Practical work 05 – 16.10.18
Supervised learning – System Design and Debugging

Summary for the organisation :

- Submit the solutions of the practical work before Monday 12h00 next week in Moodle.
- Preferred modality : archive with iPython notebook(s).
- Alternative modality : pdf report with annotated code and outputs.
- The file name must contain the number of the practical work, followed by the names of the team members by alphabetical order, for example `02_dupont_muller_smith.zip`.
- Put also the name of the team members in the body of the notebook (or report).
- Only one submission per team.

Exercice 1 Gradient descent using matrix calculation

```python
def hypothesis(theta,X):  #theta = 1xD, X = DxN, output 1xN
    return np.dot(theta,X)

def gradientDescent(X,y,learning_rate,num_epoch,verbose=False):
    N = X.shape[0]      # number of sample
    D = X.shape[1]      # number of dimensions
    theta = np.ones(D)  # init thetas to some values
    X_trans = X.transpose() # X_trans is DxN

    for i in range(0,num_epoch):
        h = hypothesis(theta,X_trans)  #N dimension
        loss = h-y                     #N dimension
        gradient = X_trans.dot(loss) * (1.0/N)
        theta = theta - learning_rate * (1.0/N) * gradient
    return theta
```

FIGURE 1 – Gradient descent

a) Analyse the code above. It relies on matrix calculation. What type of gradient descent does it implement ? Are you confident it is a correct implementation ?

b) Let's use the data from last week exercise 'lausanne-appart.xlsx' from moodle. The code above should help us to redo exercise 6b on multi-variable inputs. To use the code above, you first need to build the design matrix X as follow :

$$X = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,D} \\ 1 & & \ddots & \\ 1 & \vdots & x_{n,d} & \vdots \\ 1 & & & \ddots \\ 1 & x_{N,1} & \dots & x_{N,D} \end{pmatrix} \tag{1}$$

with living area in the second column and number of rooms in the third column. Plot the plane $h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ on top of the 3d scatter plot.

c) You may find difficulties to make the algorithm converge on the dimension of the number of rooms. We may have a normalisation problem here as the number of rooms is very small in comparison to the living area. Implement the zero-norm normalisation and apply it to the number of rooms and living area. Re-run the algorithm of point b. Does it help for the convergence ?

Exercice 2 Linear regression optimisation

We would like you to find the order O of the polynomial function that models the best the data illustrated in Figure 2.

$$h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x + +\theta_2 x^2 + \ldots + \theta_D x^O \tag{2}$$

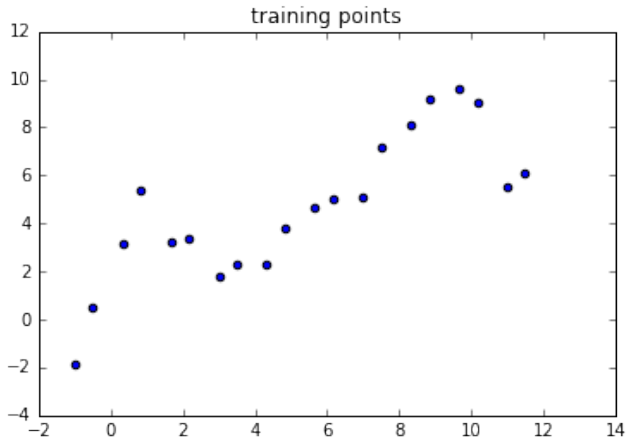

FIGURE 2 – Training data for linear regression optimization

The data set is split into a training set and a cross-validation set. You will find the data in the file `overfitting.xlsx` on moodle. The data has been split for you into a training set and a cross-validation set (in two different sheets).

a) Read the data in separate variables for the training and cross-validation sets.

b) Define a cost function $J(\theta)$ that will allow you to compute the cost on the training and cross-validation sets.

$$J_{train}(\theta) = \frac{1}{2N_{train}} \sum_{n=1}^{N_{train}} (h_\theta(\mathbf{x}_n^{train}) - y_n^{train})^2 \tag{3}$$

$$J_{cv}(\theta) = \frac{1}{2N_{cv}} \sum_{n=1}^{N_{cv}} (h_\theta(\mathbf{x}_n^{cv}) - y_n^{cv})^2 \tag{4}$$

c) Perform the training for increasing orders $O = 1, \ldots, 7$. You can use any training method developed in the previous practical work PW2 (using the *normal* equations is probably the easiest choice).

d) Plot the trained hypothesis. You should have something similar to the next figure.

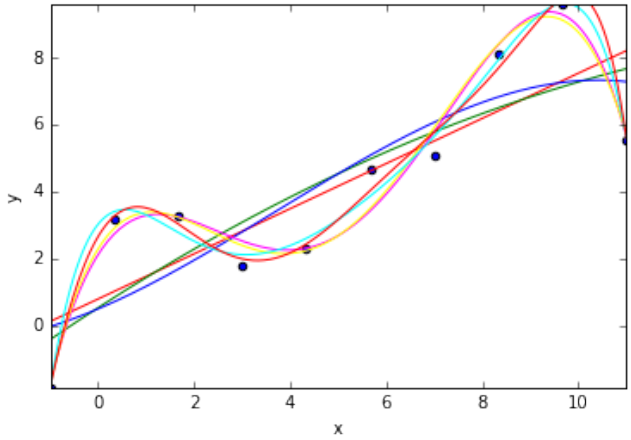

FIGURE 3 – Trained models for $D = 1, \ldots, 7$.

e) Plot the evolution of the costs $J_{train}(\theta)$ and $J_{cv}(\theta)$ as a function of the order O.

f) What is your best model according to these costs ? Comment your answer.