Université de Genève

Data Mining

TP 3: Linear Regression

Author: Joao Filipe Costa da Quinta

E-mail: Joao.Costa@etu.unige.ch

May 17, 2021

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES

Département d'informatique

Introduction

In this tp we are going to implement Linear regression, using it's analytical solution. We'll also try to optimize the solution with gradient descent and seeing the impact of different learning rates, we'll be able to compare the results of both methods.
We'll add some noise to the y target vector, and see how it affects our models.
Afterwards we'll see the impact of linear dependency between two features, and try to solve this problem with L2 regularisation, this method should also improve our results overall.

Linear Regression

To see how good our result is, we must have an error function in place, the goal is to minimize of the sum of squares, which represents the sum of distance of every point to our prediction. This is computed with the following formula:

goal : $\min_w \|Xw - y\|^2$

$\rightarrow$ we derive it

$$\nabla E(w) = \nabla (y - Xw)^T (y - Xw) \rightarrow \text{distribute}$$
$$= -2X^T y + (X^T X + (X^T X)^T) w$$
$$= -2X^T y + 2(X^T X) w$$
$$= -2X^t (y - Xw)$$

$\nabla E(w) = 0 \quad \rightarrow$

$$\boxed{X^T X} w = X^T y$$

$\hookrightarrow$ invertible (we assume)

$$w = (X^T X)^{-1} X^T y$$

Analytical Solution

To get our analytical solution we simply take the formula seen previously, the sum of squares, derive it, and set it to 0.

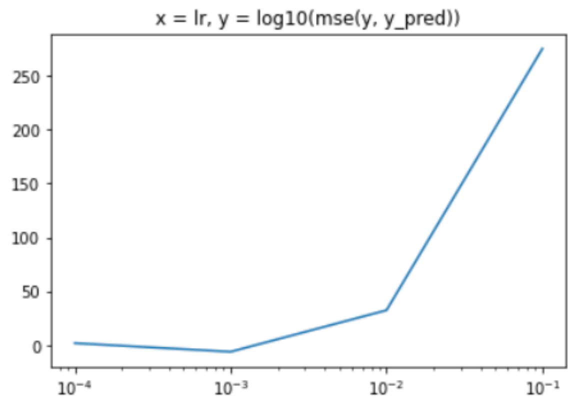
Gradient Descent

If we have an analytical solution, why would we need to optimize with gradient descent ? Because the analytical solution might not always be possible to compute, I will explain why later, when we introduce linear dependency between features.

Results

When comparing the results from both methods, we get better results with analytical solution.
Depending on the learning rate, as expected, the results vary, a lot, the best results were obtained with lr = 0.001.

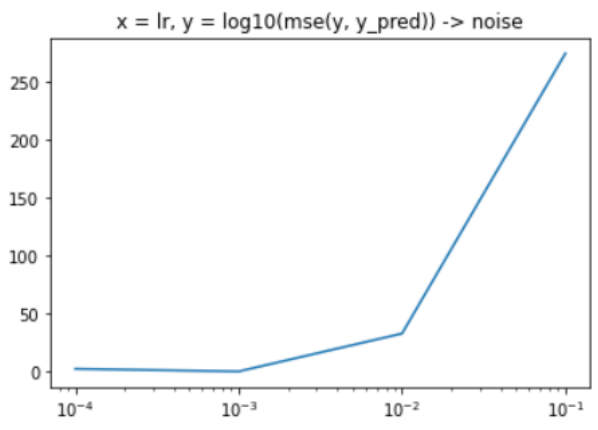
```
X_test error for analytical solution is       :  5.564421850464314e-14
lr = 0.0001 : X_test error for GD solution is :  46.85359168297029
lr = 0.001  : X_test error for GD solution is :  0.004959621352855823
lr = 0.01   : X_test error for GD solution is :  8.512620120394656e+16
lr = 0.1    : X_test error for GD solution is :  1.4186212266445061e+138
x   :  [0.0001, 0.001, 0.01, 0.1]
mse :  [109.76295267972512, 1.2298921981851709e-06, 3.623235065707396e+32, 1.0062430923431818e+275]
```



