to check it’s accuracy, MSE, etc), the more global the weight vector becomes (it covers more cases), so it fails in more cases than with fewer points because it cannot cover all the possible cases with so much precision, but can generalize better.

***2. Plot the cpu-time as a function of N.***

The CPU-Time graphic shows that, with more data points, the complexity increases linearly, as we are training and evaluating the model with a linear increasing number of data points. In fact, the step used to increment the data is 30, which we considered appropriate for the size of the data set used.

****3. Explain in detail the behaviour of both curves.***

This section is already explained inside the other exercises.

***4. Explore how the learned weights change as a function of N. Can you find an interpretation for the learned weights?***

Regarding the weight vector evolution, we have plotted the difference with respect to the computed coefficients in the previous permutation. As it can be observed (despite the noise), is that the difference tends to minimize along the number of samples. Firstly, it has to correct more the weight vector, but it approximates the best possible weight vector along N, so it is not necessary to make such big changes as with less data points. This could be correlated with the MSE figure in the following sense: when the MSE starts converging, or in other words, it stabilizes, is approximately when the weights also start to converge / stabilize.

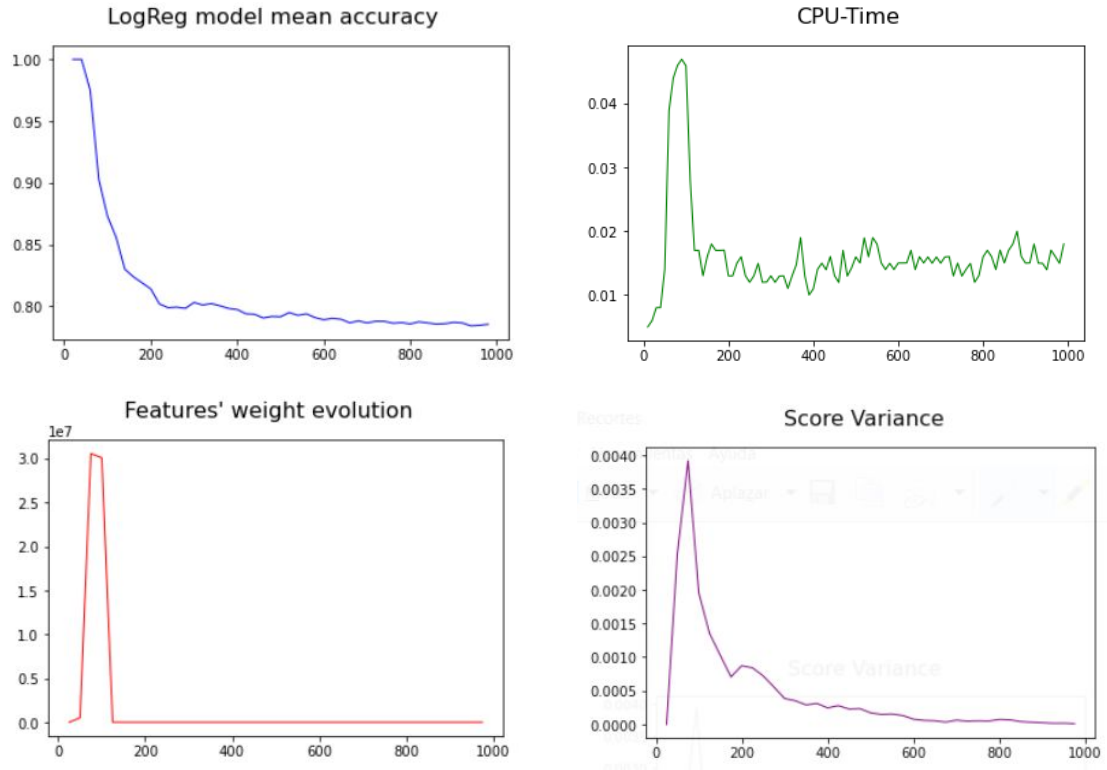
The variance has a quite similar pattern (regarding the time where the strong changes occur) than the MSE explained in the previous paragraphs. We can observe that with few data points, the variance is quite big, but once the model starts to generalize and to cover more cases, it slowly decreases along N.

$$Variance = E_{S\sim D,f}\{(h_S(x) - \overline{h}(x))^2\} \text{ where } \overline{h}(x) = E_{S\sim D,f}\{h_S(x)\}$$

To obtain smoother curves, and also to compute the variance (as the formula claims) we have computed several permutations/experiments of the linear regression simulation, shuffling the data to obtain averaged values for all the parameters.

