POLITECNICO DI TORINO

ICT For Smart Societies

ICT for Health (Lab 1)

Prof. Monica Visintin

REGRESSION
on
PARKINSON DATA

GENNARO RENDE

S218951

Contents

Chapter 1

Introduction

1.1 Parkinson's Disease

Parkinson's disease affects patients, not enabling them to control their muscles (difficulties in starting a movement) and also to control their voice because of the rigidity of the muscles' vocal cords. Patients who suffer from this disease, are given certain amounts of Levodopa, which acts by transforming itself, in the organism, into dopamine: a neurotransmitter responsible, among other things, of the movement's control.

1.2 The medical issue

"This illness is under the check of neurologists who judge the state of health of their patients by asking them to perform different movements. This type of control has the purpose of optimizing the therapy, which consists in taking a precise amount of Levodopa. It is difficult for neurologists to regulate the treatment because of the continuous progression of the illness. All the movements made by the patient, during the check, are numerically evaluated and summed: this is the final grade, which is called total UPDRS (Unified Parkinson's Disease Rating Scale). To improve the research, for each patient, It would be useful to find an automatic way to give them an objective score, which can be measured several times during the day and help the neurologist to optimize, in the best way possible, the treatment." [4]

1.3 The Data

"This dataset is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson's disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring. The recordings were automatically captured in the patient's homes. Columns in the table contain subject number, subject age, subject gender, time interval form baseline recruitment date, $\texttt{UPDRS}_{\mathbf{motor}}$,

$\texttt{UPDRS}_{\textbf{total}}$ and 16 biomedical voice measures. Each row corresponds to one of 5.875 voice recording from these individuals. The main aim of the data is to predict the motor and the $\texttt{UPDRS}_{\textbf{total}}$ scores from the 16 voice measures." [2]

1.3.1 Managing the data

"We use the first 50% of the rows of data as **training points**, the subquent 25% as **validation points and the remaining 25% of the rows as testing points**."[4]

1.3.2 Standardization of the data

Standardization is often used to avoid biased results because it centers the data around zero and divides them by their standard deviation.This procedure makes comparable all the features. A not standardized matrix requires the addition of a column of ones, giving the data a dimension of $N \times F+1$. In the case the real model remains non standardized, it has an offset and we have a matrix of F dimensions; this case, won't be able to predict new data. Standardization eliminates offsets such that we can solve the Regression problem in F dimensions.

1.4 Regression Analysis

"In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships among variables. It provides modeling and analysis of several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). Regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed."[1] The modus operandi we will follow is regressing the feature $\texttt{UPDRS}_{\textbf{total}}$ from the other features with different algorithms. The matrix $\mathbf{X}$ will contain all the features except $\texttt{UPDRS}_{\textbf{total}}$, the column vector $\mathbf{y}$ will instead filled only with the feature that $\mathbf{X}$ is missing. The correlation between these data structures will allow us to find the vector $\hat{\mathbf{w}}$ which data represent how much each feature of $\mathbf{x}$ is correlated to $\mathbf{y}$. Eventually, a new vector $\hat{\mathbf{y}}$ containing the regressed feature $\texttt{UPDRS}_{\textbf{total}}$ can be obtained by:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}} \tag{1.1}$$

Some of the algorithms require a lot of iterations in order to find $\hat{\mathbf{w}}$, for this reason it is useful to impose a stopping condition in order to stop the algorithm when the improvement of the solution becomes too small:

$$\|\hat{\mathbf{w}}_{i+1} - \hat{\mathbf{w}}_i\| < \epsilon \tag{1.2}$$

Chapter 2

Algorithms

To have constant results we used the "seed" function in Python (in this case with number **2**), $\epsilon = \mathbf{10^{-4}}$ and $\gamma = \mathbf{10^{-5}}$

2.1 Linear Least Square

In our study we did n measurements: $y(n) = \mathbf{x}^T\mathbf{w} + \boldsymbol{\nu}$ ($\boldsymbol{\nu}$ is a measurement error) ending up with matrices settled in this way:

$$\mathbf{X} = \begin{bmatrix} x_1(1) & x_2(1) & x_2(1) & \dots & x_F(1) \\ x_1(2) & x_2(2) & x_2(2) & \dots & x_F(2) \\ x_1(3) & x_2(3) & x_2(3) & \dots & x_F(3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1(N) & x_2(N) & x_2(N) & \dots & x_F(N) \end{bmatrix} = \begin{bmatrix} \mathbf{x}^T(1) \\ \mathbf{x}^T(2) \\ \mathbf{x}^T(3) \\ \vdots \\ \mathbf{x}^T(N) \end{bmatrix}; \quad \mathbf{y} = \begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \\ y(N) \end{bmatrix} \tag{2.1}$$

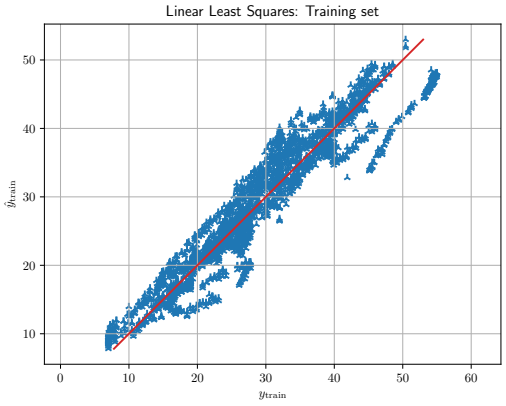
$$\text{we have: } \mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\nu} \tag{2.2}$$

$\mathbf{y}$ are our measurements, while $\mathbf{X}$ is a data-matrix of $\mathbf{N}$ rows (the number of patients) and F columns (their features). Our intent is to find the optimum $\mathbf{w}$ (the solution) to minimize the square error: $f(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$. Calculating the expression $\nabla f(\mathbf{w}) = 0$ we then find the optimum $\mathbf{w}$ as: $\hat{\mathbf{w}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$

