Assignment #3

Course: *Machine learning*
Date: *October 27th, 2024*

Assignment

In this assignment, you will learn about regularization methods. You will also implement the ridge regularization method using gradient descent and stochastic gradient descent and test it on "Wine quality" data.
Be careful not to use the same data for training (any stage of training) and testing your model.

Use the "Communities and Crime" dataset you used in the previous assignment with the same preprocessing.

The last column of the dataset (ViolentCrimesPerPop) is your target variable. Remove the attributes state, county, community, community name, and fold (columns 1 to 5).
`https://archive.ics.uci.edu/dataset/183/communities+and+crime`

Fit models using ridge and lasso regression. Try different values of the regularization parameter and evaluate its effect. Choose the optimal regularization parameter and discribe how you did it.

Use the ridge and lasso functions from Scikit-learn, and for the selection of the regularization parameter. You can also use functions that help you search the parameter space.

Compare the attributes selected with forward attribute selection (the results from the previous assignment) with the attributes lasso selected.

→ Compare results, are they the same, ok

Download the "Wine quality" dataset. Choose only the white wine data. Prepare your data for modeling.

The data is available at `https://archive.ics.uci.edu/dataset/186/wine+quality`.
You will be doing a regression of the wine quality grades (last column).

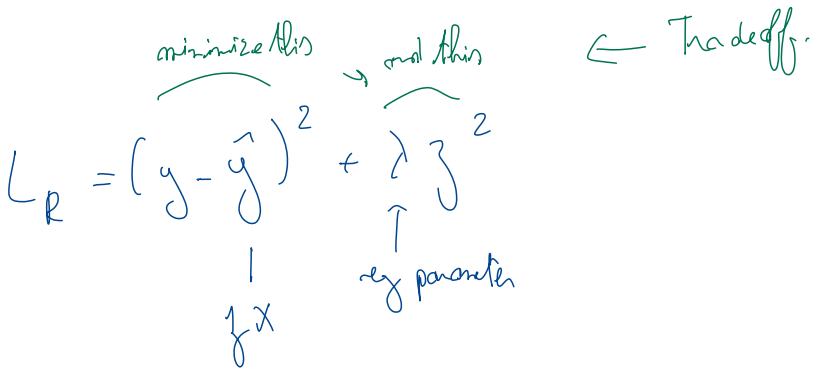
Implement ridge regression with:
- gradient descent
- stochastic gradient descent

and test it on the "Wine quality" data.

It is expected that the gradient descent is implemented from scratch.

Test different learning rates and try to find the optimal one.

Compare the time to convergence and the results of the two gradient descents you implemented.



Example

DATASET

x	y
1	2
2	3
3	4

$$y = \beta + \beta x \qquad x = \beta X$$

$$MSE = \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_i) \right)^2$$

$$\beta_1 := \beta_1 - \alpha \frac{\partial MSE}{\partial \beta_1}$$

$$\beta_0 := \beta_0 - \alpha \frac{\partial MSE}{\partial \beta_0}$$

Initialize $\beta_0, \beta_1 = 0$ ex

$\alpha :=$ learning rate $\begin{cases} \sim 0.1 \\ \sim 0.01 \end{cases}$

↳ checker Metropolis Hasting pour ce pb d'optimisation

$$\frac{\partial MSE}{\partial \beta_1} = \frac{1}{2 \cdot 3} \sum_{i=1}^{3} (-x_i) \left(y_i - (\beta_0 + \beta_1 x_i) \right) = \dots = -\frac{1}{3} \cdot 20 = -6.67$$

$$\frac{\partial MSE}{\partial \beta_0} = -\frac{1}{3} \sum_{i=1}^{3} \left(y_i - (\beta_0 + \beta_1 x_i) \right) = \dots = -\frac{1}{3}(2+3+4)$$

$$= -\frac{1}{3} \cdot 9 = -3$$

SGD

↳ we use randomly one data in the data set at each step instead of the whole data set for 1 epoch

↳ then we take the the data set, we shuffle it for the next epoch.

↳ GD is good but it doesn't give the perfect solution ⇒ prevent overfitting

SGD advantages: