Univariate Linear Regression

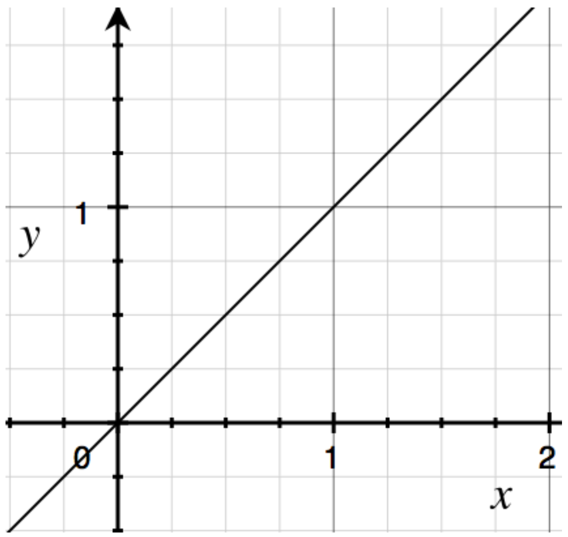
Gordon Gao at 2016.09.01

Let's start with an example. In house price prediction, we may have some information of each house such as size and the price which we would like to predict later based on other factors like size. We call the data set, size and price, is our training set. One of our friend Joe wants to buy a new house, and he wants to know what is the appropriate price of the house in the market.

In order to predict the correct price, we need to combine several machine learning algorithms or functions together.

The first function is called hypothesis function. It takes the x as an input, and, by calculation, estimate the value of y.

$$h_\theta(x) = \theta_0 + \theta_1 x$$



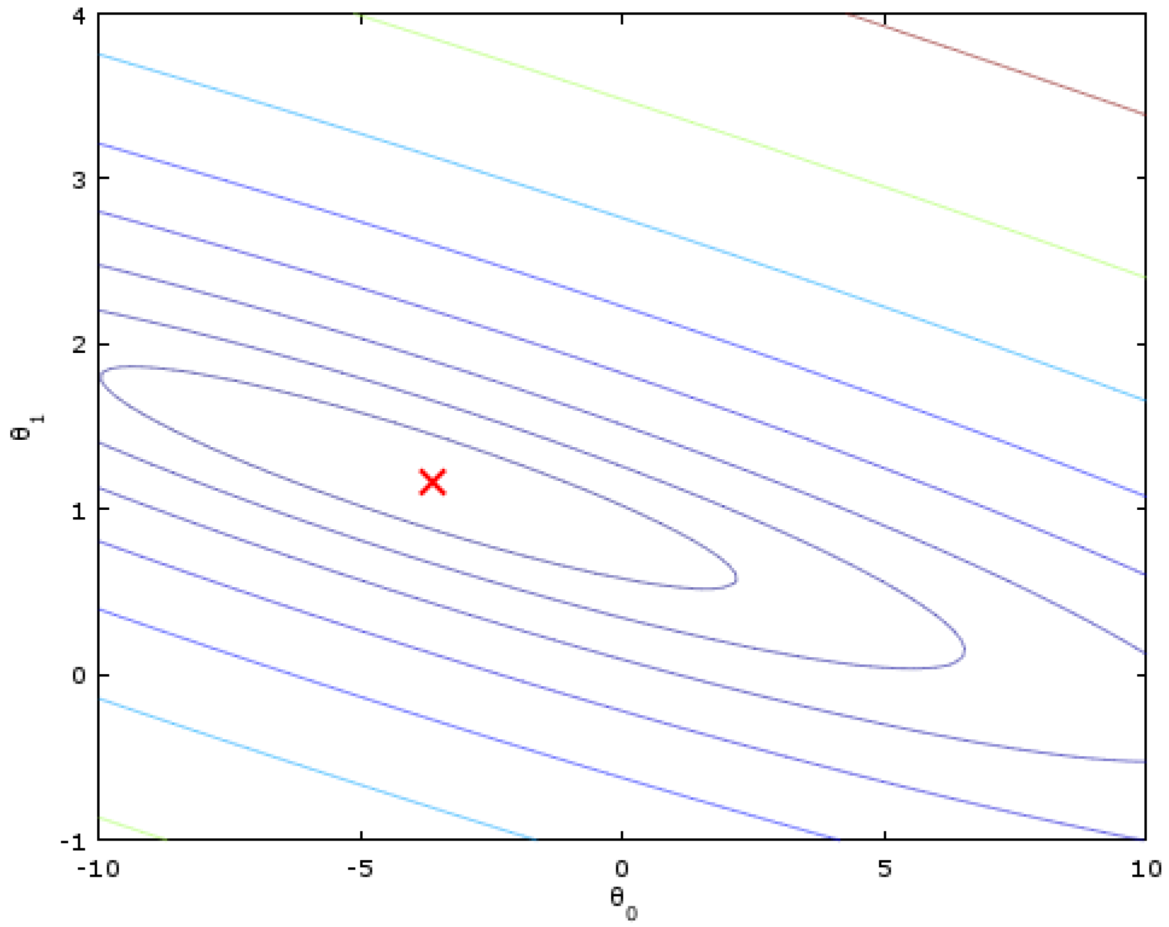
The model is called **Univariate Linear Regression**.

