# Week01 Basic Concept and Linear Regression(1)

## Basic concept

Many scientists think the best way to make progress on this is through learning algorithms called neural networks, which mimic how the human brain works.
A computer program is said to:

> "Learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

In general, any machine learning problem can be assigned to one of two broad classifications:
Supervised learning and Unsupervised learning.

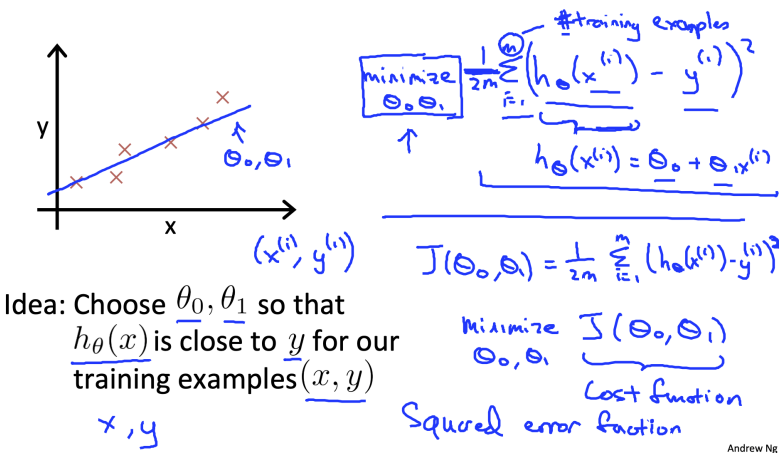
- **Supervised learning** :
  1. Regression problem. eg: house price forecast (quadratic function is better)
  2. Classification problem. eg: malignant tumor judgment 0/1
     *regression problem* could be converted to *classification problem*, such like *the price forecast* is converted into *whether it is sold*.
- **Unsupervised learning** :
  Given a bunch of data without the so-called "correct answer" for automatic classification. Such as *Social network analysis*, *Market segmentation*, *Celestial data analysis*
  1. Clustering problem. eg: genomic classification problems
  2. Non-clustering problem. eg: audio noise process

## Linear Regression

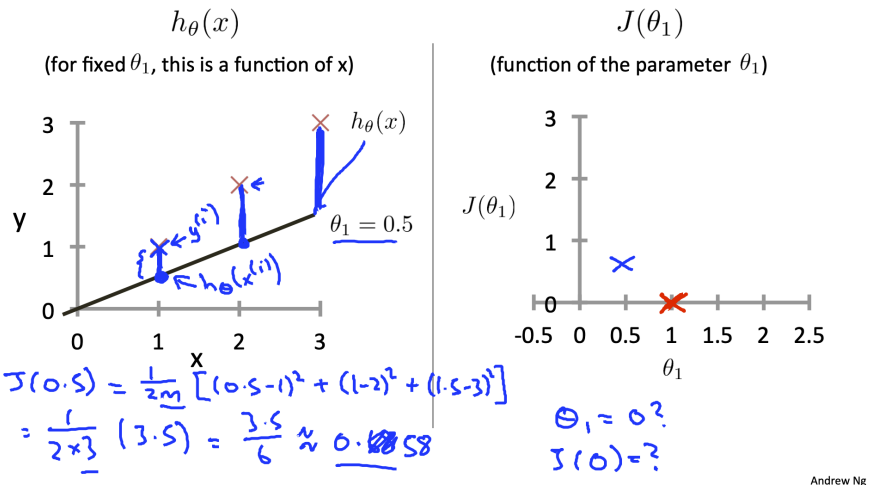
#### Model and Cost Function

To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function $h : X \to Y$ so that $h(x)$ is a "good" predictor for the corresponding value of $y$. For historical reasons, this function $h$ is called a *hypothesis*. Seen pictorially, the process is therefore like this:

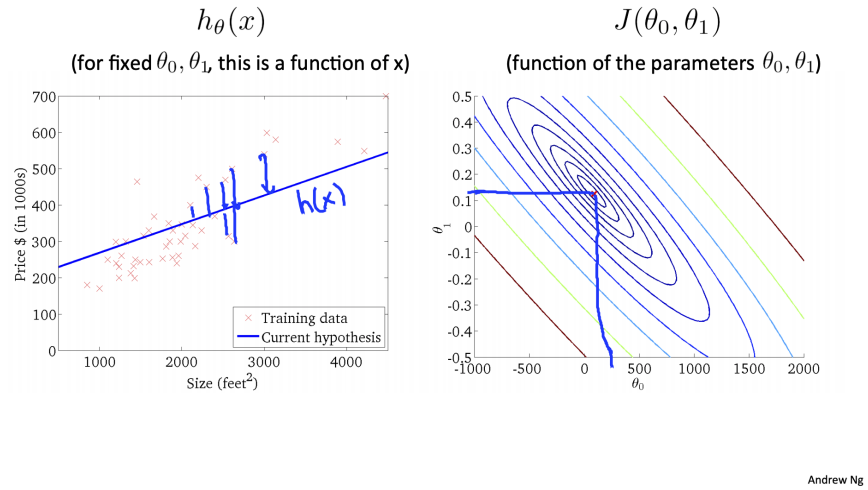
We can measure the accuracy of our hypothesis function by using a **cost function**




If only consider $\theta_1$ where $\theta_0 = 0$, we could draw a $y - x$ line and a two-dimensional $J - \theta_1$ graph:



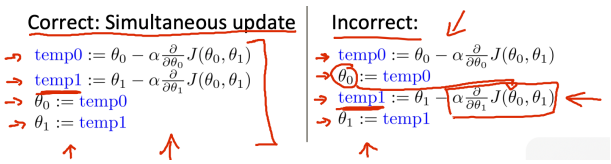

If consider two variable parameters $\theta_0$ and $\theta_1$, it's changed to a three-dimensional graph (or two-dimensional contuor graph):



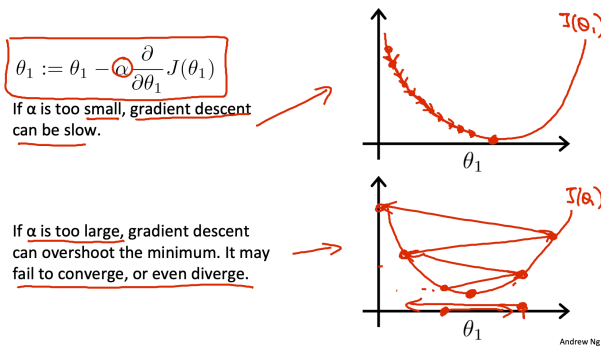

**Parameter Learning of Model**

***Gradient Descent algorithm***

One of the features of Gradient Descent algorithm: Different starting points may get different local optimal solutions.



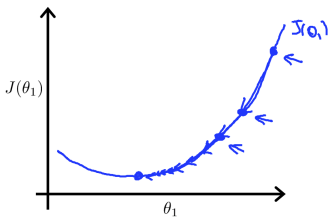

**Note:** Execute simultaneously.
The size of each step is determined by the parameter $\alpha$, which is called the *learning rate*. If $\alpha$ too small or too big, it would cause the convergence time to be too long or unable to converge.



Gradient descent can converge to a local minimum, even with the learning rate α fixed.

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.




Andrew Ng

**Summary**

| Gradient descent algorithm | Linear Regression Model |
|---|---|
| repeat until convergence {<br>$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$<br>(for $j = 1$ and $j = 0$)<br>} | $h_\theta(x) = \theta_0 + \theta_1 x$<br>$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$ |

Integrated linear recursive algorithm:

**Gradient descent algorithm**

$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

}

update $\theta_0$ and $\theta_1$ simultaneously

$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$