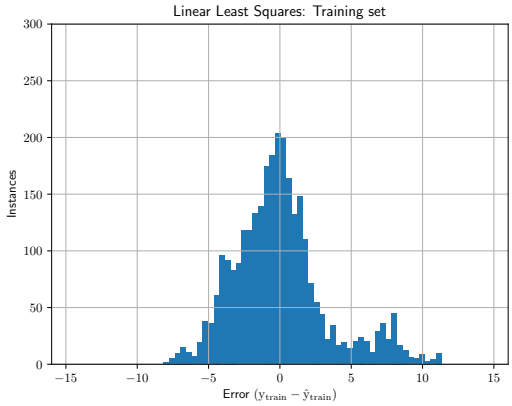
2.1.1 Results

In Figure **2.1a** and **2.1b** we see how the train data behave: the scattering plot well surrounds the diagonal (which represents the values that are supposed to show up) and the histogram has a Gaussian distribution with $\mu = 0$, meaning that our results mostly have an error around 0. In Figure **2.1c** and **2.1d** we see how the test data performed and, seeing the good results of our trained data, we receive a fine equal performance. Finally we check the Figure **2.1e** that shows of the optimum result depends especially on the values of $\texttt{Shimmer}_{\mathbf{APQ3}}$, $\texttt{Shimmer}_{\mathbf{DDA}}$, and $\texttt{UPDRS}_{\mathbf{motor}}$.

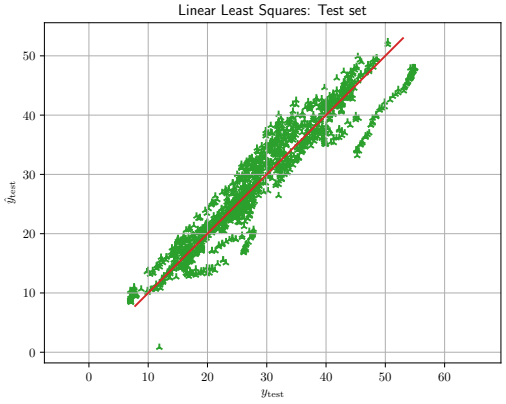
Results with the "Linear Least Square" Algorithm



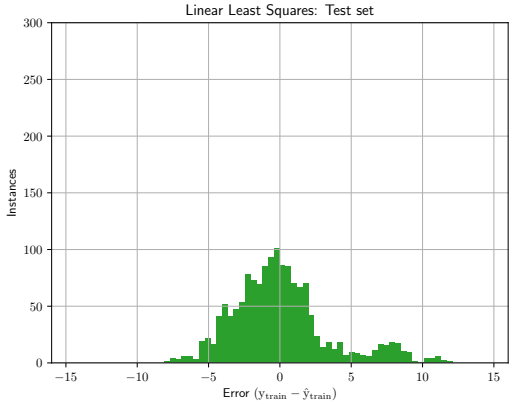
(a) $\hat{\mathbf{y}}_{\text{train}}$ *VS* $\mathbf{y}_{\text{train}}$



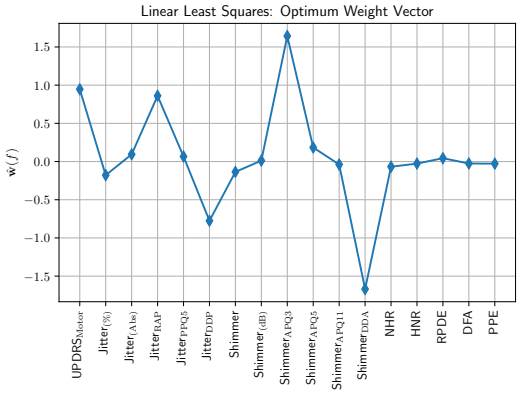
(b) $\hat{\mathbf{y}}_{\text{train}}$- $\mathbf{y}_{\text{train}}$



(c) $\hat{\mathbf{y}}_{\text{test}}$ *VS* $\mathbf{y}_{\text{test}}$



(d) $\hat{\mathbf{y}}_{\text{test}}$- $\mathbf{y}_{\text{test}}$



(e) *Optimum* $\boldsymbol{w}$ *for every feature*

Figure 2.1: Linear Least Square

2.2 Gradient Descent

"Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent,one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point."[3] This minimization algorithm relies to the fact that the maximum descent direction in a point of a function is the opposite of the direction of the gradient in that point. After having evaluated the gradient as

$$\nabla f(\mathbf{w}) = 2\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) \tag{2.3}$$

we chose the learning coefficient γ and iterate for a discrete number of iterations:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma \nabla f(\mathbf{w}_i) \tag{2.4}$$

In the end the vector $\mathbf{w}$ converges to an optimum solution to reach the point of minimum of the function as in [Figure 2.2]

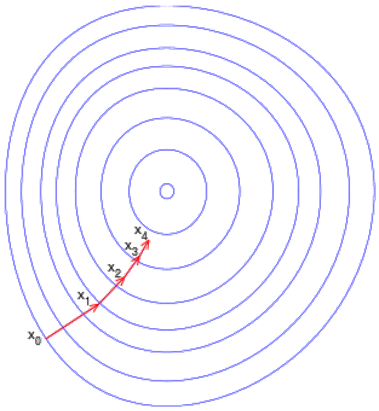
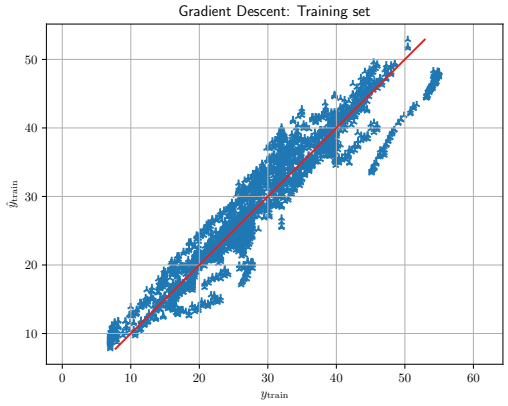


