Optimization for Machine Learning

Louis Reberga 10/01/2022

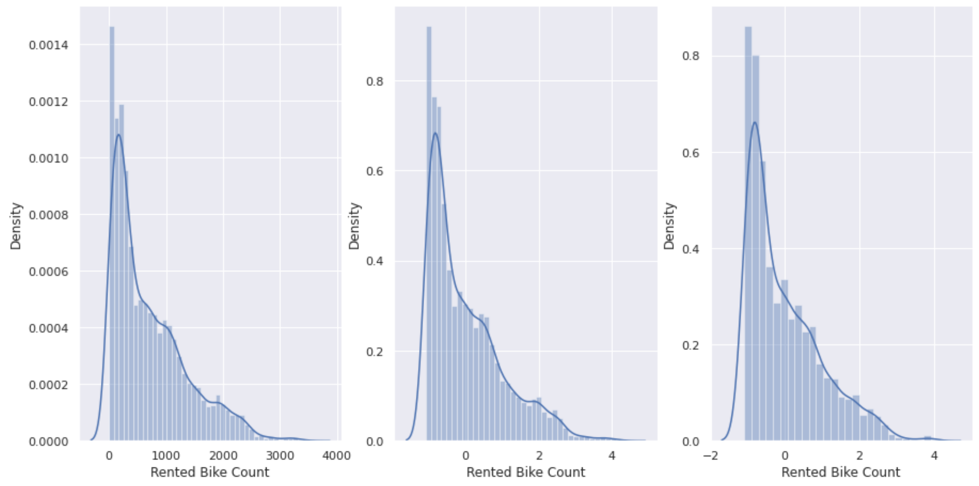
1 Presentation of the dataset

To do this project, we decided to work with the regression dataset "SeoulBikeData". This dataset describe the number of bikes rented in Seoul as a function of 12 other variables. There are 3 qualitative variables and 9 quantitative ones:

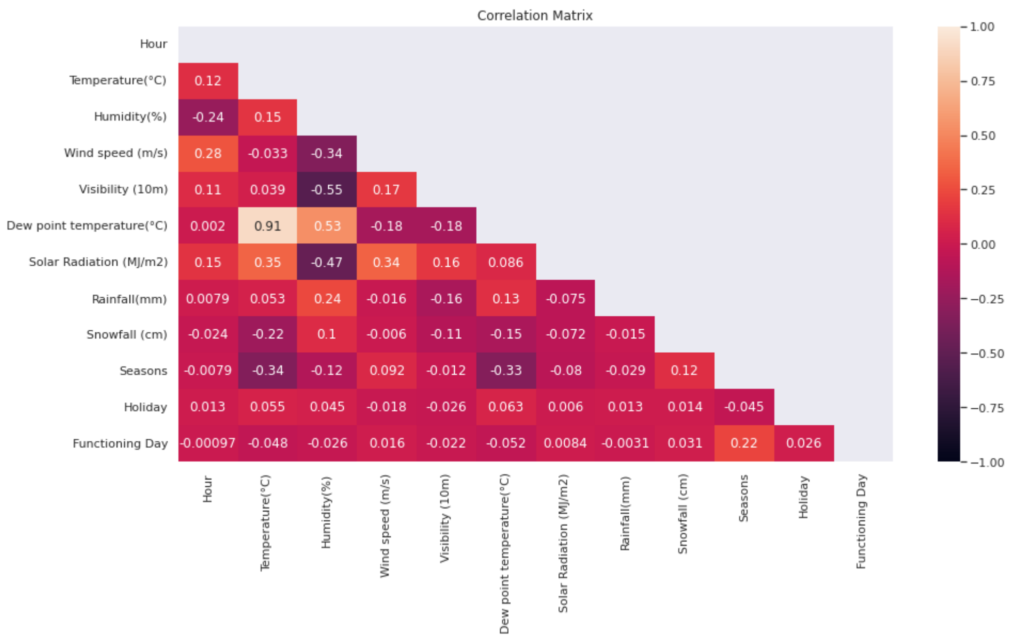
Qualitative: Seasons, Holiday, Functioning Day

Quantitative: Hour, Temperature (°C), Humidity (%), Wind speed (m/s), Visibility (10m), Dew point temperature (°C), Solar Radiation (MJ/m2), Rainfall (mm), Snowfall (cm)

