Final report on the ICT for Health laboratories

Luca Gioacchini
s257076

A.A.2018/19

Contents

1 Parkinson's Disease

Parkinson's disease is the most common chronic neurodegenerative disorder which affects about 1.2 million people in Europe alone. The symptoms are mainly motor ones, such as tremor, rigidity, akinesia and difficulties with speech. They are due to the degeneration of the *dopamine*[1] synthesizers in the *substantia nigra*[2].
Since there is no cure, the only successful treatment is the monitoring of the patients to decide which drugs may be beneficial. The status of the disease is monitored by using a scale called *Unified Parkinson's Disease Rating Scale* (UPDRS). It is a score system applicable to a clinical observation of the patients. Because of the slowness of the evaluation process and of its subjectivity, it would be useful to determine the UPDRS by analysing other features, such as vocal parameters.

The purpose of the laboratory is to estimate the total UPDRS by applying different linear regression techniques (Linear Least Squares, or LLS, Conjugate Gradient, Gradient Descent, Stochastic Gradient Descent, or SGD, Steepest Descent and Ridge Regression) on a provided dataset.

The available data are 5875 biomedical voice records from 42 patients affected by early-stage Parkinson's disease. Each record contains information about the patients (age, sex, etc.) and 16 voice measures (5 measures of fundamental frequency variation, or Jitter; 5 measures of amplitude variation, or Shimmer; motor and total UPDRS score and other voice parameters[3]).
The dataset is sliced into three subsets: the *train* one, composed by the 50% of the total amount of data, the *validation* one, composed by the first 25% of the remaining data, and the *test* one, composed by the last 25% of data. Before the application of different linear regression algorithms, the data has been normalized with the mean value and the variance of the train dataset (see Appendix A), the first four columns containing patients' information have been removed and a random shuffling has been applied.

The linear regression model is:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\nu} \tag{1}$$

where $\boldsymbol{\nu}$ is the vector of the measurement errors, $\mathbf{X}$ is the matrix of the *regressors*, or the independent variables, and $\mathbf{y}$ is the vector of the *regressands*, or the dependent variables. In order to estimate the total UPDRS, it is necessary to consider it as the regressand, while the normalized train dataset without the total UPDRS feature is the regressor.
By applying the algorithms it is possible to estimate the optimum weight vector $\mathbf{w}$, which can be used to estimate the regressand $\hat{\mathbf{y}}$ of a new dataset $\hat{\mathbf{X}}$. In this case, the new datasets are the validation one and the test one.

[1] Dopamine is a neurotransmitter responsible of memory, attention, movement, learning, etc.

[2] Is a region of the brain, part of the basal ganglia, responsible of reward and movement

[3] Recurrence Period Density Entropy, Detrended Fluctuation Analysis, Harmonic to Noise Ratio and a nonlinear measure of fundamental frequency variation

2 Linear Least Squares Pseudoinverse (LLS)

The main assumption behind the Linear Least Squares (LLS) algorithm is that the measurement error $\boldsymbol{\nu}$ of the Equation (1) is small. In this way it is possible to express the square error as a function of $\mathbf{w}$:

$$f(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 \tag{2}$$

The aim of the LLS is to determine the optimum vector $\mathbf{w}$ which minimizes the cost function $f(\mathbf{w})$. This is done by setting the gradient $\nabla f(\mathbf{w}) = 0$, so by calculating

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3}$$

where $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is the pseudoinverse of the matrix $\mathbf{X}$.

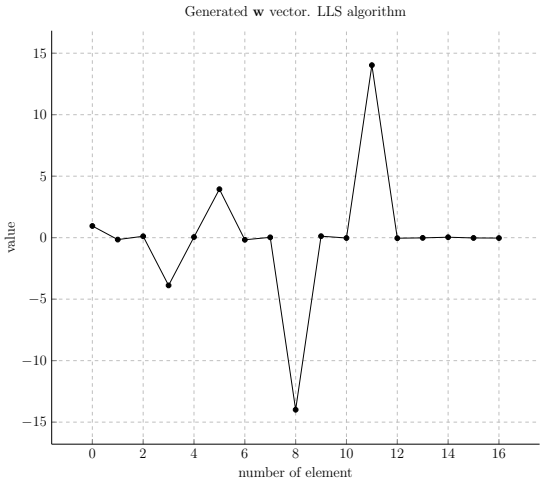
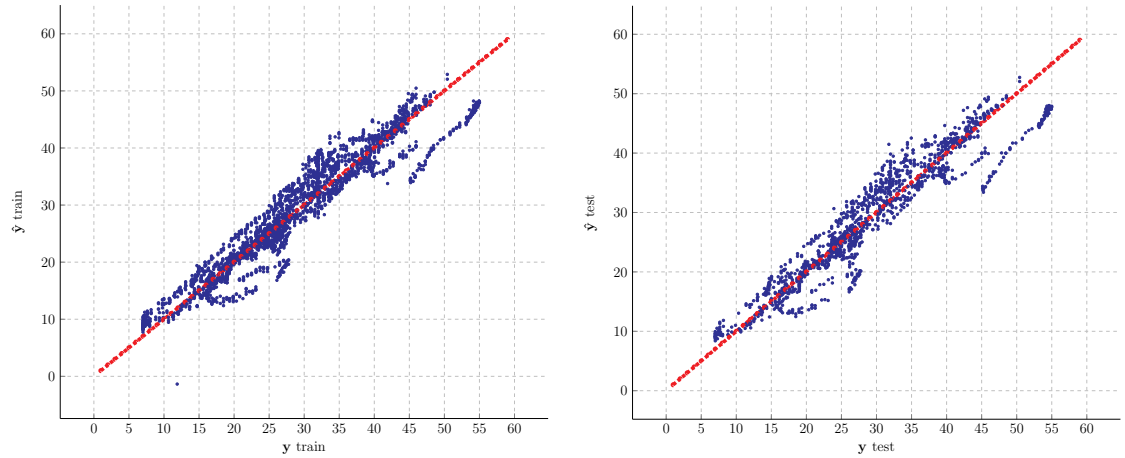