Arnau Colom - U137471
David Moreno - U114148

# Classification Task




```
Dataset used: german_number_scale.txt
      Number of classes: 2
      Number of samples: 1000
      Number of data features: 24
      SPACE_SIZE = 25
```

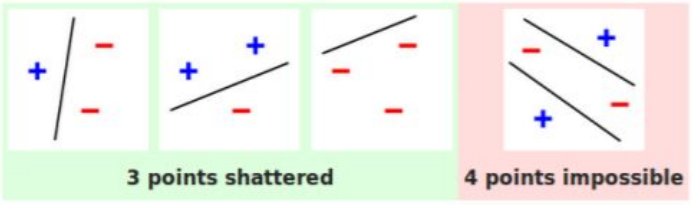
***1. Plot the approximation error (mean accuracy) on the training set as a function of the number of samples N (i.e. data points in the training set).***

The first figure represents the accuracy of the training set. The accuracy is represented as the mean accuracy, that is **a** = $\frac{1}{N} * c$, where **c** is the number of correct predictions.

As we can see the first samples always have a 100% accuracy when we predict them with the models trained with them. We can explain this through the VC-Dimension properties for binary classification using linear(affine) models, in this case logistic regression.

In binary classification the VC-Dimension can be described as the maximum number of points that can be classified correctly. Put, for instance, the case of a binary classification in a 2-D plane. The maximum number of points that can be perfectly separables are three. If we add a 4th point we might not be able to separe it correctly:

Arnau Colom - U137471
David Moreno - U114148




In the case of 3-D we are able to separe 4 points with the use of a plane, but not necessarily five. So we can see that the VC-dim of a linear model used for binary classification is **d** (number of dimensions) plus one, VC = **d** + 1.

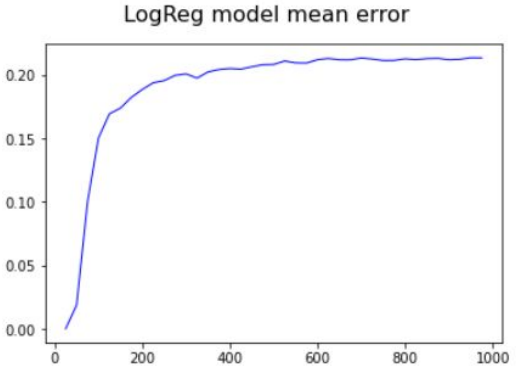
$$F = \{f : f(x) = \sigma(w^T x), w \varepsilon \Re^d\} \rightarrow \text{the } VC\text{-}dim(F) = d+1. \text{ (Eq 1)}$$

*Where if the probability is greater than 0.5 we assume that refers to one class and if it's smaller than 0.5 we assume that is the other class (similarly as perceptron)*

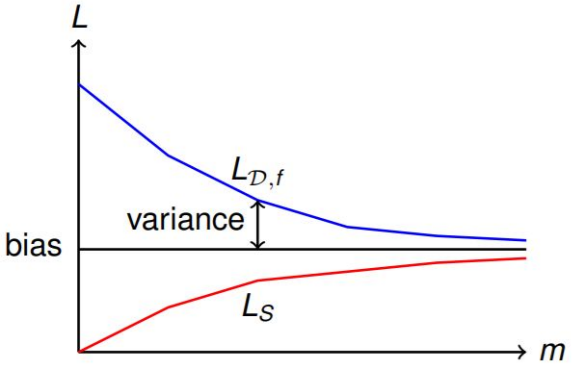
And as we can see from Eq 1, the VC dimension is defined by the dimensionality of the weights **d**, that is the number of features of the dataset.

For this reason **always**, if the sample has been trained with **d+1** samples, it will correctly classify the same points which have been trained.

We have talked about the accuracy, but the error of the model is 1-a. This is represented by figure:




That, as we saw in the theory class, is the training loss, converging into the bias and, getting closer to the real loss.



***2. Plot the cpu-time as a function of N.***

The explanation of the CPU-time can be described in three stages:

1. The model is capable of learning the points that feed the model
2. The part where the model is not able to classify correctly and need to star to generalize
3. The model has been generalized.

In the first stage, the algorithm learns easily since it only needs to classify a small number of clusters closer to the VC-dimension , so the algorithm converges faster, hence less computation time is required. In the following state, the model is fed with more data and is forced to generalize hence needs to spend more time learning how to generalize. In the last step the algorithm is approximately well generalized and it will try to polish its position. Our belief is that it needs to spend some time to converge, but as it needs less steps to generalize better it needs less time to converge than the second step.

***3. Explain in detail the behaviour of both curves.***

We can try to correlate both curves in the following way. We can see that it takes less time learning when the accuracy is always 1, that is when it can classify perfectly because of the small number of samples.

The accuracy starts to decay when the model starts to see new data that can not be classified using linear hypothesis and need to change the model so that it's more generalized. That is to minimize the error even if some points are classified incorrectly. During this part is when the accuracy decays faster and the computational time is greater.

