Computer Vision

SSI Pattern Recognition

Practical

Désiré Sidibé
dro-desire.sidibe@u-bourgogne.fr

Linear Regression[1]

I. Introduction

In this exercise, we will implement linear regression with one and multiple variables, as well as gradient descent.

II. Linear regression with one variable

Suppose you are the CEO of a restaurant franchise and are considering different cities for opening a new food truck. The chain already has trucks in various cities and you have data for profits and populations from the cities.
You would like to use this data to help you select which city to expand to next. The file `lab1data1.txt` contains the dataset: the first column is the poplulation of a city and the second column is the profit of a food truck in that city. An negative value for profit indicates a loss.

1. Before starting any task, it is always useful to understand the data by visualizing it. Write a function `plotData(X, y)` that shows the data as in Figure 1.

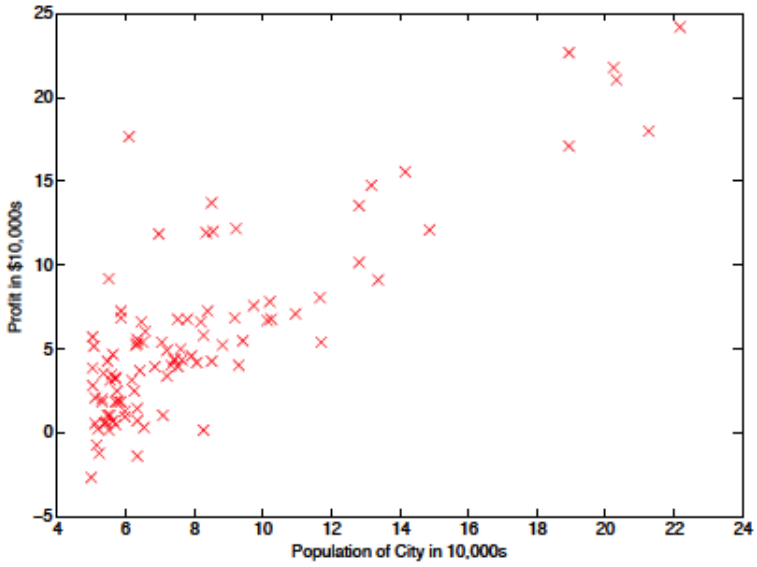

Figure 1: Scatter plot of training data using *plotData(X,y)*.

[1] Updated from Andrew Ng.

2. Assume a linear model of the form $y(x) = w_0 + w_1 x$, where x is the input variable (here the city population) and y is the output variable (here the profit).

   Write a function `w = LinearReg(X, y)` that estimates the parameters $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$ using the training data in `lab1data1.txt`.

   Note that you should use the matrix notation $y(\mathbf{x}) = w^T \mathbf{x}$, where $\mathbf{x} = \begin{bmatrix} 1 \\ x \end{bmatrix}$. So you should add one dimension to each input variable to take care of the bias term!

3. Use your estimated parameters to make predictions of profits in cities of 35,000 and 70,000 people.

   If everything is correct, you should find 2798.3687 and 44554.5463.

III. Linear regression with multiple variables

Suppose you are selling your house and you want to know what a good market price would be. One way to do this is to collect information on recent houses sold and make a model of housing prices.
The file `lab1data2.txt` contains a training set of housing prices in a city. The first column is the size of the house (in square feet), the second column is the number of bedrooms and the third column is the price of the house.

1. Assume a linear model of the form $y(x) = w_0 + w_1 x_1 + w_2 x_2$, where x_1 and x_2 are the size of the house and the number of bedrooms, respectivelly.

   Adapt your function `LinearReg` that estimates the parameters $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$ using the training data in `lab1data2.txt`.

   Note that you function sould be able to deal with data of any dimension!

2. By looking at the data, we see that house sizes are about 1000 times the number of bedrooms. When features differ by order of magnitude, it is important to perform feature *scaling* or *normalization*.

   - Write a function `[Xn,mu,sigma]=featureNormalize(X)` that subtracts the mean of each feature from the dataset, and scale the feature values by their respective standard deviation:

     $$\text{for every feature } f_i, \text{ do } \; f_i \leftarrow \frac{f_i - \bar{f}_i}{\sigma_{f_i}}.$$

     This way, each feature is now centered (zeor mean) and standardized (unit variance).

     Note that the added feature (i.e. column of ones to take care of the bias) should not be normalized!

3. Estimate again the parameters after feature normalization. Do you find a different result? Is it better?

4. Use your estimated parameters to predict the price of a house with 1650 square feet and 3 bedrooms.

   Careful: you have to normalize the new test data as well, using the same procedure!

IV. Gradient descent

We will now perform linear regression using gradient descent.
We define the following cost function

$$J(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)^2.$$

1. Show that

$$\forall i, \qquad \frac{\partial}{\partial \mathbf{w}_i}J(\mathbf{w}) = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{w}^T\mathbf{x}_k - y_k)\mathbf{x}_k^{(i)},$$

   where $\mathbf{x}_k^{(i)}$ is the i-th element of the vector $\mathbf{x}_k$.

2. Write a function `[w]=GradientDescent(X,y,w,alpha,NIter)` that implement the following algorithm:

   - Initialize $\mathbf{w}$ (with zeros for example)
   - Iterate until convergence (or for a fixed number of iterations)

$$\mathbf{w}_i = \mathbf{w}_i - \alpha\frac{\partial}{\partial \mathbf{w}_i}J(\mathbf{w})$$

     (simultaneously update $\mathbf{w}_i$ for all i).

   - You will need to try different learning rates α. To pick the best, you need to run gradient descent for about 50 iterations for each learning rate, and return the history of $J(\mathbf{w})$ values in a vector. You can then plot the $J(\mathbf{w})$ values against the number of iterations for each learning rate, and select the learning for which the algorithm converges quickly.

     We recommend trying values of α on a log-scale, i.e. 0.3, 0.2, 0.03, 0.01, 0.003, 0.001, etc.

     You way also want to adjust the number of iterartions you are running if that will help you see the overall trend in the curve.

   - Note that the features need to be normalized when applying gradient descent.

3. Using your best learning rate and the final vector $\mathbf{w}$, predict the price of a house with 1650 square feet and 3 bedrooms.

   You should find the same predicted value as in part III.