Figure 1: Optimum vector $\mathbf{w}$ generated with the LLS algorithm

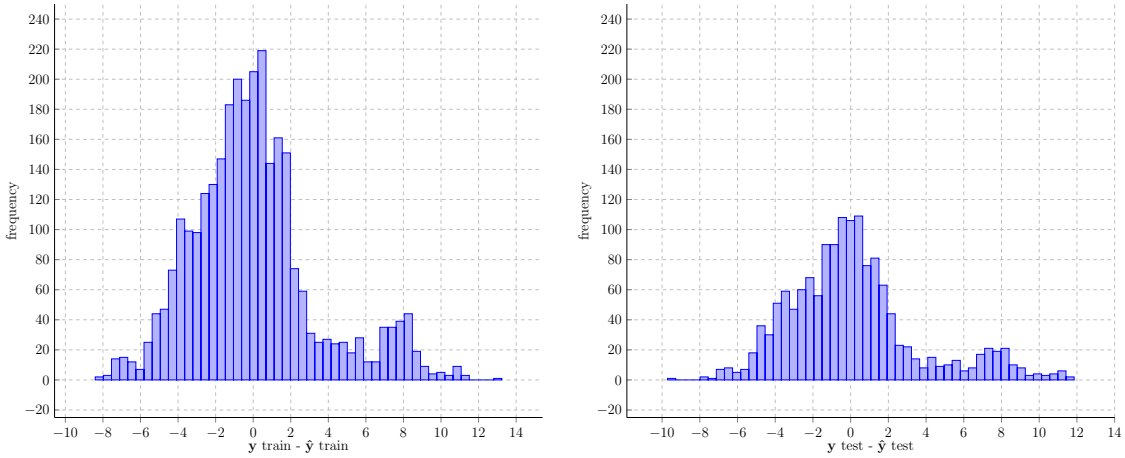


(a) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Train dataset (b) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Test dataset

Figure 2: Plots obtained by applying the LLS algorithm

Figure 1 shows the optimum $\mathbf{w}$ generated with the LLS algorithm. The dataset used is the train set described in Section 1. Once this has been done, it is possible to estimate the total UPDRS (vector $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$) for all the other datasets.

Figure 2 shows the comparison between the estimated $\hat{\mathbf{y}}$ vector, whose coordinates are plotted on the y axis, and the original $\mathbf{y}$ vector, plotted on the x axis. In both of the plots the red dotted line is the bisector of the first quadrant. If the coordinates of the two vectors perfectly matched, the dots would be aligned with the bisector.



(a) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Train dataset (b) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Test dataset

Figure 3: Histograms obtained by applying the LLS algorithm.

Figure 3 shows the histograms of the difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. By comparing Figure 3(a), in which the un-normalized train dataset has been used, and Figure 3(b), in which the un-normalized test dataset has been used, it is clear that in the first case the difference between the two vectors is greater than in the second. This happens because the size of the test dataset is half of the train one, so, even if $\mathbf{w}$ has been generated by linear regression applied to the train dataset, the model better fits smaller amount of data.

3 Conjugate Gradient

The Conjugate Gradient is an iterative algorithm which allows to solve a system of linear equations defined as:

$$\mathbf{Q}\mathbf{w} = \mathbf{b} \tag{4}$$

where $\mathbf{Q}$ is a symmetric and positive-definite matrix. $\mathbf{w}$ and $\mathbf{b}$ are two vectors.

By considering Equation (2), it is possible to write down the equation which minimizes $\nabla f(\mathbf{w})$ in the form of Equation (4) thanks to mathematical substitutions[4]. Thus, solving Equation (4) means find the optimum weight vector $\mathbf{w}$ which minimizes the cost function described in Section 2.

The idea behind the algorithm is to determine $\mathbf{w}$ by finding a sequence of N directions that are conjugate with respect to $\mathbf{Q}$ (or $\mathbf{Q}$-orthogonal[5]), where N is the dimension of $\mathbf{w}$

[4] $\mathbf{Q} = \mathbf{X}^T\mathbf{X}$ and $\mathbf{b}=\mathbf{X}^T\mathbf{y}$

[5] two vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ are conjugate with respect to $\mathbf{Q}$ if $\mathbf{d_1}^T\mathbf{Q}\mathbf{d_2}=0$.

$(\in \mathbb{R}^N)$.

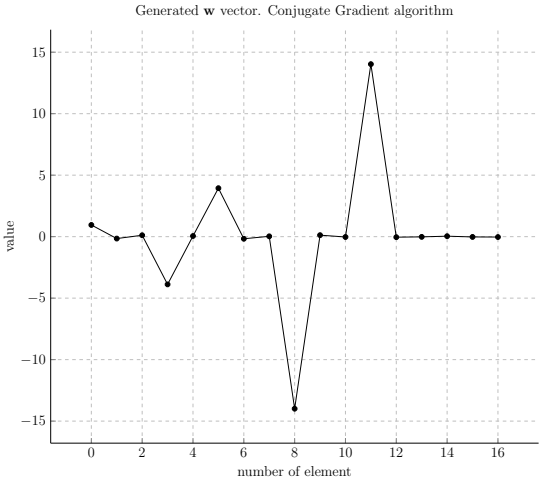
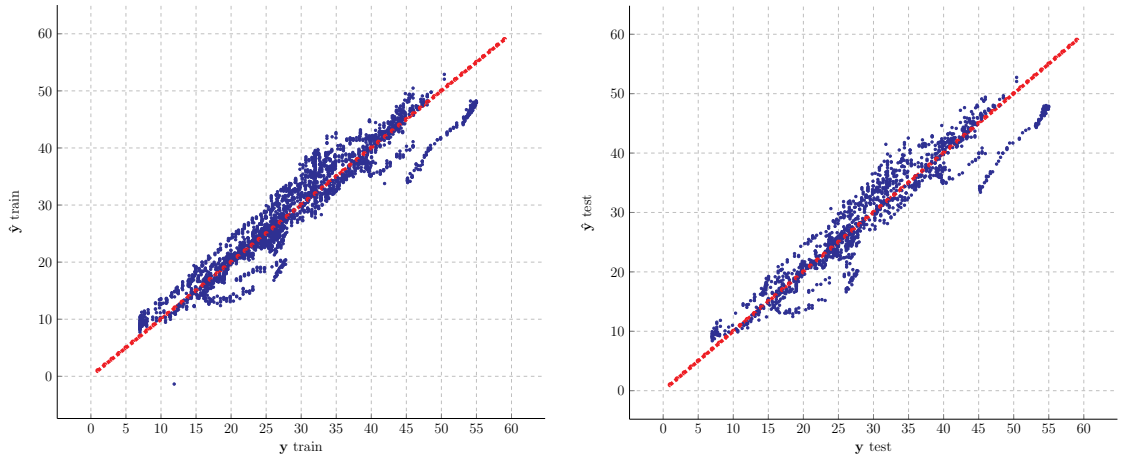