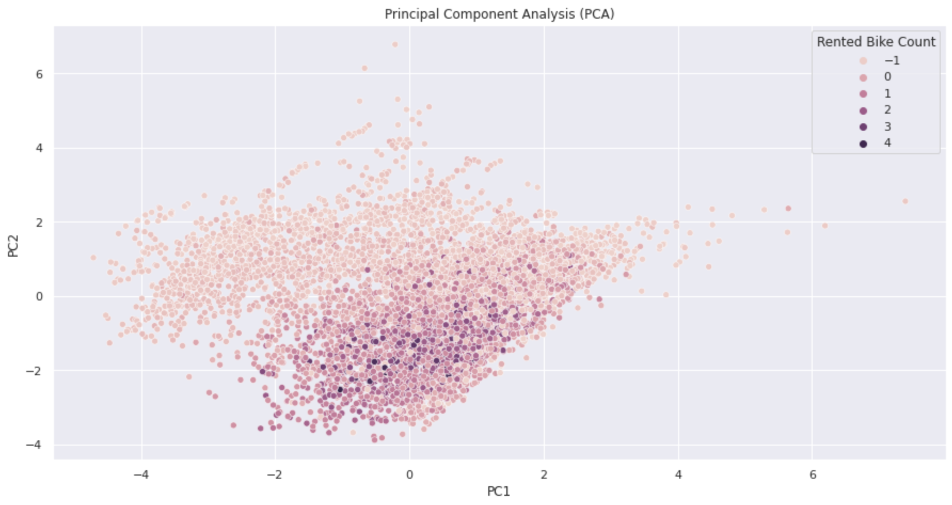
To evaluate the score of all the methods we implemented, we splitted the dataset in two parts: 70% for the trainnig and 30% to test the solution found. To be sure the datasets built are equivalently distributed, we first plotted the desnsity of the whole dataset, then the density the the train set and finaly the density of the test set. So, we can notice that the 3 datasets are equivalently distributed.



Then, we displayed the correlation matrix. It shows us that we have a correlation between the Temperature(°C) and the Dew point temperature(°C). The other variables are pretty independant.



We also did other graphs to compare the quantitative variables. You can see them in the Python notebook. To display the whole dataset in 2 dimensions we did a PCA. We colorated the points regarding their range on the target variable *Rented Bike Count*:



2 Presentation of the problem

Now, we have briefly presented the datatset, let's explain the problem. We want to solve a linear regression problem. For that we have to minimize the following loss function:

$$f(x) = \frac{1}{2}||Ax - y||^2 \tag{1}$$

where A is the matrix with all the features and y a vector with all the target values. We can find a solution to this problem by solving $\nabla f(x) = 0$.

$$\begin{aligned} \nabla f(x) &= 0 \\ A^\top(Ax - y) &= 0 \\ x &= (A^\top A)^{-1}Ay \end{aligned} \tag{2}$$

For our problem we found a solution with a relative prediction error equal to 0.677. But $A^\top A$ is not always inversible and it can be very costly to compute.

3 Gradient Descent

Another approach is to use iterative methods like Gradient Descent. The algorithm is based on the following update:

$$x_{k+1} = x_k - \tau \nabla f(x_k) \tag{3}$$

But how choose τ? We can use a fixed step size or update it after each new iteration. The Descent Property said if $0 < \tau < \frac{2}{L}$, with f L-lipschitz, then the Gradient Descent algorithm will converge. So let's calculate L, $\forall(u, v)$:

$$\begin{aligned} ||\nabla f(u) - \nabla f(v)|| &= ||A^\top(Au - y) - A^\top(Av - y)|| \\ &= ||A^\top Au - A^\top y - A^\top Av + A^\top y|| \\ &= ||A^\top A(u - v)|| \\ &\leq ||A^\top A||.||(u - v)|| \end{aligned} \tag{4}$$