Adding Noise to y_test

The difference in results is quite large, but more importantly, we lose a lot of accuracy in the analytical solution, which means it will be preferable to use Gradient descent, it has worse results in the best case scenario, but much better when the data is corrupted.

```
X_test error for analytical solution is       :  3.5069375464947483
lr = 0.0001 : X_test error for GD solution is :  46.51876213512659
lr = 0.001  : X_test error for GD solution is :  3.507183730906596
lr = 0.01   : X_test error for GD solution is :  8.44169422253485e+16
lr = 0.1    : X_test error for GD solution is :  1.4083748899948299e+138
x   :  [0.0001, 0.001, 0.01, 0.1]
mse :  [108.1997615292244, 0.6150168861167955, 3.5631100673389134e+32, 9.917599153839746e+274]
```



Linear dependency

Let c1 be the first column of the matrix X, we simply add it to the end of X as a new column, and we can multiply it by any value, the result will be the same, for good measure we can do it for the second column as well.

Let X be a matrix nxd
Let's add a new column equal to the first one

$$X^t \begin{bmatrix} - & l_1 & - \\ - & l_2 & - \\ & \vdots & \\ - & l_{d+1} & - \end{bmatrix} \cdot X \begin{bmatrix} | & | & & | \\ c_1 & c_2 & \cdots & c_{d+1} \\ | & | & & | \end{bmatrix} = \overset{X^tX}{\begin{bmatrix} l_1 \cdot c_1 & & l_1 \, c_{d+1} \\ l_2 \cdot c_1 & \cdots & l_2 \, c_{d+1} \\ \vdots & & \vdots \\ l_{d+1} \cdot c_1 & & l_{d+1} \, c_{d+1} \end{bmatrix}}$$

$l_{d+1} = l_1$ $c_{d+1} = c_1$

$\downarrow$ Since $c_1 = c_{d+1}$ these two columns are the same

(1) There is a theorem that says that if two columns of a matrix are linearly dependant, then its determinant will be equal to 0.
(2) There is a second theorem that says that if a matrix X is inversible, then its determinant is different from 0.
Knowing how X was created, and with (1) and (2), we know that X^tX is not inversible.

However Gradient descent is still doable, and adding 2 columns that are linearly dependant with two others didn't impact the results much.

L2 regularisation

When we have linear dependency between the features, we then have many possibilities for w, because we can change w_i at will by compensating with w_j, where j != i.
To fix this, we have to somehow add the w_i values to the function we are trying to minimize, the ultimate goal, is to minimise the norm of the target vector w.

goal: $$\min_w \underbrace{\|Xw - y\|^2}_{\text{same as before}} + \lambda w^t w$$

$$\Rightarrow -2x^t(y - Xw)$$

we derive $\lambda w^t w = \nabla_w \sum w_i^2$

$$= \left[\frac{\partial \sum w_i^2}{\partial w_1} \dots \frac{\partial \sum w_i^2}{\partial w_d}\right]$$

$$= [2w_1, \dots, 2w_d] = 2w$$

Set to 0

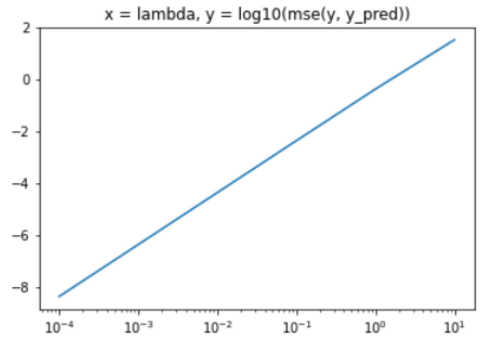
$$\rightsquigarrow w = \underbrace{(X^t X + \lambda I)}_{\text{will be inversible}}{}^{-1} X^t y$$

L2 regularisation - analytical solution

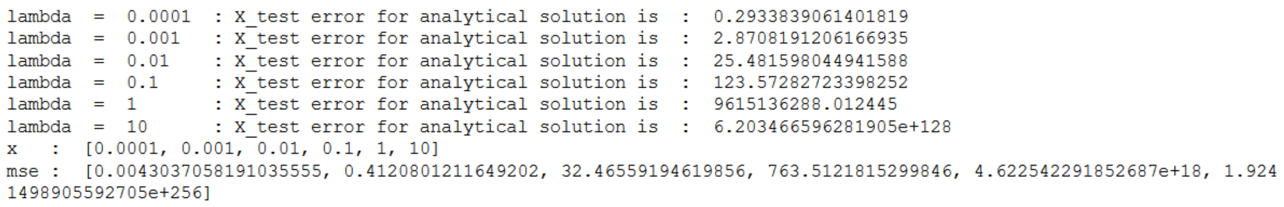
Like before, we derive our formula, and set to 0, to find the analytical formula.
We can see that instead of looking for the inverse of X^tX, we want the inverse of $X^tX + \lambda I$, which means that if X^tX were to have linear dependencies, then they are eliminated by adding $+\lambda I$.

However, the results weren't as good as previously, but at least the computation was possible.

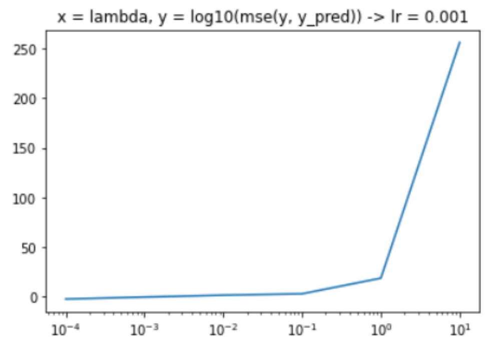
```
lambda  =  0.0001  : X_test error for analytical solution is  :  0.00029098887295368807
lambda  =  0.001   : X_test error for analytical solution is  :  0.002909850978401778
lambda  =  0.01    : X_test error for analytical solution is  :  0.029094735221431422
lambda  =  0.1     : X_test error for analytical solution is  :  0.29057050176303406
lambda  =  1       : X_test error for analytical solution is  :  2.8686148960653908
lambda  =  10      : X_test error for analytical solution is  :  25.48133205098631
x   :  [0.0001, 0.001, 0.01, 0.1, 1, 10]
mse :  [4.2337262091428815e-09, 4.233616358252891e-07, 4.2325180880260096e-05, 0.004221560824741068, 0.41144757109641
27, 32.46491415463113]
```



L2 regularisation - gradient descent



```
lambda  =  0.0001  : X_test error for analytical solution is  :  0.2933839061401819
lambda  =  0.001   : X_test error for analytical solution is  :  2.8708191206166935
lambda  =  0.01    : X_test error for analytical solution is  :  25.481598044941588
lambda  =  0.1     : X_test error for analytical solution is  :  123.57282723398252
lambda  =  1       : X_test error for analytical solution is  :  9615136288.012445
lambda  =  10      : X_test error for analytical solution is  :  6.203466596281905e+128
x   :  [0.0001, 0.001, 0.01, 0.1, 1, 10]
mse :  [0.004303705819103555, 0.4120801211649202, 32.46559194619856, 763.5121815299846, 4.622542291852687e+18, 1.924
1498905592705e+256]
```



Results

For both analytical and gradient descent, the best value for lambda was the lowest. However, for Gradient Descent, it gets worse much faster, so we have to be much more careful, when doing L2 reg with Gradient Descent.

Conclusion

In this TP we got to witness Linear regression models in action, I got better results with the analytical solution, hoewever, Gradient descent was capable of performing where the analytical solution couldn't, when the data set had linear dependant columns.
This problem was fixed with the introduction of l2 regularisation, where the results with analytical solution are very promising, when we reduced lambda value, from an already low value, the mse matrix got even better results, and with gradient descent we saw that it was 'dangerous' as the results get very bad very fast, but both are still a good option, just need to make sure we input the good values.
With this being said, one might want to use l2 regularisation with analytical solution, and call it a day, however, if the data is corrupt, as it was when we added noise, gradient descent performed much better than analytical solution.