Figure 2.2: Gradient Descent [3]

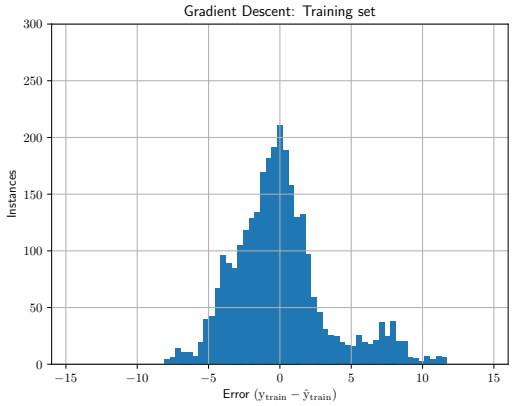
2.2.1 Results

Also in this case the output of $\hat{\mathbf{y}}$ is well performed, as we can see in [**2.3a**] and the error shown in [**2.3b**] confirms how little it affects the results since the $\mu = 0$. On the other hand we see a different result for the output of the $\hat{\mathbf{w}}$: the feature that has a significant correlation with $\texttt{UPDRS}_{\mathbf{total}}$ is $\texttt{UPDRS}_{\mathbf{motor}}$ while the others have a little impact (exception made for `Shimmer` and $\texttt{Shimmer}_{\mathbf{APQ5}}$). The figure [**2.3f**] shows the trend of the Mean Square Error VS the number of iterations made to improve the result of $\hat{\mathbf{w}}$: after almost 500 iterations, the training set, the test set and the validation set have the same trend and values and no overfitting is seen since the two functions almost overlap.

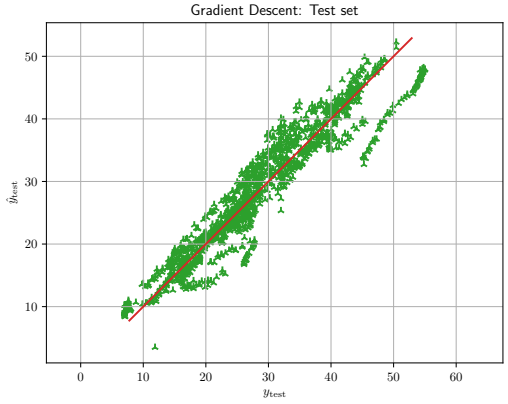
Results with the "Gradient Descent" Algorithm



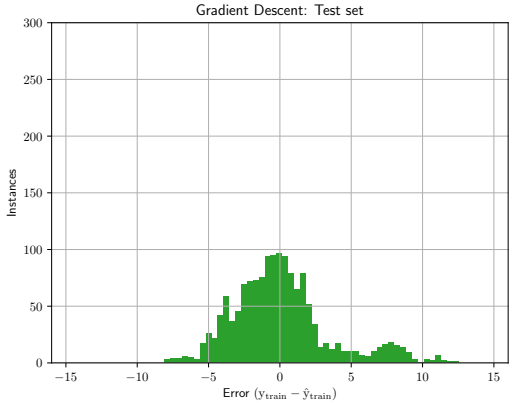
(a) $\hat{\mathbf{y}}_{\text{train}}$ *VS* $\mathbf{y}_{\text{train}}$



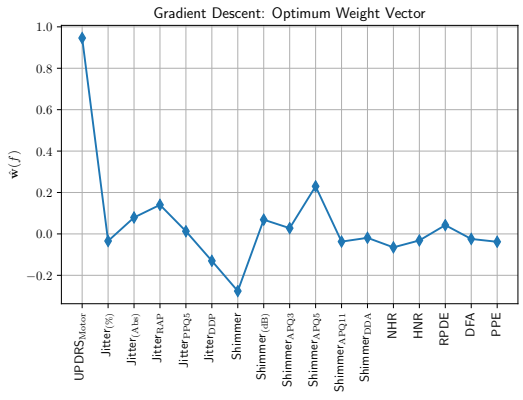
(b) $\hat{\mathbf{y}}_{\text{train}}$ - $\mathbf{y}_{\text{train}}$



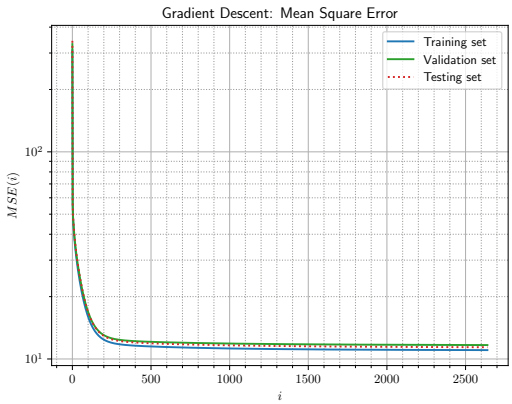
(c) $\hat{\mathbf{y}}_{\text{test}}$ *VS* $\mathbf{y}_{\text{test}}$