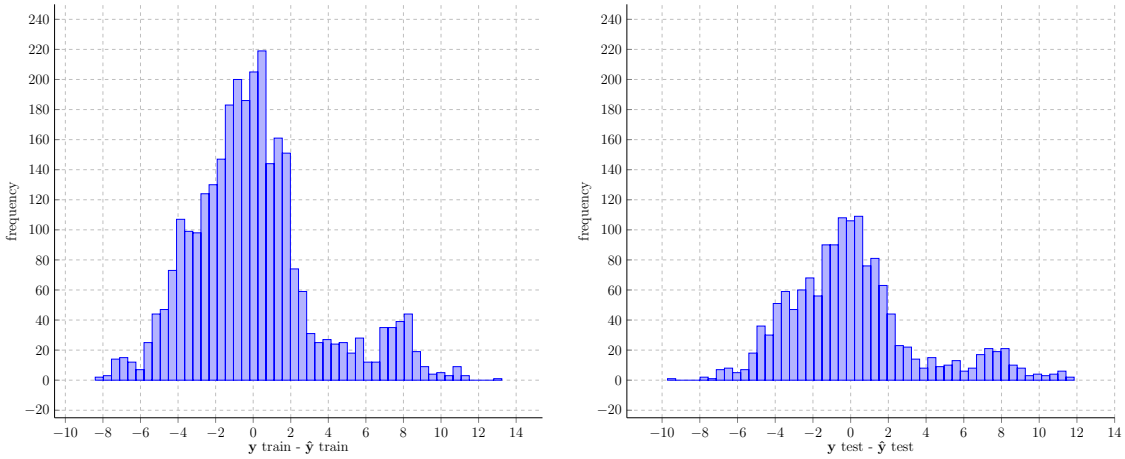
Figure 4: Optimum vector **w** generated with the Conjugate Gradient algorithm



(a) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Train dataset (b) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Test dataset

Figure 5: Plots obtained by applying the Conjugate Gradient algorithm.

Figure 4 shows the optimum **w** generated with the Conjugate Gradient algorithm. By comparing it with Figure 1 it is clear that the results are the same. According to this, all the results shown in Figure 5 and Figure 6 are the same discussed in Section 2. The main benefit of this method compared to the LLS one is that sometimes calculating the pseudoinverse matrix may be too expensive in terms of computational power, so, performing simple math for a certain number of iterations may be the best solution.



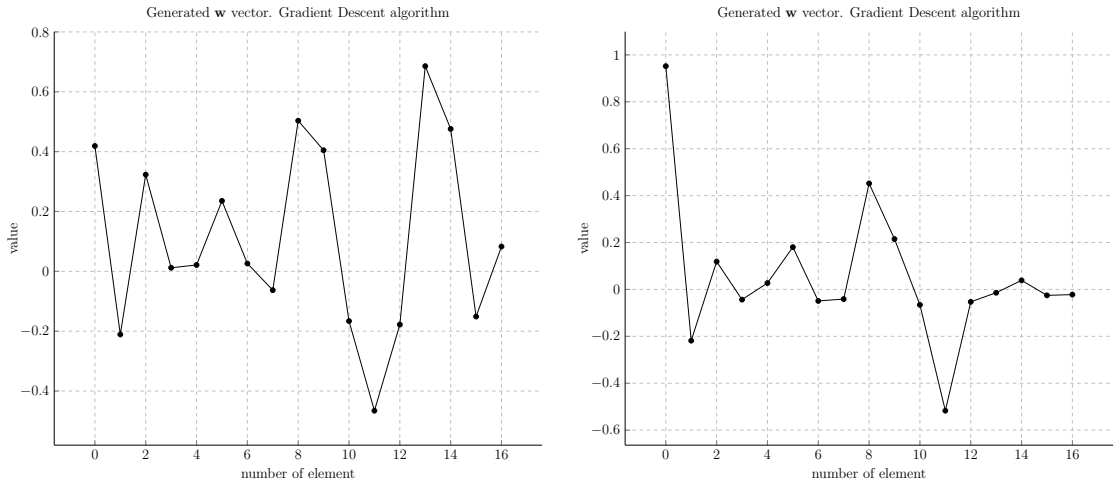
(a) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Train dataset (b) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Test dataset

Figure 6: Histograms obtained by applying the Conjugate Gradient algorithm.

4 Gradient Descent

Gradient Descent algorithm is an iterative optimization algorithm which allows to find the minimum points of a function.

By considering Equation (2), it is known that $\nabla f(\mathbf{w})$ indicates the direction of the maximum points of the function. Thus, after having evaluated $\nabla f(\mathbf{w}_i)$ corresponding to a certain $\mathbf{w}_i$ and after having fixed a constant γ called *learning coefficient*, a new $\mathbf{w}$ is determined by moving in the opposite direction of $\nabla f(\mathbf{w}_i)$ multiplied by γ. This is repeated for a certain number of iterations.



(a) $\gamma = 10^{-6}$, 100 iterations (b) $\gamma = 10^{-6}$, 10000 iterations

Figure 7: Vector $\mathbf{w}$ generated with the Gradient Descent algorithm by setting different number of iterations

In contrast to the LLS method and the Conjugate Gradient one, which can generate the same $\mathbf{w}$, this time the result depends on the number of iterations and on the value of γ. According to Figure 7, it is clear that if the loop is interrupted after 100 iterations,

the generated vector is quite different from the optimum one, while after 10000 iterations it tends to be more similar to the Section 2 and Section 3 ones.
With regard to γ, if it is chosen very small, the decrease of the gradient is very slow and a greater number of iterations is required. On the other hand, by choosing a big learning coefficient, it may occur a wide oscillation around the optimum point and the algorithm can't reach it.

In order to analyze the algorithm behavior, $\gamma = 10^{-6}$ has been chosen. In this way, the oscillating trend should be avoided and a reasonable computational power should be requested.

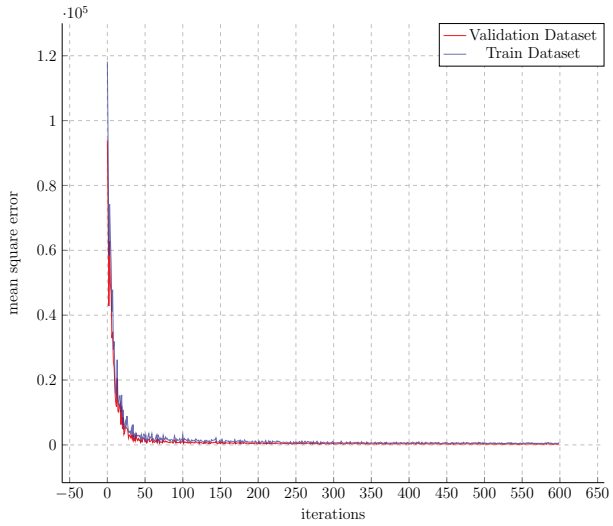
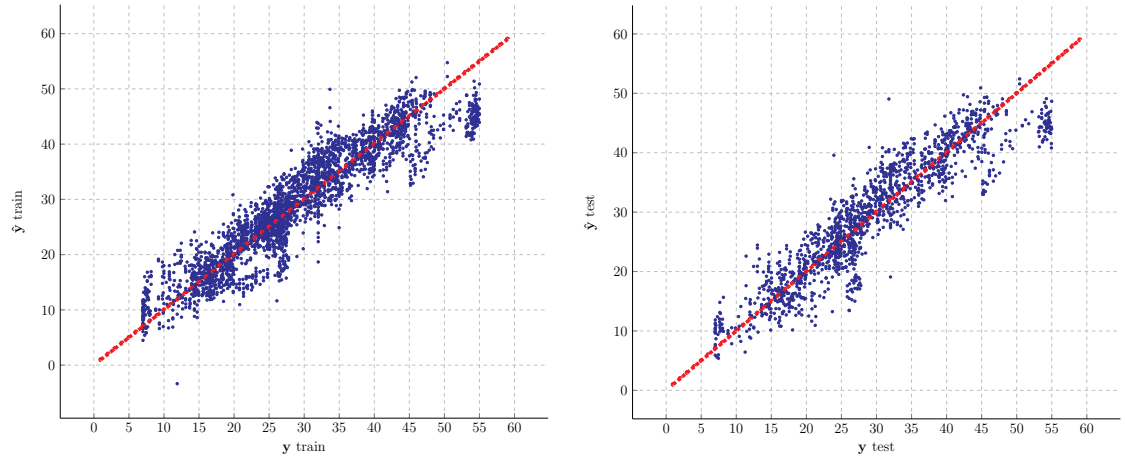


Figure 8: Comparison between the validation dataset mean square error (normalized) and the train dataset one.

Regarding the loop stopping condition, a very powerful method in case of overfitting is to study the mean square error plots. If overfitting occurs, the mean square error has a descending trend until a certain number of iterations is reached. After that, the generated $\mathbf{w}$ vector provides a model too suited to the training dataset, so the prediction when it is applied to the validation one fails. This behavior leads to a diverging trend. Thus, the stopping condition is set as the number of iterations which ensures the minimum mean square error when the validation dataset is applied.
Figure 8 shows an attempt of this strategy, but, since the validation set consists of 50% of the train dataset, the overfitting does not occurs and both plots has the same trend.

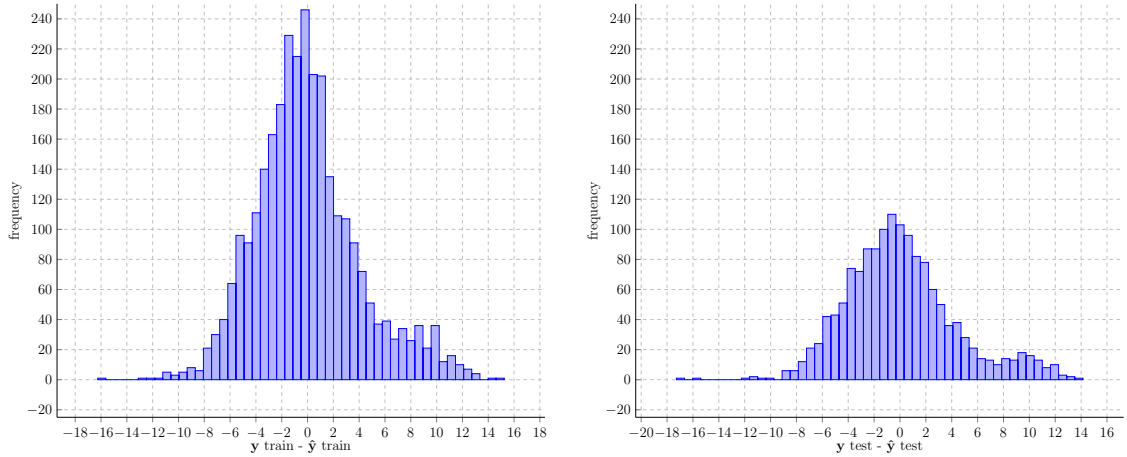
According to what has been said, the stopping condition has been set to 600 iterations which is something in-between 100 and 10000 (used in Figure 7). Moreover, Figure 8 ensures that by using such a parameter, the mean square error should tend to 0.

Even by applying the Gradient Descent algorithm with a learning coefficient $\gamma = 10^{-6}$ and 600 iterations, when the train dataset is used, the performance is better than the one with the test dataset. Moreover, by comparing Figure 10 and Figures 3 and 6 it is clear that, when the Gradient Descent is applied, the difference between the two vectors $\hat{\mathbf{y}}$ and $\mathbf{y}$ has a 0 with a higher frequency (more than 240 for train dataset) than the other algorithms (about 200 for train dataset when the LLS and Conjugate Gradient are applied). On the other hand, the variance of the histograms trend is larger in Figure 10.



(a) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Train dataset (b) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Test dataset

Figure 9: Plots obtained by applying the Gradient Descent algorithm.



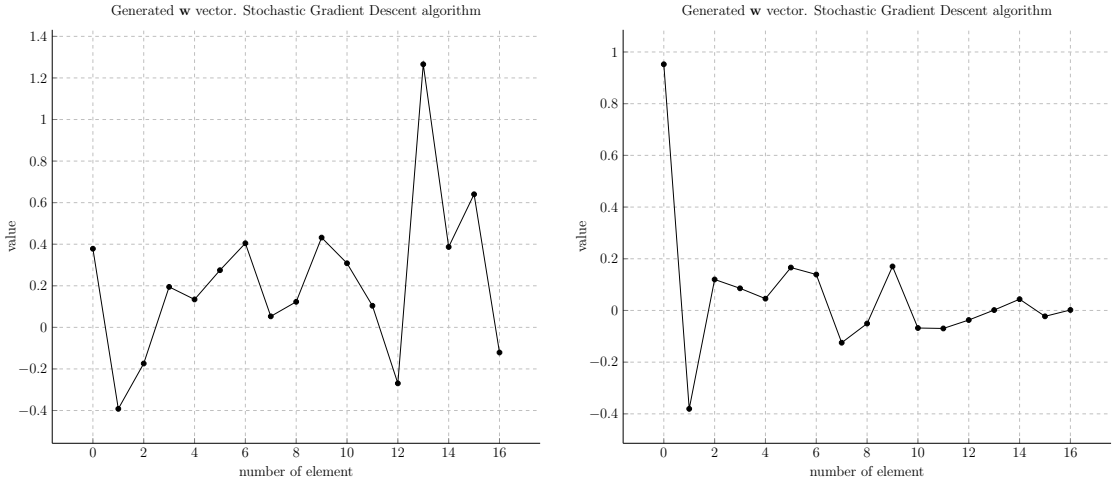
(a) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Train dataset (b) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Test dataset

Figure 10: Histograms obtained by applying the Gradient Descent algorithm.