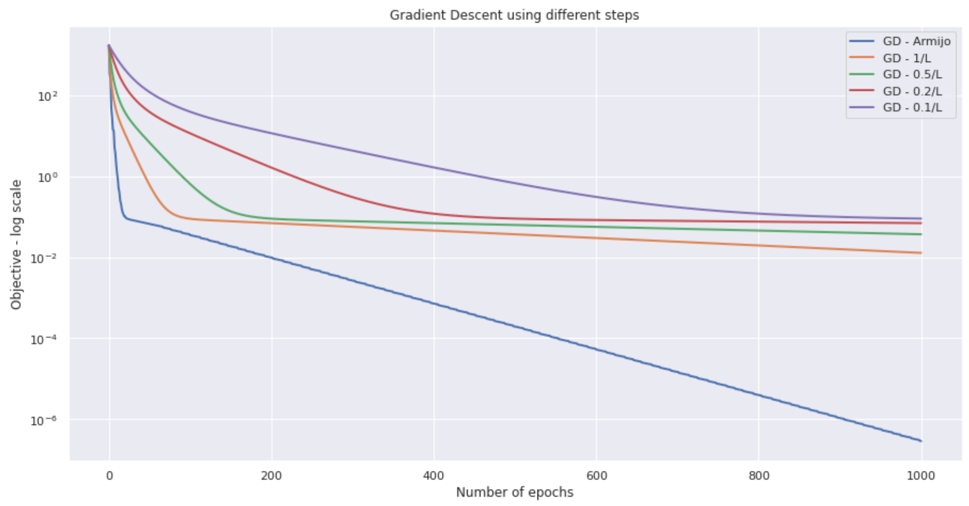
So ∇f is $||A^\top A||$-lipschitz.

For this project we also used the Armijo's step, a backtracking line search step based on the Armijo–Goldstein condition.

```python
def armijo(xk, dk, w=1e-4, p0=1, tau=0.25):
    k = 0
    pk = p0
    while loss_function(xk + pk * dk) > loss_function(xk) + w * pk * np.vdot
    (grad(xk), dk):
        pk = tau * pk
        k += 1

    return pk
```

Then, we computed the Gradient Descent Algorithm. Here is the results we obtained using Armijo and different fixed steps τ :



Aflter computing GD with different step size we can notice that more the step is close to $\frac{2}{||A^\top A||}$, more the Gradient Descent converges faster. When we use Armijo's step, the convergence is much more faster than using fixed steps. These are the theoretically expected results.

4 Accelerated Gradient: Nesterov

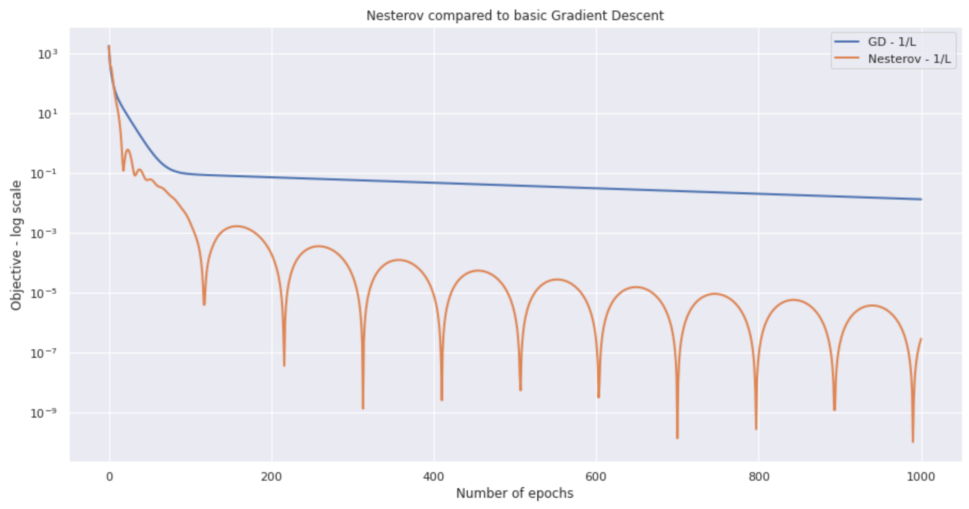
The loss function is a convex function. So we implemented Nesterov method for this kind of minimization problem, defined as follows:

$$\begin{cases} x_{k+1} = z_k + \tau \nabla f(z_k) \\ z_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k) \end{cases} \tag{5}$$

with,

$$\begin{cases} \beta_k = \dfrac{t_k - 1}{t_{k+1}} \\ t_{k+1} = \dfrac{1}{2}(1 + \sqrt{1 + 4t_k^2} \end{cases} \tag{6}$$

On the following graph, plotting Gradient Descent and Nesterov with $\tau = \frac{1}{L}$, we can observe that Nesterov converges faster. We can also observe "jumps" on the convergence curve of Nesterov, this behavior is due to the momentum added to x_k update.