(d) $\hat{\mathbf{y}}_{\text{test}}$ - $\mathbf{y}_{\text{test}}$



(e) *Optimum* $\boldsymbol{w}$ *for every feature*



(f) *Mean Square Error*

Figure 2.3: Gradient Descent

2.3 Steepest Descent

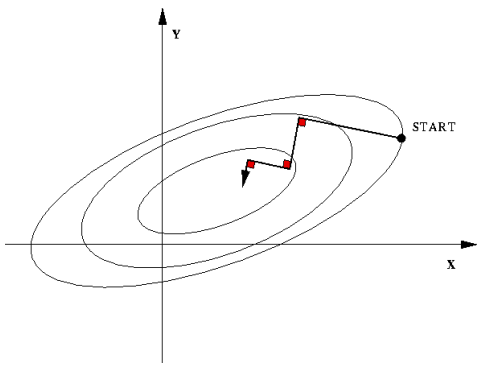


Figure 2.4: Steepest Descent

An other algorithm used to find the minimum of a function is the Steepest Descent. The main task with it, is choosing the best learning coefficient γ. This turns out to be quite tricky. If we use a constant learning coefficient, but make it too small, convergence will be very slow, but if we make it too large, the method can fail to converge at all. Our goal is always to find the optimum vector $\mathbf{w}$ and with this iterative method, we will improve step by step our γ by evaluating, for each $\mathbf{x_i}$ the $\nabla f(\mathbf{x_i})$ and the Hessian Matrix in that point: $\mathbf{H}(\mathbf{x}_i) = 4\mathbf{X}^T\mathbf{X}$.

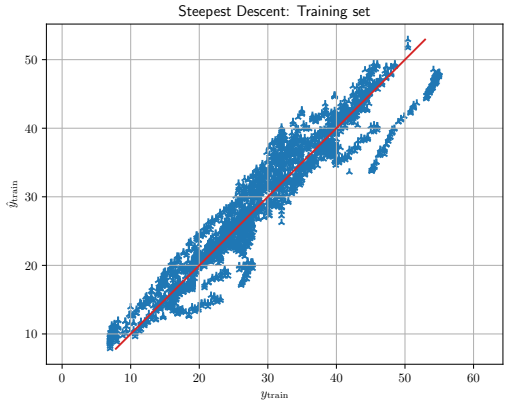
$$\gamma_i = \frac{\|\nabla f(\mathbf{x}_i)\|^2}{\nabla f(\mathbf{x}_i)^T \mathbf{H}(\mathbf{x}_i) \nabla f(\mathbf{x}_i)} \tag{2.5}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{\|\nabla f(\mathbf{x}_i)\|^2}{\nabla f(\mathbf{x}_i)^T \mathbf{H}(\mathbf{x}_i) \nabla f(\mathbf{x}_i)} \nabla f(\mathbf{x}_i) \tag{2.6}$$

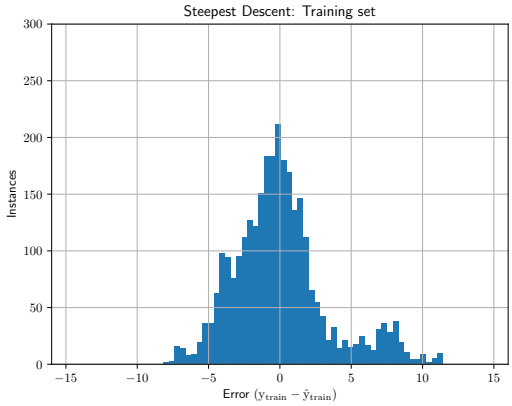
2.3.1 Results

As in the previous algorithms, in the figures[**2.5a**] and [**2.5c**]we see how well the algorithm performs and the enhancement of it is also proven in the figures [**2.5c**] and [**2.5d**] where the mean of the error is around 0.More than in figure [**2.3e**] we see how crucial is the feature of $\texttt{UPDRS}_{\mathbf{motor}}$ among the other features which has the highest in defying, among the other features, the correlation with the $\texttt{UPDRS}_{\mathbf{total}}$. The Mean Square Error here performs better than in [**2.3f**] because the number of iterations required to have a constant trend is only around 50.

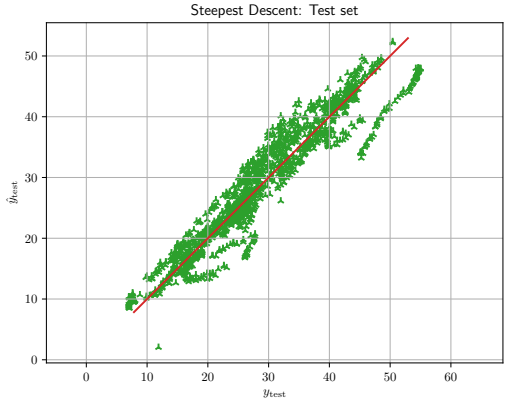
Results with the "Steepest Descent" Algorithm



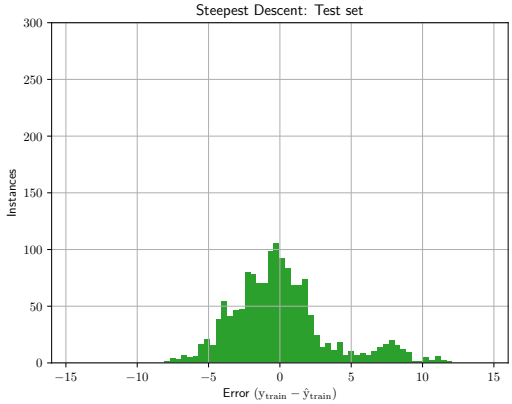
(a) $\hat{\mathbf{y}}_{\text{train}}$ *VS* $\mathbf{y}_{\text{train}}$



