IMC-4302C – Statistical learning
1. Linear regression and stochastic gradient descent

Kevin Zagalo

<kevin.zagalo@inria.fr>

January 13, 2020

Use a `Jupyter` notebook and send it to the depository before the next session.
Any notebook sent after that won't be considered. Thank you !

Exercise 1 *Gradient descent*

We want to minimize a function $F : \mathbf{R}^d \to \mathbf{R}$, differential on $\mathbf{R}^d$. Let ∇F be its gradient. The gradient descent corresponds to the iterative algorithm :

— **Initialization** $x_0 \in \mathbf{R}^d$

— **Iteration** $x_{k+1} = x_k - \alpha \nabla F(x_k)$

The iteration is repeated until a stopping criterion is reached.

Question 1

Suggest some criterion to stop the algorithm.

Question 2

Implement this algorithm in Python. We will define a function which takes as input the function F and its gradient ∇F and others.

Question 3

What happens if F is not convex ?

Question 4

Discuss the influence of the x_0 point, in the non convex case, then in the convex case.

■

Exercise 2 *Least squared method*

A linear regression with n inputs $x_1, \ldots, x_n \in \mathbf{R}^d$ and an output defined by $d+1$ constants : the weights parameters $w_1, \ldots, w_d$ and a bias b. We define the regression function $h_w(x) = \langle x, w \rangle - b$.

We define the local and global error, respectively :

$$e(x; w) = \frac{1}{2}(y_x - h_w(x))^2 \; ; \; E(x_1, \ldots, x_n; w) = \frac{1}{n}\sum_{i=1}^{n} e(x_i; w) \tag{1}$$

To estimate the final weight parameters β, we want to minimize the global error.

We want to implement the algorithm :

— **Input** $X = (x_1, \ldots, x_n)$ and the associated responses $(y_1, \ldots, y_n)$, and $\epsilon > 0$ a parameter.

— $t = 0$

— $w(t) = \vec{0}$

— **Repeat**

  — **Compute** $L(w) = E(x_1, \ldots, x_n; w)$

  — **Update** $w(t+1) = w(t) - \epsilon \nabla L(w(t))$

Until all data is explored and *convergence* of the weights sequence to β.

Question 1

We treated the bias b as a parameter. Show that we can consider it as a weight parameter. How to adapt the problem of regression ?

Question 2

Write a function taking the weights parameters, an observation and its associated response that updates the weight parameters.

Question 3

Implement the algorithm.

Question 4

Plot the global error over the iterations using `matplotlib`.

Question 5

Modify the algorithm by implementing SGD and plot the global error over the iterations.

Question 6

Compare execution times of GD and SGD with the library `time`.

■

Exercise 3

Download **the *house.csv* file** containing 3 columns that represent the area, the number of rooms and the price of 600 houses (one per row). Comment every result.

Part 1 : Linear regression with 1 feature (house area)

In this first part, we will train a linear model for house price prediction using only one feature the house area. We will start by implementing a cost function and the gradient of this cost function. Then, we will implement the gradient descent algorithm that minimizes this cost function and determine the linear model parameter θ in the equation $h_\theta(x) = \theta_1 x$.

Question 1

Open this file with a file editor to understand more the data. Load the data in a `house_data` variable and check its size.

> **Hint:** You could use loadtxt function from the `numpy` library.

Question 2

Extract m – the size of the sample and n – the number of features. Extract the house area and price columns respectively in X and y arrays to visualize them.

Hint: The shape of X and y arrays should be $(m, 1)$ for the following questions and not $(m,)$. You could use newaxis `numpy` object to add a new axis of length one.

Question 3

Implement the `cost_func` function that evaluate and return the previous equation of the mean squared error (1).

Question 4

Implement the `grad_cost_func` function that evaluates the gradient of the cost function at the point theta considering the j^{th} component as given on the previous equation.

Question 5

Implement the `grad_descent` algorithm that updates, iteratively, the parameter vector θ according to the previous equation.

Call the `grad_descent` function to compute θ_{opt}. You could use `max_iteration` of 1000 and α equal to 0.0001.

Use the calculated θ_{opt} to estimate the price of a house with $330m^2$ area.

Part 2 : Linear regression with 2 features (house area + bias term)

In this part, we will train a linear model for house price prediction using the house area and the bias term that represents the constant term (y-intercept) in the linear model equation $h_\theta(x) = \theta_1 x + \theta_0$.

Question 6

– Build the matrix X with shape $(m, 2)$ that represents 2 features: a column of ones that represents the bias term and a column of house area. (Use `numpy`'s concatenate function)

– Change the value of n to be equal to the number of features (number of columns of matrix X equal to 2 in this example).

– Make all needed modification in `cost_func`, `grad_cost_func` and `grad_descent` functions if your implementation was not generalizable for any number of features n.

Question 7

Use the calculated θ_{opt} of the new model to estimate the price of a house with $330m^2$ area.

Question 8

Try to normalize on the house area feature to enhance the convergence of the model. You should modify the X matrix in the previous code block and re-execute the code. What do you notice?

Part 3 : Linear regression with 3 features (house area + number of rooms + bias term)

In this part, we will train a linear model for house price prediction using the house area, number of rooms and the bias term that represents the constant term in the linear model equation $h_\theta(x) = \theta_2 x_2 + \theta_1 x_1 + \theta_0$.

Question 9

– Build the matrix X with shape $(m, 3)$ that represents 3 features: a column of ones that represents the bias term, a column of house area and a column of number of rooms.

– Change the value of n to be equal to the number of features (number of columns of matrix X equal to 3 in this example).

– Use the calculated θ_{opt} of the new model to estimate the price of a house with $330m^2$ area and 5 rooms. Compared to the previous model which predict better house prices ?

Bonus: You could also try to add other feature columns to the matrix X like $\texttt{area}^2$ or $\sqrt{\texttt{area}}\ldots$ and see the effect on the model and the error.

■