5 Stochastic Gradient Descent (SGD)

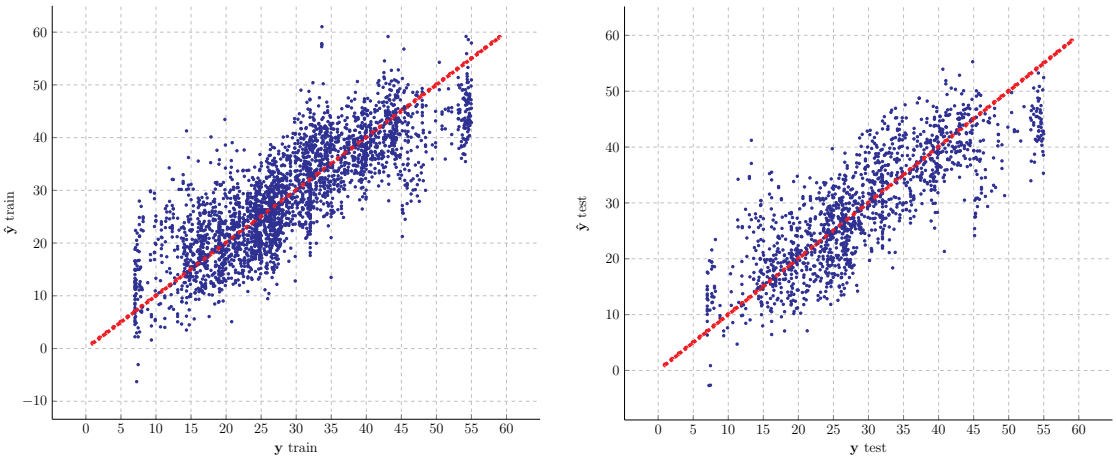
Stochastic Gradient Descent is a stochastic approximation of the Gradient Descent algorithm. The idea behind the method is to consider the cost function described in Equation (2) as summation of other functions $f(\mathbf{w}) = \sum_{i=0}^{N} f_i(\mathbf{w})$. In this way, while the Gradient Descent returns the value of $\nabla f(\mathbf{w})$ after each iteration, the SGD returns an approximation of the gradient by evaluating it only for a group of function addenda.

Even in this case the obtained results are strictly linked to the value of the learning coefficient γ and to the number of iterations. The main disadvantage of this algorithm is that, from an implementation point of view, two nested loop are requires, so, even by using 600 iterations, an high computational power is required.



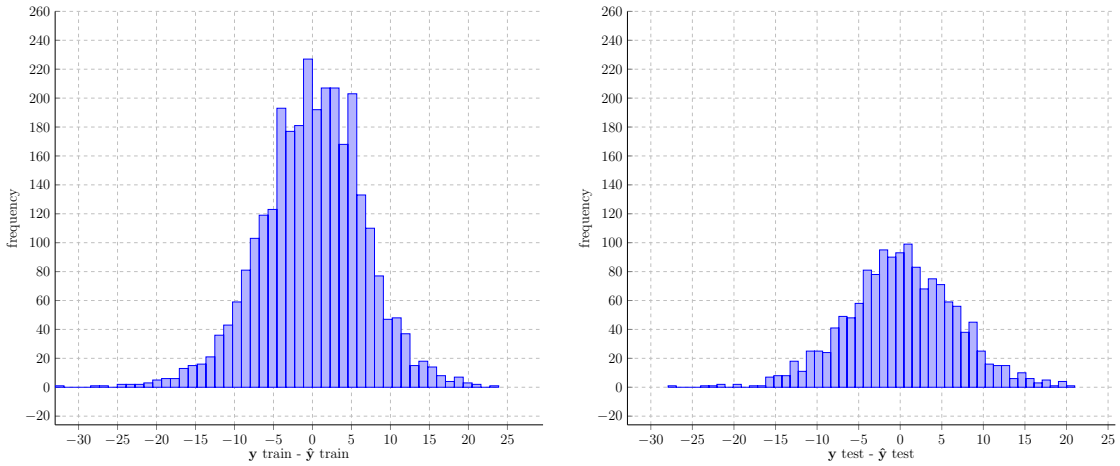
(a) $\gamma = 10^{-6}$, 100 iterations (b) $\gamma = 10^{-6}$, 10000 iterations

Figure 11: Vector $\mathbf{w}$ generated with the Stochastic Gradient Descent algorithm by setting different number of iterations



(a) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Train dataset (b) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Test dataset

Figure 12: Plots obtained by applying the SGD algorithm.



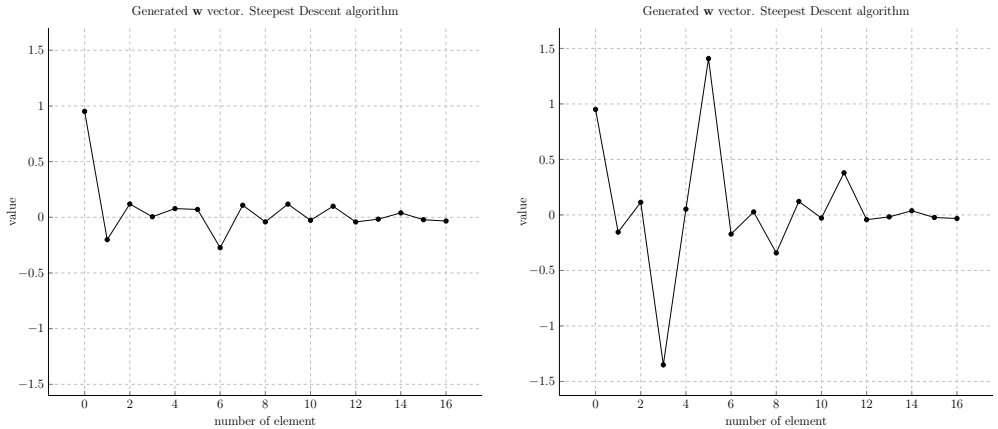
(a) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Train dataset (b) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Test dataset

Figure 13: Histograms obtained by applying the SGD algorithm.

The main advantage of the Stochastic Gradient Descent is that it is allowed the choice of a greater learning coefficient, thanks to the stochastic approximation. In fact, while $\gamma = 10^{-3}$ leads to a diverging mean square error when the Gradient Descent algorithm it is used, with the Stochastic it is reached an un-normalized error of 11.21 for the train dataset, 12.05 for the test dataset and 10.68 for the validation one.