The second function is the cost function which recursively calculates the θ_i that corresponds to a good fit to our training set. In other words, we want to minimize (θ_0, θ_1) through the training.

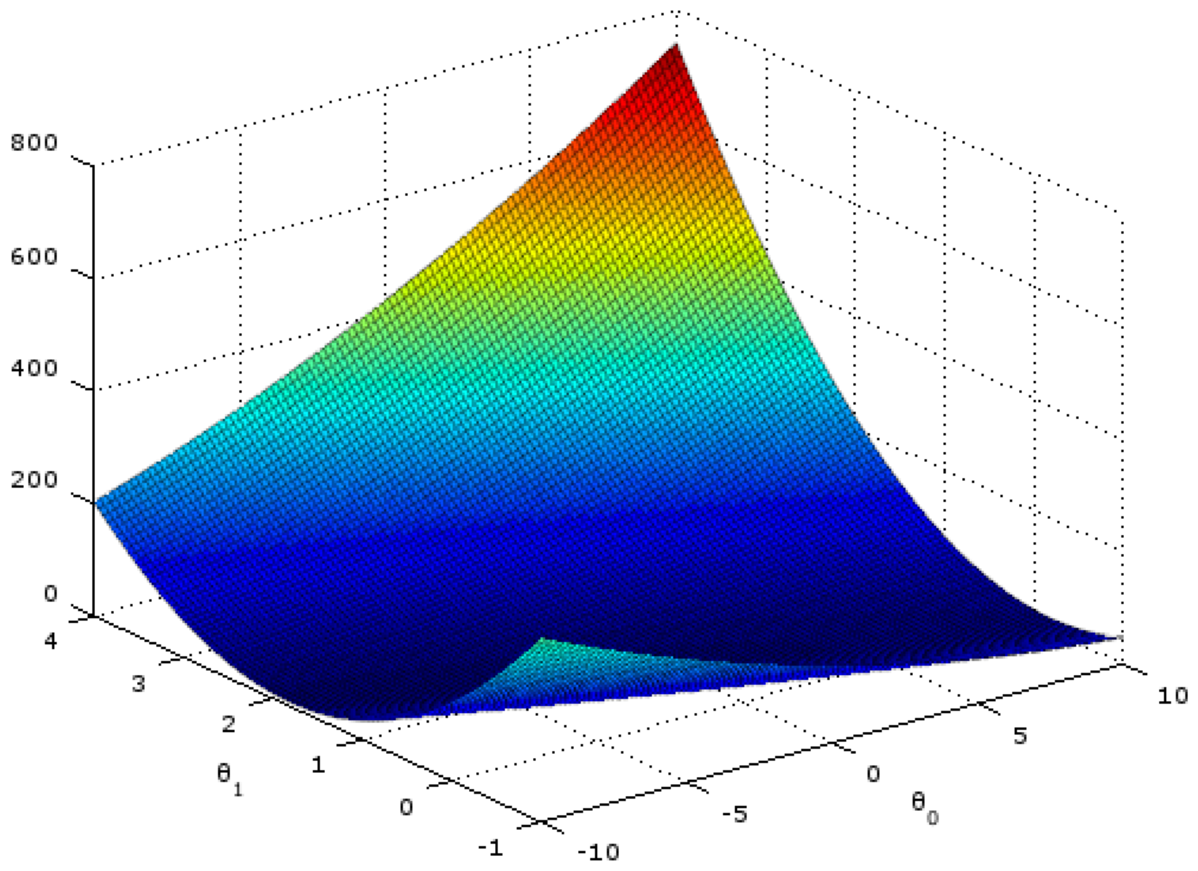
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2$$

This function is also known as **Square Error Cost Function**.

Based on our training set, we can visualize the Cost Function. I utilize Octave to generate these figures.
Surface plot:



Contour plot:



A question comes in. How can we find the minimum $J(\theta_0, \theta_1)$?

The way to find the minimum value of $J(\theta_0, \theta_1)$ is to check the point's slope and keep changing the (θ_0, θ_1) with an algorithm called Gradient Descent. Be ware of the local optimum value.

```
repeat until convergence {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$   (simultaneous update)
}
```

For j=0 and j=1:

$$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := temp0$$

$\theta_1 := temp1$

α is the learning rate. If it is too small, the training time will be very long. If it is too large, the training may fail to converge.

Let’s spread out the algorithm.

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

And that’s it! After the calculation, we will get the appropriate (θ_0, θ_1) .