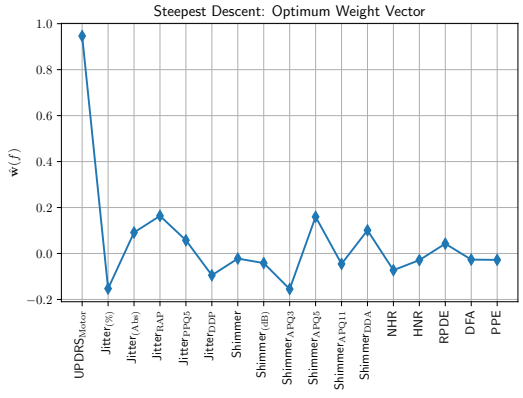
(b) $\hat{\mathbf{y}}_{\text{train}}$ - $\mathbf{y}_{\text{train}}$



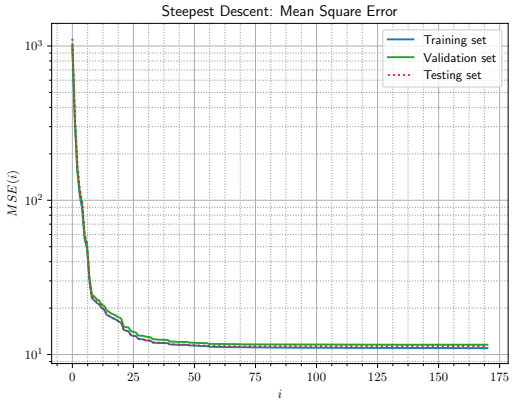
(c) $\hat{\mathbf{y}}_{\text{test}}$ *VS* $\mathbf{y}_{\text{test}}$



(d) $\hat{\mathbf{y}}_{\text{test}}$ - $\mathbf{y}_{\text{test}}$



(e) *Optimum* $\boldsymbol{w}$ *for every feature*



(f) *Mean Square Error*

Figure 2.5: Steepest Descent

2.4 Stochastic Gradient Descent

This algorithm it's very similar to the Gradient Descent (2.2) but the initial approach is totally different. The iterations to find the minimum of the function, start with a stochastically generated $\mathbf{w}$. Our function is defined as in (2.7), while the gradient as in (2.8)

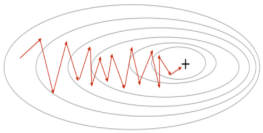


Figure 2.6: Stochastic Descent

$$f(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 = \sum_{n=0}^{N} [[\mathbf{x}(n)]^T \mathbf{w} - y(n)]^2 = \sum_{n=1}^{N} f_n(\mathbf{w}) \tag{2.7}$$

$$\nabla f(\mathbf{w}) = \sum_{n=0}^{N} \nabla f_n(\mathbf{w}) = \sum_{n=0}^{N} [[\mathbf{x}(n)]^T \mathbf{w} - y(n)]\mathbf{x}(n) \tag{2.8}$$

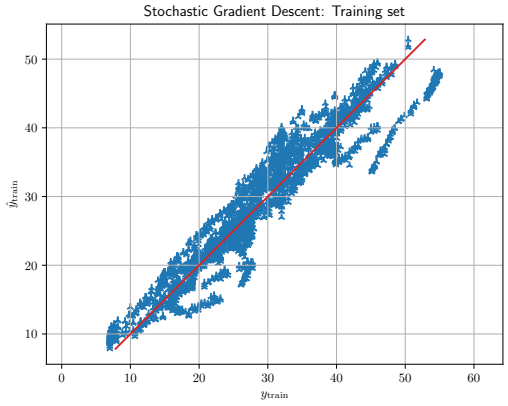
Now, as we did in (2.4), we iterate until our stopping condition is met:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma \nabla f_i(\mathbf{w}_i) \tag{2.9}$$

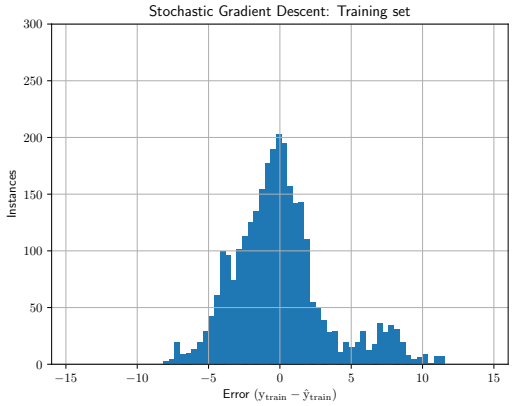
2.4.1 Results

Like in sections before, the figures **[2.7a]**, **[2.7c]**, **[2.7b]**and **[2.7d]** are representative of the good results given by the algorithm; And also here is crucial the amount of importance that the feature $\texttt{UPDRS}_{\mathbf{motor}}$, in figure **[2.7e]**, has in the regression analysis. The particularity of this algorithm is, as we said in the initial description of it, that it updates the vector w using an estimate of the gradient based on the single rows of the matrix. The evaluation will surely be faster but not so accurate and this is because it takes so much time to reach convergence. For each iteration shown in figure **[2.7f]** the algorithm evaluated the gradient 2938 times that is the number of rows of the $\mathbf{X}_{train}$ matrix.

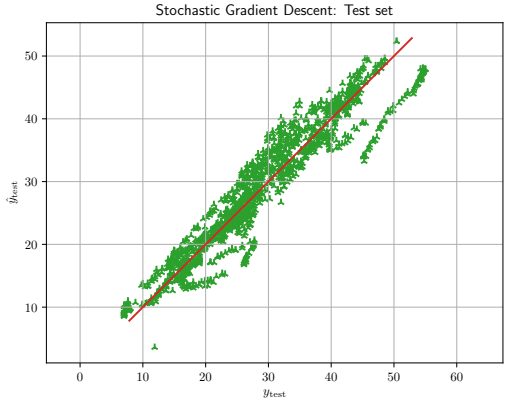
Results with the "Stochastic Gradient Descent" Algorithm