6 Steepest Descent

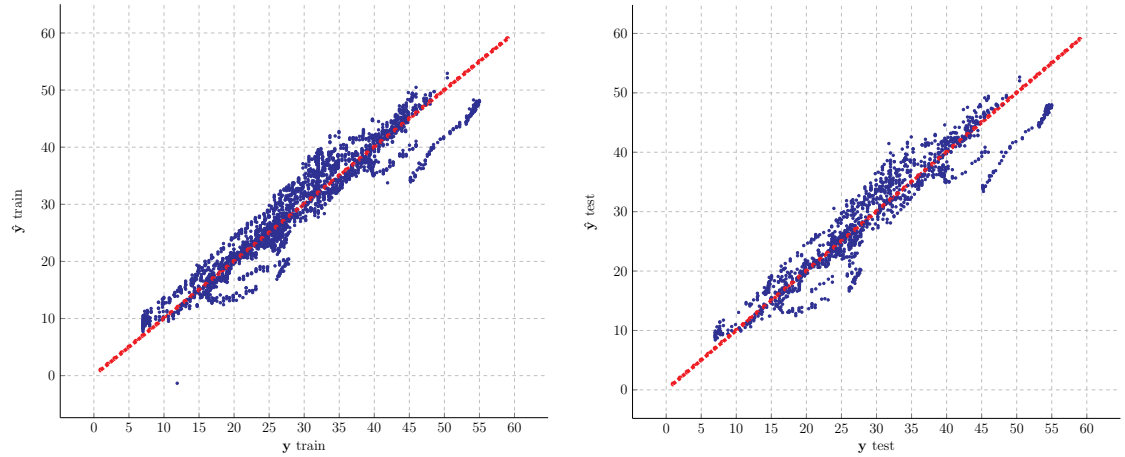
The Steepest Descent method is an approximation of the Gradient Descent algorithm, based on the idea that the learning coefficient γ is not arbitrarily chosen, but it is related to the Hessian matrix, $\mathbf{H}(\mathbf{w})$. In this way it is ensured that the gradient evaluated in the next step is orthogonal to the one in the previous step and this reduce the number of steps required to minimize the cost function.



(a) $\gamma = 10^{-6}$, 100 iterations (b) $\gamma = 10^{-6}$, 10000 iterations

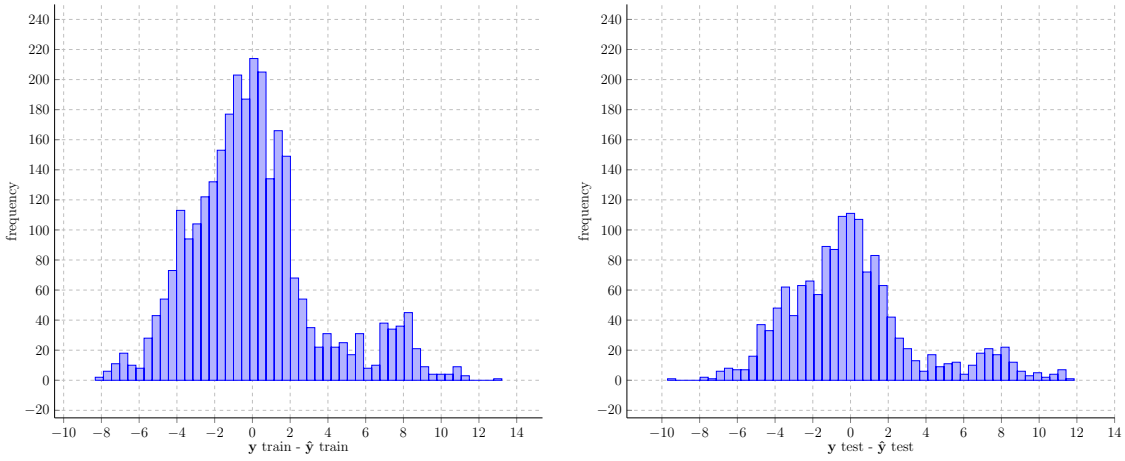
Figure 14: Vector $\mathbf{w}$ generated with the Steepest Descent algorithm by setting different number of iterations

According to what has been said in Section 4, by increasing the number of iterations, $\mathbf{w}$ is closer to the optimum vector, but, in contrast to Figure 7, the shape of plot shown in Figure 14(b) is not so different to the one in Figure 14(a); this is due to the number of steps optimization introduced by the Steepest algorithm.



(a) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Train dataset (b) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Test dataset

Figure 15: Plots obtained by applying the Steepest Descent algorithm.



(a) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Train dataset (b) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Test dataset

Figure 16: Histograms obtained by applying the Steepest Descent algorithm.

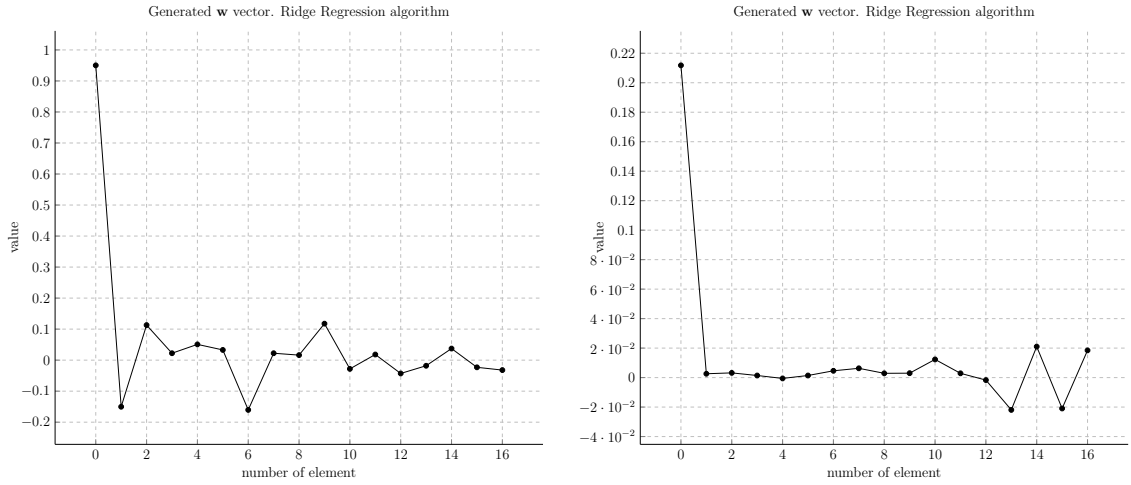
Results shown in Figure 15 and Figure 16 are obtained by applying the Steepest Descent algorithm with a $\gamma = 10^{-6}$ and 600 iterations. By comparing them to the ones obtained with the LLS and Conjugate Gradient methods, they are quite similar, proving the effectiveness of the Steepest in respect of the Gradient Descent and the SGD ones.

7 Ridge Regression

Sometimes Equation (3) may have an ill-conditioned $\mathbf{X}$. This means that in response to small changes in the input argument, the output value suffers from big changes. This leads to an overfitting issue. However this problem can be solved by applying the Ridge Regression technique.
In this way the cost function is:

$$f(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 + \lambda||\mathbf{w}||^2 \tag{5}$$

where $\mathbf{X}$, $\mathbf{y}$ and $\mathbf{w}$ are the same of Equation (2), while λ is a coefficient. This problem can be solved by using the pseudoinverse, but it is strictly linked to the value of λ.



