Note: Linear Regression Model

Sun Zhao

January 10, 2013

1 Model Representation

Remind that regression algorithms are tying to infer a function predicting continuous values when given training samples. Let's first focus on the notation of the training samples. Regarding to Table 1, training samples are represented by a vector of $\mathbf{x}$ and a vector of $\mathbf{y}$ with identical length of m and each sample becomes a pair of $(x^{(i)}, y^{(i)})$. The inferred function h(which means hypothesis) is shown as below:

Table 1:

Notation	Meaning
m	number of training examples
$\mathbf{x}$	input variables/features
$\mathbf{y}$	output variables

$$h_\Theta(x) = \Theta_0 + \Theta_1 x \tag{1}$$

For simplicity, (1) consists of only one variable, thus we call this linear regression as univariate linear regression.

2 Cost Function

Θ_0 and Θ_1 are the two arguments of hypothesis function, so the idea here is to choose optimal values of Θ_0 and Θ_1. Intuitively, the optimal h should minimize the average error distance between the predict values of each $x^{(i)}$ and $y^{(i)}$. Then we can define the cost function as below:

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2 \tag{2}$$

Obviously, the object function is defined as:

$$\underset{\Theta_0, \Theta_1}{minimize} \frac{1}{2m} \sum_{1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2 \tag{3}$$

Using the cost function, we can easily calculate the cost value of any hypothesis functions. For example, Fig. 1 shows three samples(cross pointers) and a hypothesis function $h(x) = 0.5x$. The cost value of it is calculated as:

$$J(0.5) = \frac{1}{2 \times 3} [(0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2] = \frac{3.5}{6} \tag{4}$$

$$h_\theta(x)$$
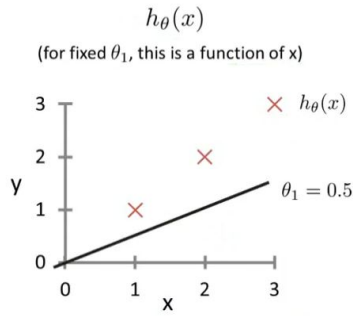
(for fixed θ_1, this is a function of x)

Figure 1:

3 Gradient Descent

Now we turn to the calculation methods of the objection function. Suppose we have a training set as shown in Fig. 3. We can enumerate all the possible values of Θ_0 and Θ_1 and calculate the corresponding cost $J(\Theta_0, \Theta_1)$ and show them in a 3D space like Fig. 2. In Fig. 2, if we start from a random pointer$(\Theta_0, \Theta_1, J(\Theta_0, \Theta_1))$, we can iteratively migrate a small step through the negative direction of the gradient direction of this point and hopefully, we will converge to the lowest point of the surface.
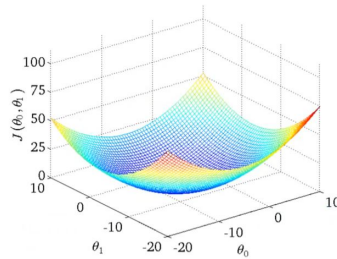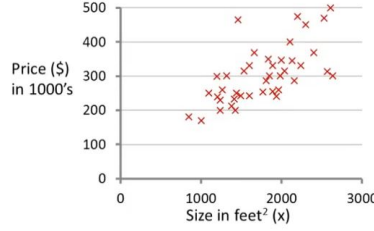

Figure 2:

Figure 3:

Formally, we describe gradient descent algorithm as following:

Algorithm1:

$\Theta_0 = 0$;
$\Theta_1 = 0$;
repeat until convergence{
$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) for (j = 0 and j = 1)$
}

Issue1: α is called the learning rate. It is a pre-choosing value, and if it is too small, the convergence process will be much slow, otherwise it is too large, it can not converge and possibly diverge. As the algorithm approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.

Issue2: The update process should be a simultaneous update which should be performed as:
$temp0 = \Theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$temp1 = \Theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_0 = temp0$
$\theta_1 = temp1$
If we do not take a simultaneous update, the output of gradient descent algorithm is not defined.

Issue3: Gradient Descent algorithm starts from a random point, so the output is a local optima instead of global optima. However, if the object function is a convex function, it is guaranteed that gradient descent algorithm will reach the global optima.

Getting back to the linear regression problem, we use gradient descent to solve the problem of (3). If we apply (1) to Algorithm1 and calculate the differential result, we will get:
repeat until convergence {
$\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$
$\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$
}
Note that we use all the samples to update the arguments in each step, so

4

we called it "batch" gradient descent. Until now, linear regression problem is successfully solved by "batch" gradient descent.

4   Summary

Univariate linear regression model is represented as (1), and its cost function and objection function are shown as (2) and (3) separately. Applying object function with gradient descent algorithm will solve linear regression problem.