MACHINE LEARNING

What is Machine Learning ?

« The field of study that gives computers the **ability to learn** without being explicitly programmed. » *(Arthur Samuel)*

« 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. » *(Tom Mitchell)*

Two broad classifications : Supervised Learning & Unsupervised Learning

Supervised Learning

- We are given a data set and already know what our correct output should look like
- => relationship between the input and the output
- Categorized into *« Regression »* and *« Classification »* Problems :
  - **Regression Problem** : We try to predict results within a **continuous** output
    Ex. : Predict prize of a house given a data set about size of houses
  - **Classification Problem** : We try to predict results in a **discrete** output
    Ex. : Given a patient with a tumor, predict if the tumor is malignant or begnin

Unsupervised Learning

- We approach a problem with little or no idea of what our result should look like
- We derive a structure from data without knowing the effect of the variables
- No feedback based of the prediction results
- *Ex. : Take a collection of 1 millions genes and find a way to group them*

Model representation

$x^{(i)}$: input variable
$y^{(i)}$: output variable
$(x^{(i)}, y^{(i)})$: training set
h : X -> Y
h(x) : *hypothesis*

<u>**Cost Function**</u> *(or « Squared error function/Mean squared error »)*

- Measures the accuracy of the hypothesis function
  => *« How well the hypothesis function fit into a given data »*
- Takes the average difference of all the results of the hypothesis with inputs from x's and the actual output y's.

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

The goal is to minimize the cost function to get the best hypothesis possible

<u>**Gradient Descent**</u>

- Used to estimate the parameters in the hypothesis function
- Ajusts the value of the parameters by minimizing the cost function J.
- The gradient descent algorithm is :

  Repeat until convergence :

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

  – j : *feature index number*
  – α : *learning rate*

<u>/!\</u> The update of each parameters should be simultaneous ! <u>/!\</u>

<u>Correct: Simultaneous update</u>

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

<u>Incorrect:</u>

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \boxed{\alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)}$$
$$\theta_1 := \text{temp1}$$

➔ We want the derivative to reach zero ($\theta_1 := \theta_1 - \alpha * 0$)

➔ If α is too small, Gradient Descent can be very slow and so inefficient

➔ If α is to large, Gradient Descent can overshoot the minimum and may fail to converge, and even diverge

<u>**Gradient Descent for Linear Regression**</u> *(« Batch Gradient Descent »)*

- Better than normal Gradient Descent because it <u>converges to only one global minima</u>
- When specifically applied te the case of Linear Regression, the Gradient Descent equation can be derived. The new algorithm of the Gradient Descent would be :

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$$

}

- m : size of the training set
- We separated θ_0 and θ_1 due to the derivative

Multivariate Linear Regression