(a) $\lambda = 1$ (b) $\lambda = 1000$

Figure 17: Vector $\mathbf{w}$ generated with the Ridge Regression algorithm by setting different number of iterations

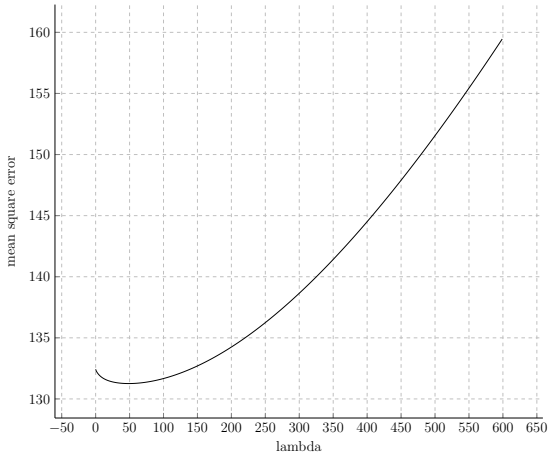
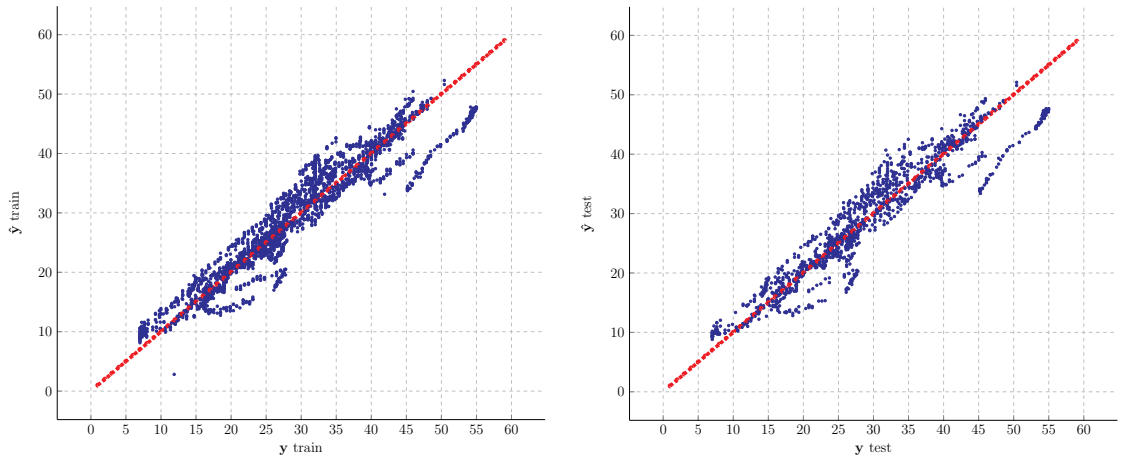


Figure 18: mean square error trend at an increasing value of λ

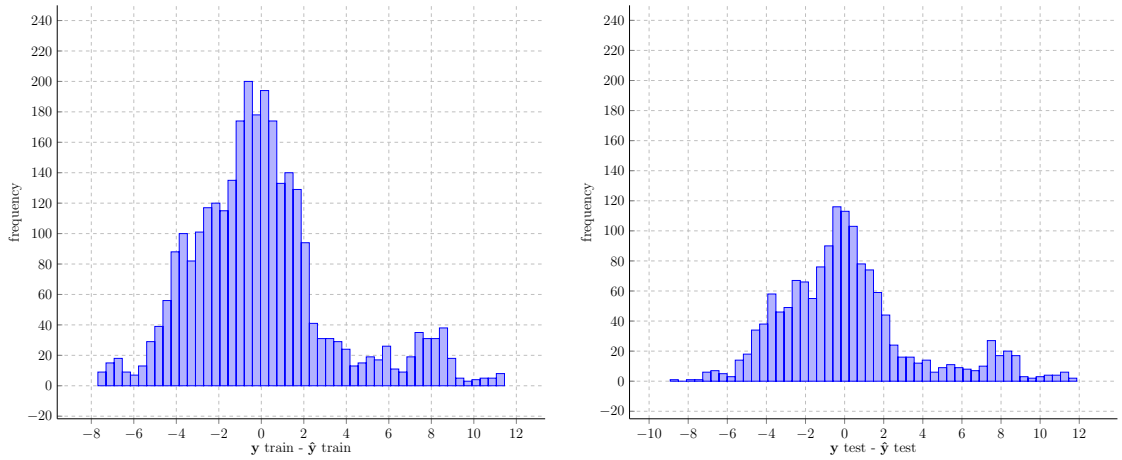
According to what has been said, for small value of λ the generated $\mathbf{w}$ is similar to the LLS and Conjugate Gradient one. On the other hand, by setting a very high value

of the coefficient, Equation (5) heavily changes the cost function, so $\mathbf{w}$ is different from all the other result. In this way, an hypothetical overfitting problem should be solved.



(a) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Train dataset (b) Comparison between $\hat{\mathbf{y}}$ and $\mathbf{y}$. Test dataset

Figure 19: Plots obtained by applying the Ridge Regression algorithm.



(a) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Train dataset (b) Difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$. Test dataset

Figure 20: Histograms obtained by applying the Ridge Regression algorithm.

Results shown in Figure 19 and Figure 20 are obtained by setting the optimum $\lambda = 49$. This value has been obtained by applying the same $\mathbf{w}$ obtained with the train dataset to the validation one. The measurements have been lead with an increasing number of λ (as shown in Figure 18) and than the optimum value has been chosen as the one which minimizes the error. Since the coefficient value is optimum, by comparing the results with the ones in Section 2 and Section 3, the Ridge Regression ones are almost the same.

8 Conclusions

After what has been stated, it is clear that the choice of the algorithm which best estimates the total UPDRS depends on many factors.

First of all it is necessary to analyze the amount of data: at first sight the LLS or the Conjugate Gradient methods seem to be perfect, but, if the data to work on are too much (especially if the test dataset is so much bigger than the train one), the optimum weight vector may correctly fit the second one, but it may fail the estimation of new data (overfitting). Even if in this case the Ridge Regression might solve the problem, however, by setting a bigger value of the λ coefficient would change the cost function too much introducing a bigger error.

Other two notable factors are the parameters of the Gradient Descent algorithm and its approximations. Even if a small learning coefficient γ might avoid a random oscillating movement around the minimum function point, it is required a very high number of iterations to obtain a model which best fits new data, and so a very high computational power requirements and a very slow processing.
The Steepest Descent seems to solve the iterations problem, but it is important to notice that a shorter time is balanced by the computational resources, since the Hessian matrix has to be determined at each step.

