Linear Regression

Linear regression is a statistical method for modeling the relationship between a dependent variable y and one or more independent variables x. It is used to make predictions in machine learning.

Let's start with **univariable regression**. Assume we have a dataset named Housing.csv and it contains two parameters – lot size and price. Here we want to use linear regression to figure out the quantitative relationship between lot size and price, and make future predictions on prices of houses with known lot size.

Here lot size is defined as the independent variable x and price is defined as the dependent variable y. We aim to figure out the optimum parameters θ_0 and θ_1 so that the predicted dependent variable $\boldsymbol{h}_\theta$ is as close to the real y as possible. Here

$$\boldsymbol{h}_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 \boldsymbol{x}$$

Quantitatively, we want to minimize cost function defined as follows:

$$\boldsymbol{J}(\theta_{0,}\theta_1) = \frac{1}{2\boldsymbol{m}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}} (\boldsymbol{h}_\theta(\boldsymbol{x}^{(\boldsymbol{i})}) - \boldsymbol{y}^{(\boldsymbol{i})})^2$$

Here $(\boldsymbol{x}^{(\boldsymbol{i})}, \boldsymbol{y}^{(\boldsymbol{i})})$ is the *i*th (lot size, price) pair from the dataset and m is the number of (lot size, price) pair in the dataset.

We use gradient descent to optimize θ_0, θ_1 in order to minimize the cost function $\boldsymbol{J}(\theta_0, \theta_1)$. Gradient descent is a 1st order iterative optimization algorithm used to find minimum value of any function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point. That is, to update θ as follows:

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial \boldsymbol{J}}{\partial \theta_j}$$

Here, we have

$$\frac{\partial \boldsymbol{J}}{\partial \theta_0} = \frac{1}{\boldsymbol{m}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}} (\boldsymbol{h}_\theta(\boldsymbol{x}^{(\boldsymbol{i})}) - \boldsymbol{y}^{(\boldsymbol{i})})$$

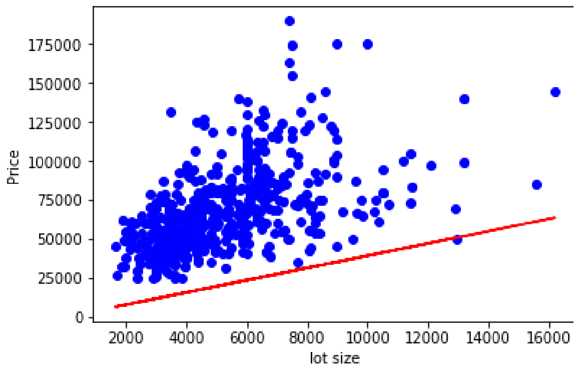
$$\frac{\partial \boldsymbol{J}}{\partial \theta_1} = \frac{1}{\boldsymbol{m}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{m}} (\boldsymbol{h}_\theta(\boldsymbol{x}^{(\boldsymbol{i})}) - \boldsymbol{y}^{(\boldsymbol{i})})\boldsymbol{x}^{(\boldsymbol{i})}$$

Below is the pseudo code:

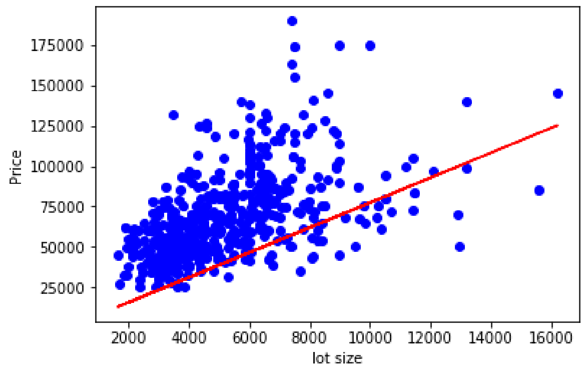
```
repeat until convergence {
    $\theta_0 := \theta_0 - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$
    $\theta_1 := \theta_1 - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$
}
```

α is the learning rate, which determines how fast the gradient descent is. If α is too small, the optimization of θ_0, θ_1 will be too slow. If α is too big, the gradient descent step might be too big so that it can overshoot the minimum, making the resulting J value not converge or even diverge. Empirically, we try to use α as 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1 etc.

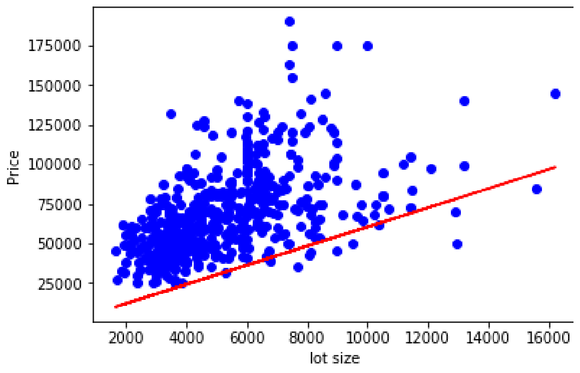
For example, if we use linear regression to fit the Housing.csv data, we need to carefully tune the learning rate to avoid divergence of the cost function. In fact, we have to decrease the learning rate from 0.1 to as small as 1e-10 to get the cost function converge (see Python file for code). The red line in the following graphs shows how the fitted line $h_\theta(x) = \theta_0 + \theta_1 x$ change along with increasing iteration number.



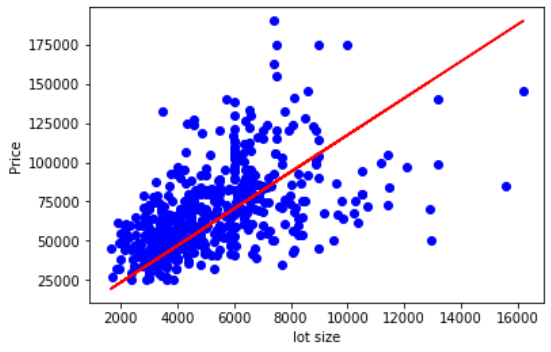
200th iteration



100th iteration

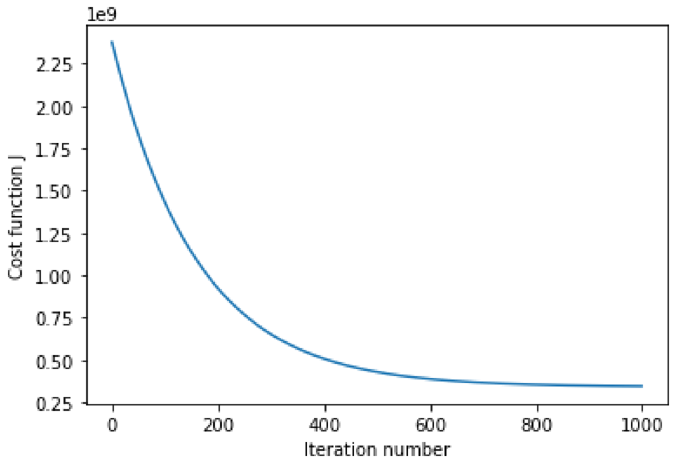


300th iteration



1000th iteration

The figure below shows how the cost function J change with increasing iteration number.



The linear regression algorithm can also handle **multivariate regression** with the form of

$$h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

Similarly, θ values can be optimized by gradient descent.

If $x_1 = x, x_2 = x^2, x_3 = x^3, x_4 = x^4, \ldots.$, then the linear regression algorithm can be used to handle **polynomial regression**.

Reference:

1, Machine Learning by Andrew Ng, from Coursera
2, https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html