# Machine Learning Stanford Coursera Online Course

# Equations

## Week 1.

### Linear Regression With Two Variables

$h(x) = \theta_0 + \theta_1 x$

**Function to minimize**

$$\min(J(\theta_0, \theta_1)) = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$$

**Gradient descent algorithm for 2 paramaters $\boldsymbol{\theta_j}$**

*Repeat until convergence {*

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

*}*

*Repeat until convergence {*

$$\theta_0 := \theta_0 - \alpha \tfrac{1}{m}\textstyle\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})\, x^{(i)}$$

*}*

## Week 2.

### Linear Regression With Multiple Variables

$n$ – number of feature

$x^{(i)}$ – input (features) of $i^{th}$ training example

$x_j^{(i)}$ – value of feature $j$ in $i^{th}$ training example

$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ (for convenience of notation we define $x_0 = 1$)

**Vector notation**

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ . \\ . \\ \theta_n \end{bmatrix}$$

$h(x) = \theta^T X$

**Function to minimize**

$$\min(J(\theta_0, \theta_1, \dots, \theta_n)) = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$$

**Gradient descent algorithm for *n* paramaters $\boldsymbol{\theta_j}$**

*Repeat until convergence {*

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

*}*

*Repeat until convergence {*

$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})\, x_j^{(i)}$ (simultaneously update $\theta_j$ for *j=0,....,n)*

*}*

**Feature scaling (normalization)**

$$x_{normalized(i)} := \frac{x_i - \mu}{\sigma}$$

**Vectorization**

$$\theta := \theta - \alpha\delta$$

$$\delta = \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})\, x^{(i)} \qquad \delta = \begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \\ . \\ . \\ \delta_n \end{bmatrix}$$

**Normal Equation**

$$\theta = (X^T X)^{-1} X^T y$$

# Week 3.

## Binary Classification Problem (Logistic Regression)

**Condition**

$$0 \le h(x) \le 1$$

$$h(x) = g(\theta^T X)$$

$g(z) = \frac{1}{1+e^{-z}}$ Sigmoid Function

**Explanation**

$h(x) = p(y = 1 \mid x_i \theta)$ probability that *y=1*, given *x*, parameterized by $\theta$

If $h(x) \ge 0.5$ then $y = 1$, else $y = 0$

If $z \ge 0$ then $y = 1$, else $y = 0$

**Cost Function**

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h(x^{(i)}), y^{(i)})$$

$$Cost(h(x), y) = \begin{cases} -\log(h(x)) & if\ y = 1 \\ -\log(1 - h(x)) & if\ y = 0 \end{cases}$$

$$Cost(h(x), y) = -ylog(h(x)) - (1 - y)\log(1 - h(x))$$

**$\boldsymbol{\theta}$ minimization**

*Repeat until convergence {*

$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})\, x_j^{(i)}$ (simultaneously update $\theta_j$ for $j=0,\ldots,n$)

*}*

**Vectorized Implementation**

$$\theta := \theta - \alpha \frac{1}{m}\left[\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})\, x^{(i)}\right]$$

**Overfitting**

If we have to many features, the learned hypothesis may fit the training set very well, but fail to generalize to new examples (predict prices on new examples)

**Regularization**

Small values for parameters $\theta_0, \theta_1, \theta_2 \ldots \theta_n$

- Simple hypothesis
- Less prone to overfitting

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

*Repeat until convergence {*

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})\, x_0^{(i)}$$

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})\, x_j^{(i)}$$

*}*

$$\theta = (X^T X + \lambda \begin{bmatrix} 0 & & \\ & 1 & \\ & & 1 \end{bmatrix})^{-1} X^T y$$

For logistic regression

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h(x^{(i)}), y^{(i)}\right) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

*Repeat until convergence {*

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})\, x_0^{(i)}$$

$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h(x^{(i)}) - y^{(i)}\right) x_j^{(i)} + \frac{\lambda}{n}\theta_j\right]$ (simultaneously update $\theta_j$ for $j=0,\ldots,n$)

*}*

# Week 4.

## Neural networks

$a_i^{(j)}$ : activation of unit *i* in layer *j*

$\Theta^{(j)}$ : matrix of weights controlling function mapping from layer *j* to layer *j +1*

If network has $s_j$ units in layer *j*, $s_{j+1}$ units in layer j+1, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$

**Vectorized implementation**

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_2^{(2)} \end{bmatrix}$$

$z^{(2)} = \Theta^{(1)}\mathrm{a}^{(1)}$

$\mathrm{a}^{(2)} = g(z^{(2)})$

# Week 5.

## Cost function and backpropagation

**Cost function for neural networks**

$$J(\theta) = \frac{-1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log\left(h(x^{(i)})\right) + \left(1 + y_k^{(i)}\right)\log(1 - (h(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\theta_{ji}^{(l)}\right)^2$$

**Algorithm for minimizing cost function**

$\delta_j^{(L)} = a_j^{(L)} - y_j$ - error of node *j* in layer *L*

$\delta^{(L-1)} = (\theta^{(L-1)})^T\delta^{(L)}.* g'(z^{(L-1)})$ - error vector for level *L-1*

**Backpropagation algorithm**

Training set $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0 \ (for\ all\ l, i, j)$.

For $i = 1\ to\ m$:

- $Set\ a^{(1)} = x^{(i)}$
- Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \dots, L$
- Using $y^{(i)}$, compute $\delta^{(i)} = a^L - y^{(i)}$
- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \delta^{(L-3)}, \dots, \delta^{(2)}$
- $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)}\delta_i^{(l+1)}$