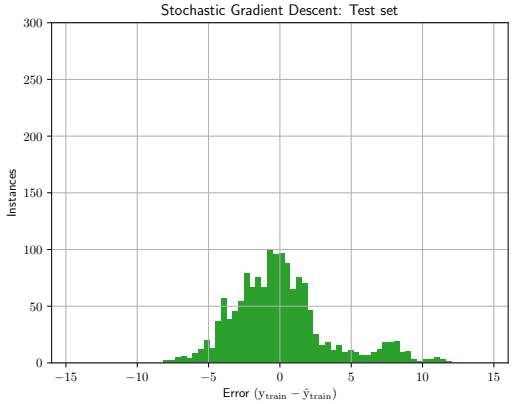
(a) $\hat{\mathbf{y}}_{\text{train}}$ *VS* $\mathbf{y}_{\text{train}}$



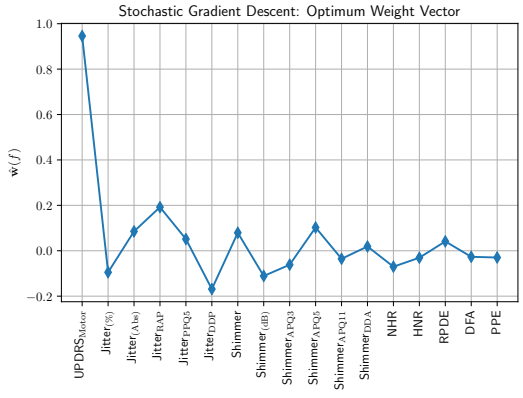
(b) $\hat{\mathbf{y}}_{\text{train}}$- $\mathbf{y}_{\text{train}}$



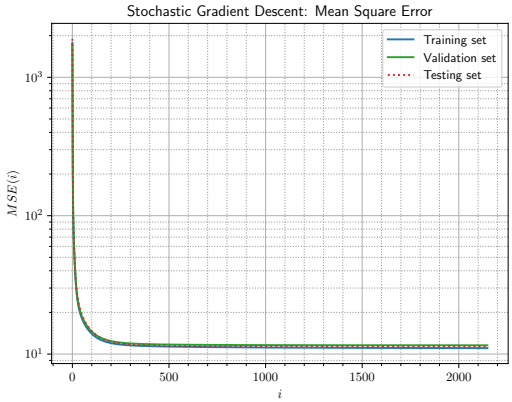
(c) $\hat{\mathbf{y}}_{\text{test}}$ *VS* $\mathbf{y}_{\text{test}}$



(d) $\hat{\mathbf{y}}_{\text{test}}$- $\mathbf{y}_{\text{test}}$



(e) *Optimum* $\boldsymbol{w}$ *for every feature*



(f) *Mean Square Error*

Figure 2.7: Stochastic Gradient Descent

2.5 Conjugate Gradient Descent

The conjugate gradient method operates similarly to the gradient descent. The difference is that the directions taken after each iteration are the conjugate of the previous ones. This leads to a very fast convergence since it skips a lot of steps that the gradient descent would have made [6].

The problem to solve is (2.10):

$$\mathbf{Q}\mathbf{w}^* - \mathbf{b} = 0 \tag{2.10}$$

A solution can be found with the help of *conjugate vectors* which are vectors orthogonal with respect to a matrix $\mathbf{Q}$. It means that the vectors $\mathbf{d}_i$ and $\mathbf{d}_k$ are Q-orthogonal if $\mathbf{d}_i^T\mathbf{Q}\mathbf{d}_k = 0$.

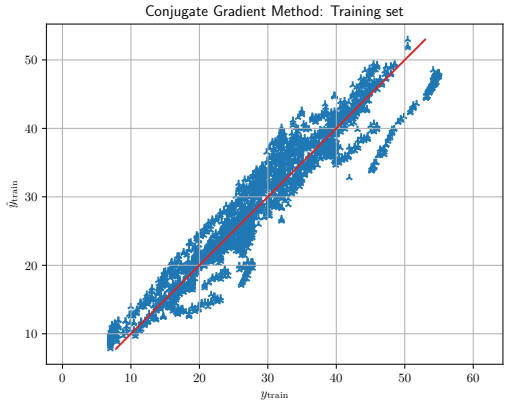
At first we set $\mathbf{d}_0 = -\mathbf{g}_0 = \mathbf{b}$ and $\mathbf{w}_0 = 0$ as the initial solution, where $\mathbf{g}$ is the direction of the gradient and $\mathbf{d}$ is the actual direction taken by the algorithm. The next directions are computed as $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k\mathbf{d}_k$ where β is a coefficient that depends on the other parameters.

The algorithm converges in N_f steps where N_f is the dimension of the problem.

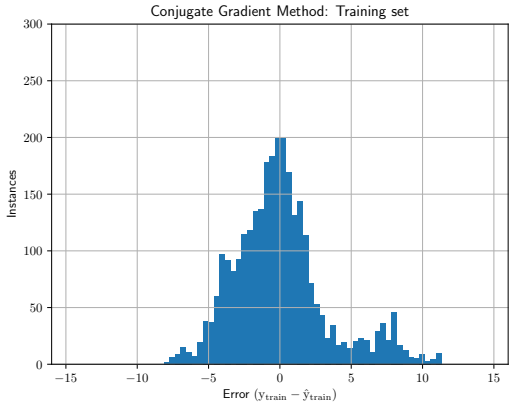
2.5.1 Results

The good results of this algorithm are not only explained by the plots [**2.8a**] and [**2.8c**] and by the histograms of figure[**2.8b**] and [**2.8d**] but especially by the Mean Square Error graph [**2.8f**]. The Conjugate Gradient Method, with just few iterations, is able to reach a constant train (in our case it has stopped after 17 iterations) and the amount of overfitting is visible only in the first few iterations.

