# IMC-4302C – Statistical learning

# 1. Linear regression and stochastic gradient descent

Kevin Zagalo

<kevin.zagalo@inria.fr>

2019 – 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 $\nabla F$ 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 $\nabla F$ 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 $w^*$, 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 $\beta$.

**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.

### Question 1

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

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

### Question 2

Extract the house area and price columns respectively in $X$ and $y$ lists. Scatter prices against areas.

### Question 3

The cost function we will use for this linear model training is the **Mean Squared Error** function defined by

$$L(w) = MSE(X \cdot w, y) = \frac{1}{2n} \sum_{i=1}^{n} (x_i \cdot w - y_i)^2$$

First of all, transform $X$ and $y$ to be respectively $(1, n)$ and $(n, 1)$-numpy arrays.

**Question 4**

Implement the mean squared error cost function. Then implement its gradient :

$$\nabla L(w) = \partial_w MSE(X \cdot w, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i \cdot w - y)\ x_i$$

**Question 5**

The update equation of the gradient descent algorithm is given by:

$$w^{(t+1)} = w^{(t)} - \alpha \nabla L(w^{(t)})$$

Where $\alpha$ represents the step or the **learning rate**. Compute three steps of the gradient descent with different values of $\alpha$. For each $\alpha$, plot $w \to L(w)$, your initial point $(w_0, L(w_0))$ and the three points associated to $w_1$, $w_2$ and $w_3$ you've just computed. What the best $\alpha$ according to you ?

**Question 6**

Implement the gradient descent algorithm, taking as input the gradient function, the learning rate, one or several stopping criterions and an initial parameter $w_0$.

**Question 7**

Compute the optimal parameter $w^*$. Compare on the same frame the actual prices and the ones predicted by the linear model against the areas.

**Question 8**

Introduce the bias term.

**Question 9**

Same as question 7.

**Question 10**

Add the number of rooms in $X$. Same as question 8.

*Question 11 You could also add other features to the matrix $X$ like $area^{0.5}$. Is it still linear ?

### Question 12

Create a `LinearRegression` class object as follows :

```python
class LinearRegression:

    def __init__(self, alpha=1.0, tol=1e-3, max_iter=None, normalize=False):
        """
        Initalizes the hyper parameters of the model
        alpha : Regularization strength; must be a positive float.
        max_iter : Number of steps of gradient descent.
        normalize : Whether to normalize features or not.
        solver : Gradient descent or Stochastic gradient descent
        """
        pass

    def fit(self, X, y):
        """
        Fit the linear model
        X : {ndarray, sparse matrix} of shape (n_samples, n_features)
        y : ndarray of shape (n_samples,) or (n_samples, n_targets)
        """
        pass


    def predict(self, X):
        """
        Predict using the linear model
        X : array_like or sparse matrix, shape (n_samples, n_features)
        """
        pass
```

### Question 13

How much would cost a $330m^2$ flat with 5 rooms ?

■