5 Conjugate Gradient: Polak-Ribière

Polak-Ribière is an extension of the Conjugate Gradient method. The algorithm update is:

$$x_{k+1} = x_k + \tau d_k \tag{7}$$

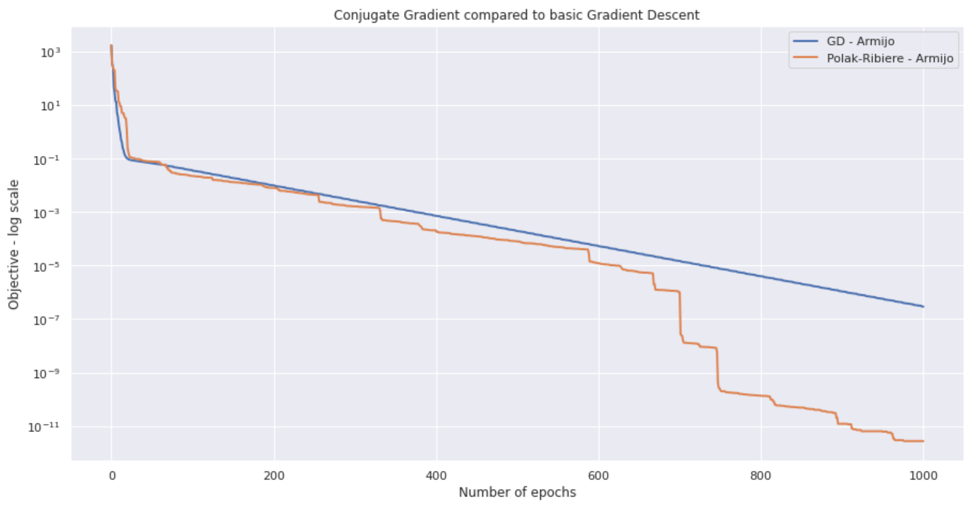
where,

$$d_k = -\nabla f(x_k) + \beta_k d_{k-1} \tag{8}$$

with,

$$\beta_k = \frac{\langle \nabla f(x_k), \nabla f(x_k) - \nabla f(x_{k-1}) \rangle}{||\nabla f(x_k)||^2} \tag{9}$$

On the following graph, plotting Gradient Descent and Polak-Ribière with an Armijo's step, we can observe that Polak-Ribière converges as fast as Gradient Descent until the 600^{th} epochs. Then, Polak-Ribière converges faster. The convergence curve of Polak-Ribière is not "smooth", this behavior is due to the conjugated term β_k added to x_k update.



6 Newton

Newton method is a second order Gradient Descent method because. Indeed, we use the Hessian of the loss function to compute the algorithm defined as follows:

$$x_{k+1} = x_k + \tau d_k \tag{10}$$

where,

$$d_k = -H_f(x_k)^{-1} \nabla f(x_k) \tag{11}$$

We observe a convergence after only one iteration. But this method is very costly. Indeed, we have to compute the Hessian of the function and also its inverse.

7 Stochastic Gradient

Stochastic Gradient method update is defined as:

$$x_{k+1} = x_k - \tau \nabla f_{i_k}(x_k) \tag{12}$$

where i_k is a random index drawn in $\{1, ..., n\}$.

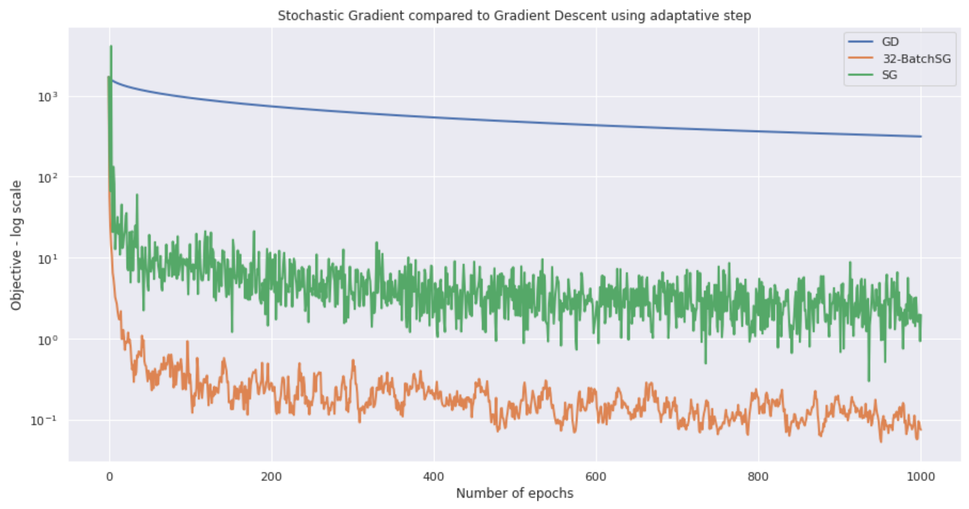
We can generalised Stochastic Gradient using Batch Stochastic Gradient defined as follows:

$$x_{k+1} = x_k - \tau \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(x_k) \tag{13}$$

where S_k is a set of random indexes drawn in $\{1, ..., n\}$.

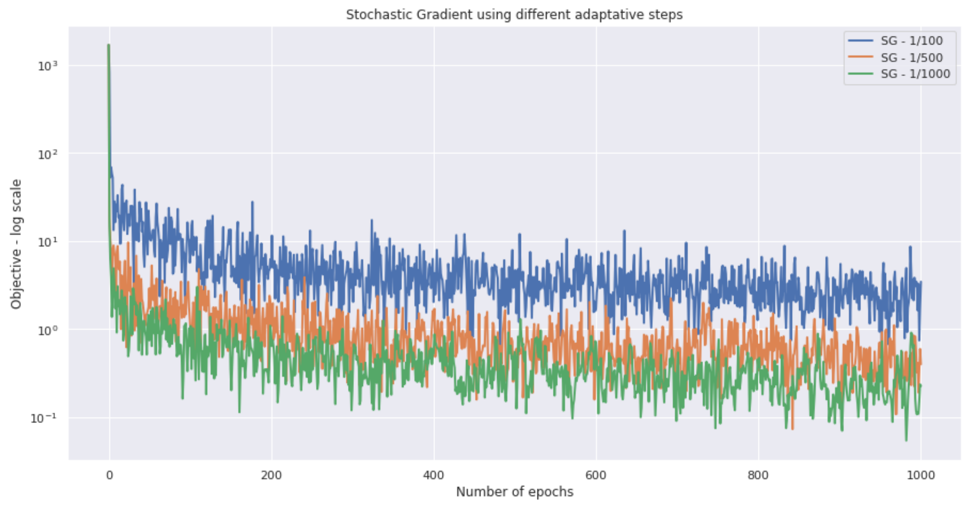
There are two specific cases for Batch Stochastic Gradient. If $|S_k| = 1$, we have Stochastic Gradient and if $|S_k| = n$ we have Gradient Descent.

For our computation, we decided to compare Gradient Desecent, Stochastic Gradient and Batch Stochastic Gradient with a batch size equal to 32. We also decided to choose an adaptative step size $\tau = \tau_k = \frac{\alpha}{\sqrt{k+1}}$ where α is a predefined constant.



With the same adaptative step size, Batch Stochastic Gradient converges faster than Gradient Descent and Stochastic Gradient. We can observe "distorsion" on BSG and SG curves. The "distorsions" are larger on SG because the difference between the gradients computed is bigger.

Then, we decided to study the impact of the step size on Stochastic Gradient. We kept $\tau_k = \frac{\alpha}{\sqrt{k+1}}$ and we tried 3 different values for α.



Trying different step sizes, we can notice that the convergence is faster when we use lower α.

8 Ridge Regularization

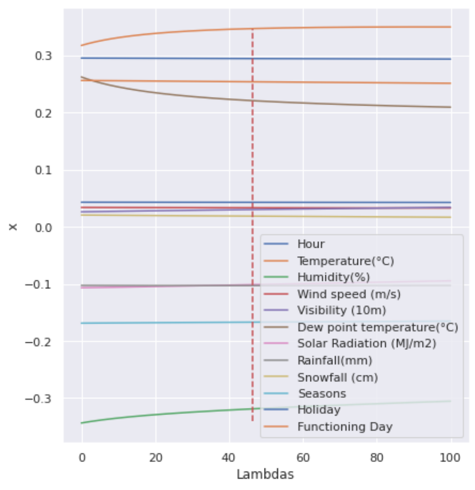
Compute Ridge Regularization consist in minimizing the following loss function :