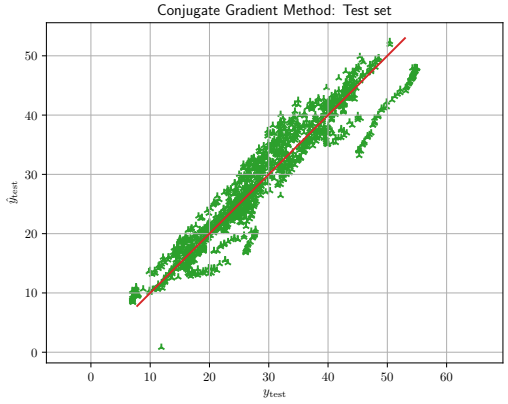
Results with the "Conjugate Gradient Descent" Algorithm



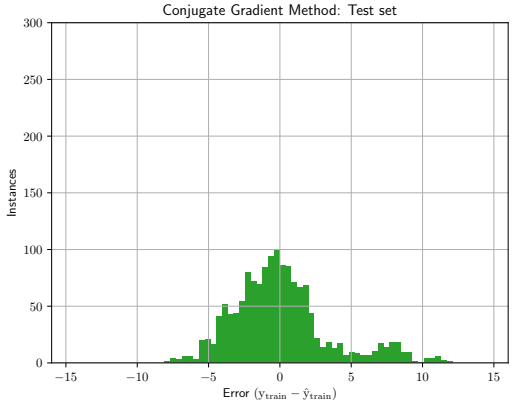
(a) $\hat{\mathbf{y}}_{\text{train}}$ *VS* $\mathbf{y}_{\text{train}}$



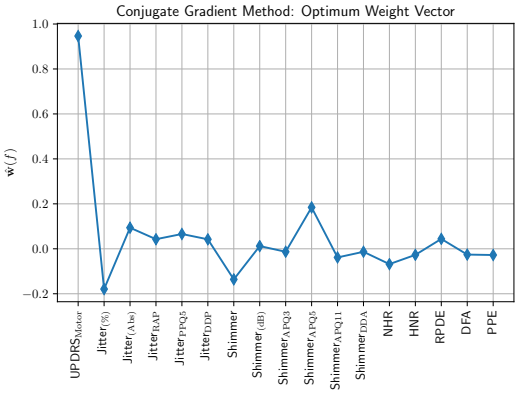
(b) $\hat{\mathbf{y}}_{\text{train}}$- $\mathbf{y}_{\text{train}}$



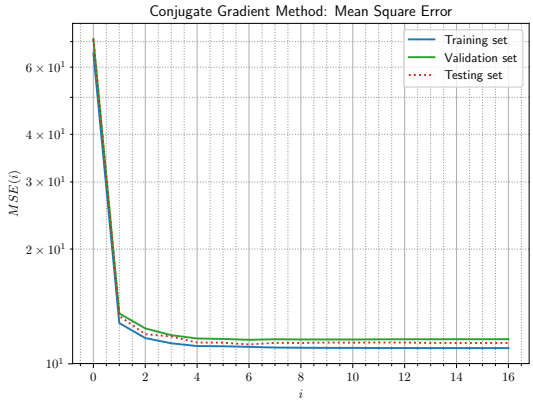
(c) $\hat{\mathbf{y}}_{\text{test}}$ *VS* $\mathbf{y}_{\text{test}}$



(d) $\hat{\mathbf{y}}_{\text{test}}$- $\mathbf{y}_{\text{test}}$



(e) *Optimum* $\boldsymbol{w}$ *for every feature*



(f) *Mean Square Error*

Figure 2.8: Conjugate Gradient Descent

2.6 Ridge Regression [5]

If $\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\nu}$ has some large values of noise/error, it is possible that vector $\hat{\mathbf{w}}$ takes very large values and overfitting occurs. Then, it might be convenient to solve the new problem:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2 = \min_{\mathbf{w}} f(\mathbf{w}) \tag{2.11}$$

where μ has to be set conveniently (trial and error). The solution of this problem can then be obtained using the pseudoinverse, in fact

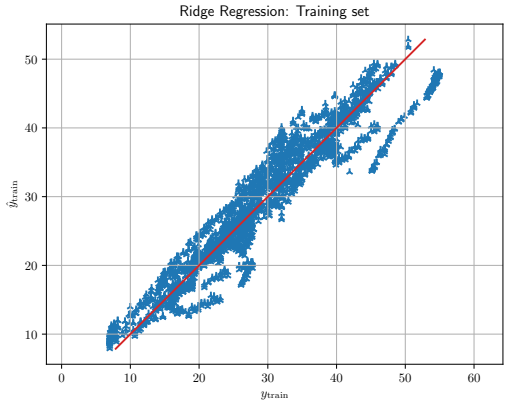
$$\nabla f(\mathbf{w}) = 2\mathbf{X}^\top\mathbf{X}\mathbf{w} - 2\mathbf{X}^\top\mathbf{y} + 2\lambda\mathbf{w} = 0 \tag{2.12}$$

$$\hat{\mathbf{w}} = \left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y} \tag{2.13}$$

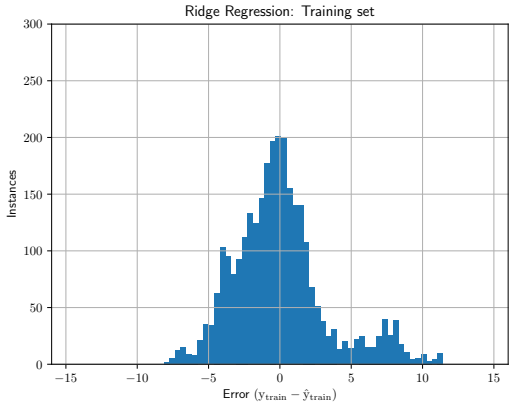
2.6.1 Results

With this algorithm we tried to find the best result by choosing the best λ between $[0, 80]$. The result for us was $\lambda = 6.835085$, since in our case it is the value that minimizes the mean squared error on the validation set, as we can see in figure **[2.9f]**. The rest of the figures, similarly as before, are significant of the good work of the algorithm in doing regression analysis.