The last step is when the accuracy decays slower, and is where the model will change very little. That is because the hypothesis is more generalized, and it keeps trying to reduce the error, but as it is really closer, the changes are very subtle.

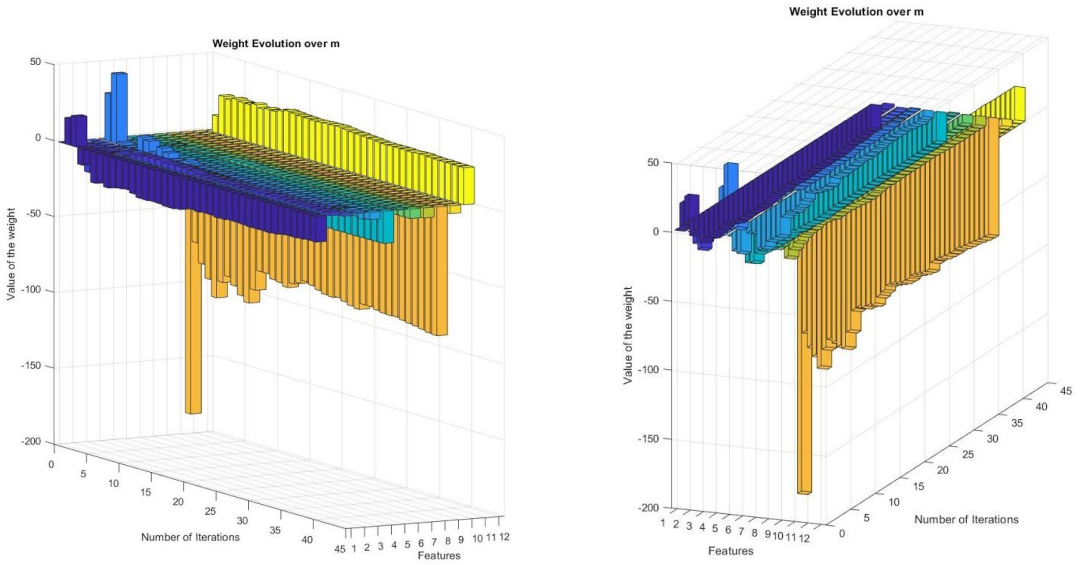
***4. Explore how the learned weights change as a function of N. Can you find an interpretation for the learned weights?***

For this explanation we will use the same third stages presented above: In the first stage the weights will almost not change since the model can separate the data perfectly and do not need to make big changes. In the following it encounters the new data that can not classify and need to change more abruptly to find the more optimum hipotesis given the maximum number of iterations. From there we jump to the last stage, where it will try to minimize the error as well as possible.

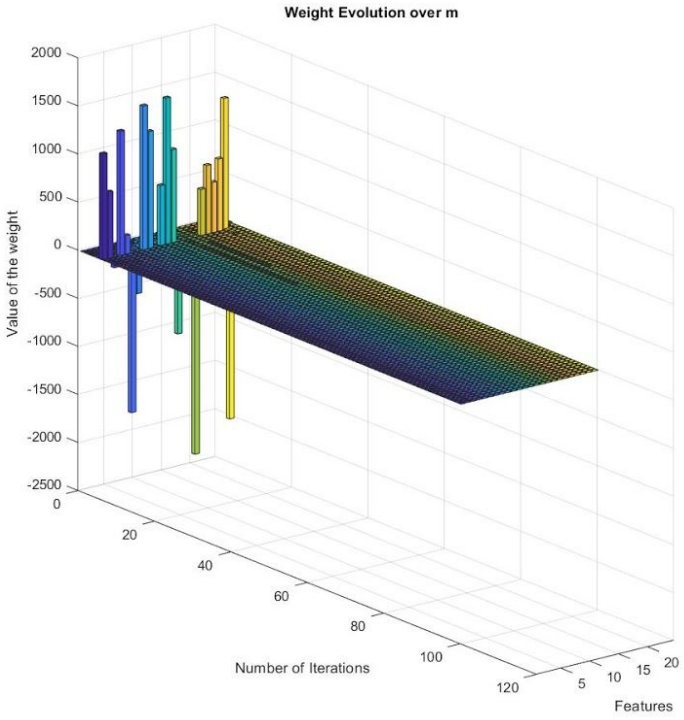
The same way we have done in linear regression, we smooth the curves by computing several permutations of the logistic regression  experiment. In this case we shuffle the data 30 times.

Regarding the **variance**, the behaviour is similar to the one explained for linear regression.

# ANEX




*Evolution of the weights for linear regression for one experiment. On the left we have a top view of the values and on the right figure we have the bottom view.*




*Evolution of the weights for one experiment for the logistic regression.*