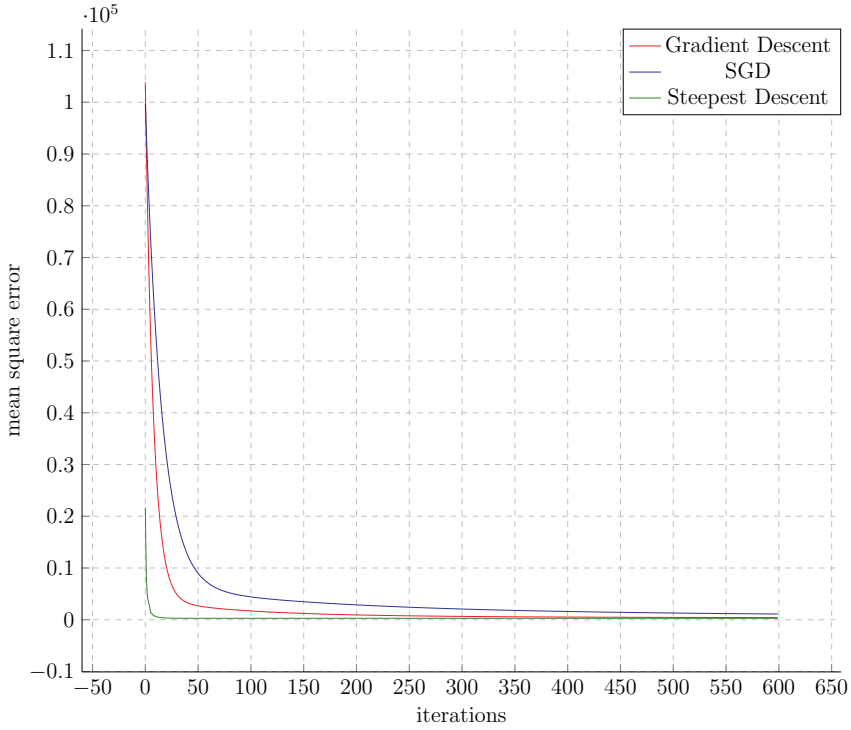


Figure 21: Comparison between the trend of the cost function related to an increasing number of iterations when three different algorithms are used. Normalized data.

In order to have an idea of the minimization speed of Gradient algorithms, Figure 21 shows how the cost function of Equation (2) decrease with an increasing number of iterations (from 1 to 600) and with $\gamma = 10^{-6}$. While all the plotted data in this report has been un-normailzed, in Figure 21 they are still normalized.

	Train Dataset	Test Dataset	Validation Dataset
LLS Pseudoinverse	11.168854	11.919981	10.630288
Conjugate Gradient	11.168854	11.919981	10.630288
Gradient Descent	16.497659	18.166697	15.960038
Stochastic Gradient	44.451097	43.065946	43.783110
Steepest Descent	11.171598	11.915038	10.627262
Ridge Regression	11.240603	11.956055	10.536509

Table 1: Mean Square Error between the estimated vector $\hat{y}$ and the original vector y.

Finally, an important parameter to be considered is the mean square error:

$$e_{MSE} = \frac{1}{N}||\mathbf{y} - \mathbf{X}\mathbf{w}||^2 \tag{6}$$

where N is the number of rows of the $\mathbf{X}$ matrix. Obviously, if the aim of the algorithms is to minimize the cost function, the mean square error needs to be minimized too.
In Table 1 are shown the value of un-normalized mean square error obtained by applying all the algorithms to the three datasets. The parameters are $\lambda = 100$, $\gamma = 10^{-6}$ and 600 iterations.

According to what has been said, the LLS and the Conjugate Gradient methods return the same results, since the generated $\mathbf{w}$ is the optimum one. The Gradient Descent introduces a slightly bigger error because of the number of iteration which is not big enough. Consequently, the Stochastic Gradient method performed the worst results, while the Steepest Descent improves the outcomes. Finally, the Ridge Regression with $\lambda = 100$ introduces a slight distortion in the $\mathbf{w}$ generation, which slightly increases the error.

A Normalization

According to the linear regression algorithms it is necessary to insert a column of ones in the $\mathbf{X}$ matrix. In this way, a constant offset it is introduced to ensure that the estimated $\hat{\mathbf{y}}$ coordinates are aligned in relation to the first quadrant bisector.
However, it is possible to avoid the additional ones column by applying the standard score normalization.

By considering $\mathbf{X} \in \mathbb{R}^{MxN}$ and $\mathbf{y} \in \mathbb{R}^{M}$, when the ones column is added, $\mathbf{X} \in \mathbb{R}^{Mx(N+1)}$. In this way, the linear problem becomes:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = [\mathbf{x_1}, \ldots, \mathbf{x_N}, 1] \begin{bmatrix} \hat{w_1} \\ \vdots \\ \hat{w}_{N+1} \end{bmatrix} \tag{7}$$

Now, $\mathbf{y}$ is:

$$\mathbf{y} = \mathbf{x_1}\hat{w_1} + \mathbf{x_2} + \ldots + \hat{w}_{N+1} = \hat{w}_{N+1} + \sum_{k=1}^{N} \mathbf{x_k}\hat{w_k} \tag{8}$$

Where $\hat{w}_{N+1}$ is the constant offset obtained by the multiplication of the ones column to the last element of generated $\hat{\mathbf{w}}$.

To proof that the standard score normalization is sufficient to avoid the column addition, it is necessary to write down the normalized $\mathbf{y}$:

$$\frac{\mathbf{y} - \mu_y}{\sigma_y} = \frac{\mathbf{x_1} - \mu_{\mathbf{x_1}}}{\sigma_{\mathbf{x_1}}} w_1 + \ldots + \frac{\mathbf{x_N} - \mu_{\mathbf{x_N}}}{\sigma_{\mathbf{x_N}}} w_N \tag{9}$$

And so:

$$\mathbf{y} = \left(\frac{\sigma_y}{\sigma_{x_1}} w_1 \mathbf{x_1} + \ldots + \frac{\sigma_y}{\sigma_{\mathbf{x_N}}} w_N \mathbf{x_N} \right) + \left(\mu_y - \frac{\sigma_y}{\sigma_{\mathbf{x_1}}} w_1 + \ldots + \frac{\sigma_y}{\sigma_{\mathbf{x_N}}} w_N \right) \tag{10}$$

Where μ_y and μ_{x_i} are the mean values of $\mathbf{y}$ and the *i-th* column of $\mathbf{X}$, while σ_y and σ_{x_i} are the standard deviations of $\mathbf{y}$ and the *i-th* column of $\mathbf{X}$.

Now, with a mathematical substitution it is possible to write down Equation (10) as Equation (8):

$$\begin{cases} \mathbf{y} = \left(\sum_{k=1}^{N} \mathbf{x_k} \hat{w_k} \right) + \hat{w}_{N+1} \\ \hat{w_k} = \frac{\sigma_y}{\sigma_{\mathbf{x_k}}} w_k \\ \hat{w}_{N+1} = \mu_y - \sum_{j=1}^{N} \frac{\sigma_y}{\sigma_{\mathbf{x_j}}} \mu_{x_j} w_j \end{cases} \tag{11}$$

Finally, it is clear that, thanks to a simple mathematical substitution, by applying a standard score normalization it is possible to avoid the ones column addition.