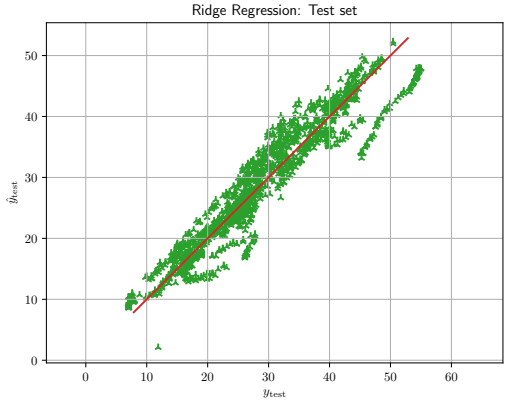
Results with the "Ridge Regression" Algorithm



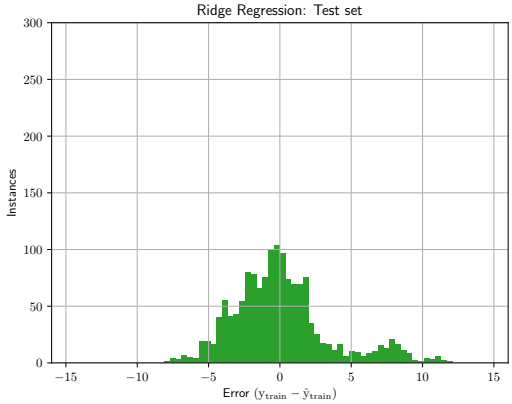
(a) $\hat{\mathbf{y}}_{\text{train}}$ *VS* $\mathbf{y}_{\text{train}}$



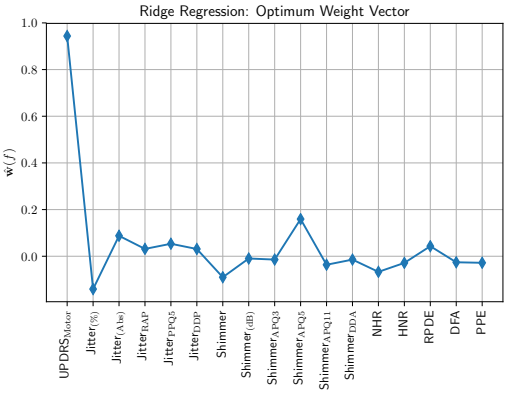
(b) $\hat{\mathbf{y}}_{\text{train}}$- $\mathbf{y}_{\text{train}}$



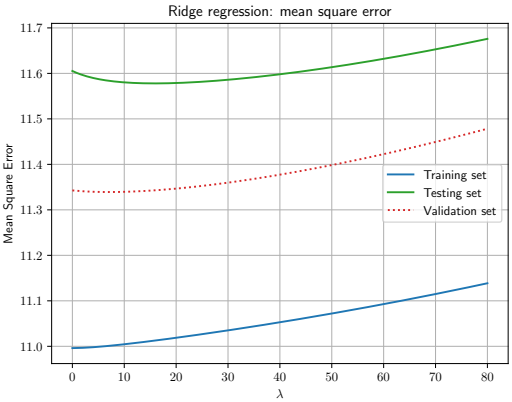
(c) $\hat{\mathbf{y}}_{\text{test}}$ *VS* $\mathbf{y}_{\text{test}}$



(d) $\hat{\mathbf{y}}_{\text{test}}$- $\mathbf{y}_{\text{test}}$



(e) *Optimum* $\boldsymbol{w}$ *for every feature*



(f) *Mean Square Error*

Figure 2.9: Ridge Regression

Chapter 3

Comparisons beetween the algorithms

To conclude we saw how, among all the features of the data matrix, $\texttt{UPDRS}_{\mathbf{motor}}$ was the most significant one for the regression on the $\texttt{UPDRS}_{\mathbf{total}}$ feature. Table 3.1 shows the mean squared errors for each algorithm and each dataset. We can see that the MSE for the training set is always the lowest one, due to the fact that the optimum weight vector is evaluated on the training set itself. The mean squared errors on the test and validation sets do not differ too much since they contain the same $\hat{\mathbf{w}}$.

Algorithm	**Iterations**	**Training MSE**	**Testing MSE**	**Validation MSE**
Linear Least Square	1	10.9960	11.6060	11.3431
Gradient Descent	2641	11.0363	11.6739	11.3950
Steepest Descent	172	11.0046	11.5785	11.3496
Stochastic Gradient Descent	6312400	11.0413	11.5692	11.3542
Conjugate Gradient Descent	17	10.9961	11.6053	11.3429
Ridge Regression	1	11.0008	11.5840	11.3391

Table 3.1: Final results in terms of MSE and Number of Iterations for Training, Testing and Validation Data

Bibliography

[1] Regression Analysis
https://en.wikipedia.org/wiki/Regression_analysis

[2] Machine Learning Data
https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring

[3] Gradient Descent
https://en.wikipedia.org/wiki/Gradient_descent

[4] ICT for Health Laboratory 1: Regression on Parkinson data
https://didattica.polito.it/pls/portal30/sviluppo.scheda_pers_swas.show?m=1834

[5] Linear Regression: Slides of Lecture 3
https://didattica.polito.it/pls/portal30/sviluppo.scheda_pers_swas.show?m=1834

[6] On My Phd: *Conjugate Gradient Method*
http://www.onmyphd.com/?p=conjugate.gradient.method