What is Machine Learning?

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition: "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."

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

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

Supervised Learing: right answers given

Regression: Predict coutinuous valued output(price)

Classification: Discrete valued output(0 or 1)

Support Vector:

Unsupervised Learning

Can be Used to:

1.Organize computing clusters

2.Social network analysis

3.Market segmentation

4.Astronomical data analysis

Cocktail party problem algorithm

```
[W,s,v] = svd((repmat(sum(x.*x,1),size(x,1),1).*x)*x');
```

Unsupervised Learning

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

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

To break it apart, it is $\frac{1}{2}\ \bar{x}$ where $\bar{x}$ is the mean of the squares of $h_\theta(x_i) - y_i$, or the difference between the predicted value and the actual value.

Gradient Descent

同步更新：simultaneous update

The gradient descent algorithm is:

repeat until convergence {

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

}

where

j=0,1 represents the feature index number.

必须同步更新：

At each iteration j, one should simultaneously update the parameters $\theta_1, \theta_2, \ldots, \theta_n$. Updating a specific parameter prior to calculating another one on the $j^{(th)}$ iteration would yield to a wrong implementation.

α： learning rate 学习速率

α过小：梯度下降过慢

α过大：梯度下降过快，可能直接超过最低点，并最终导致无法收敛（converge）和发散（diverge）

一开始就位于局部最小值（local optimum）的情况：不再下降（接近最低点时自动减速并最终恒定）

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

这里只要记住结论：

When specifically applied to the case of linear regression, a new form of the gradient descent equation can be derived. We can substitute our actual cost function and our actual hypothesis function and modify the equation to :

repeat until convergence: {

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

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

}

where m is the size of the training set, θ_0 a constant that will be changing simultaneously with θ_1 and $x_i, y_i$are values of the given training set (data).

Note that we have separated out the two cases for θ_j into separate equations for θ_0 and θ_1; and that for θ_1 we are multiplying x_i at the end due to the derivative. The following is a derivation of $\frac{\partial}{\partial \theta_j} J(\theta)$ for a single example :

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\
&= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= (h_\theta(x) - y) \, x_j
\end{aligned}$$

代价函数是凸函数（convex quadratic function），因此只有一个全局最优解而没有局部最优解

"Batch" Gradient Descent（批量梯度下降法）：

looks at every example in the entire training set on every step 每一步都用到全部样本（数据量大时优于正规法）

Matrix: Rectangular array of numbers

Dimension of matrix: number of rows x(times) number of columns

Matrix Elements(entries of matrix)

Aij = "i,j entry" in the i(th) row, j(th) column

Vector: An n x 1 matrix

1-indexed vs 0-indexed: start from 1 or 0

Matrix Addition: same dimension

Matrix Vector Multiplication：矩阵列数和向量行数相等，将矩阵每一行的所有元素和对应向量列的所有元素相乘，最后相加，来得到对应行的结果

A x x = y

A: m x n matrix (m rows, n columns)

x: n x 1 matrix (n-dimensional vector)

To get yi, multiply A's i(th) row with elements of vector x and add them up

Matrix Matrix Multipliction：矩阵相乘，将第二个矩阵想像成几个向量的组合，以此类推

矩阵乘法不符合交换律： A x B ≠ B x A（除非B是单位矩阵）

矩阵乘法符合结合律 A x B x C = A x（B x C）= （A x B）x C

Identity Matrix（单位矩阵）：（1，1）到（n，n）都是1，其余都是零的矩阵（方阵）

相当于实数运算中的1，任何数乘以1等于任何数本身

A x I （单位矩阵）= I x A = A

Matrix inverse 矩阵的逆

矩阵乘以它的逆 = 它的逆乘以矩阵（交换律） = 单位矩阵

只有方阵（square matrix）可以有逆

在Octave中可以用pinv(A)求出A的逆（前提是A有逆）

Matrix Transpose 转置矩阵： 行变列 A（ij）= AT（ij）