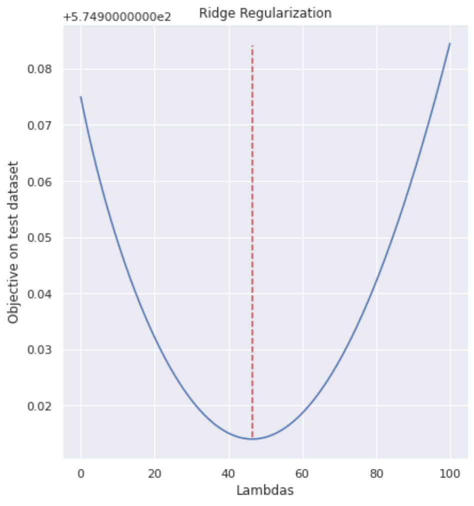
$$f(x) = \frac{1}{2}||Ax - y||^2 + \lambda||x||^2 \tag{14}$$

The main goal to minimize this function is to find the best λ possible. To do it, we compute x on the train dataset:

$$x = (A^\top A + \lambda I_d)^{-1} Ay \tag{15}$$

and we evaluate the value of the loss function (not regularized) on the test set. Then, we choose the λ where the loss function is the lowest. For our dataset, we obtained $\lambda = 46.47$. Ridge Regularization is also used to select the best features in a dataset. On the first graph bellow, you can observe the value of x as a function of the different values of λ.





9 Lasso Regularization - ISTA

An other regularization method is Lasso. The main goal is the same same: minimize the following function choosing the best λ possible.

$$f(x) = \frac{1}{2}||Ax - y||^2 + \lambda||x||_1 \tag{16}$$

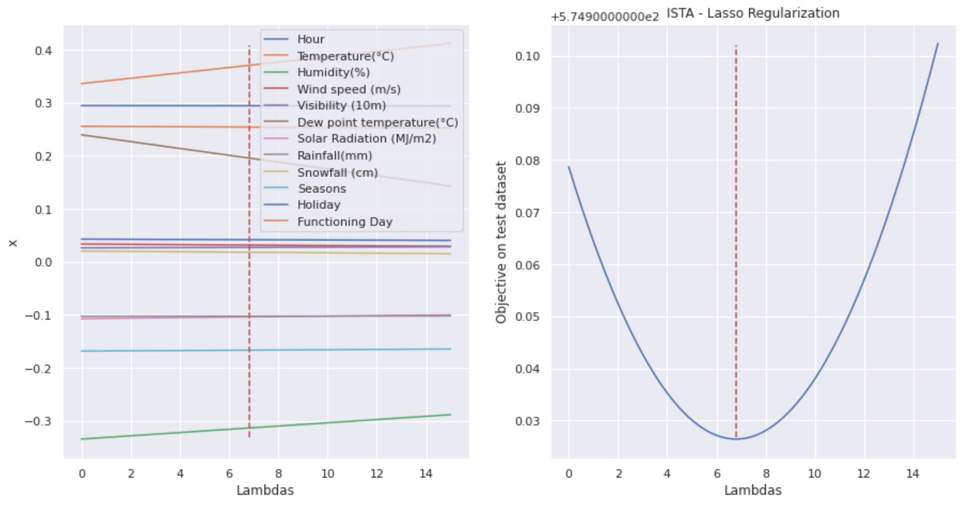
But we can't evaluate the best λ as for Ridge Regularization because of the L1-norm used for the regularized term. The trick to solve Lasso Regularization is the using ISTA algorithm. The ISTA algorithm works like the Gradient Descent but with a different update for x_k.

$$x_{k+1} = S^1_{\lambda\tau}(x_k - \tau\nabla f(x_k)) \tag{17}$$

where $S^1_s(x)$ is the Soft Thresholding defined as:

$$S_s^1(x) = max(|x| - s, 0).sign(x) \tag{18}$$

To find the best λ, we compute ISTA with different values of λ as for Ridge Regularization (compute on train set and evaluate on test set). For our dataset, we obtained $\lambda = 6.8$. As with Ridge Regularization, Lasso Regularization is also used to select the best features in a dataset. Our Lasso Regularization keep all the features, none is equal to 0.



References

[1] Clément W. Royer (2021) *Lecture notes on advanced gradient descent.*

[2] Clément W. Royer (2021) *Lecture notes on stochastic gradient methods.*

[3] Gabriel Peyré (2021) *Regularization, Lasso and ISTA lectures.*