1 Week 1

1.1 Model Representation

$$h_\theta(x) = \theta_0 + \theta_1 x + \epsilon$$

1.2 Cost Function

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

1.3 Gradient Descent

Want: $minJ(\theta_0, \theta_1)$

Algorithm: repeat until convergence {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1), j = 0, 1$

simutaneously update θ_j

}

2 Week 2

2.1 Multivariate Linear Regression Model

Hypothesis:

$$h_\theta(x) = X\Theta$$

where

$$X = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{pmatrix}_{m \times (n+1)}, \Theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}_{(n+1) \times 1}$$

2.2 Cost function(Vectorization)

$$J(\theta) = \frac{1}{2m} (X\Theta - \vec{y})^T (X\Theta - \vec{y})$$

2.3 Normal Equation

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

Deduction:

$$\begin{aligned}
J(\theta) = \frac{1}{2m}(X\Theta - \vec{y})^T(X\Theta - \vec{y}) \\
= \frac{1}{2m}\|X\Theta - \vec{y}\|^2 \\
\nabla J(\Theta) = \frac{1}{m}X^T(X\Theta - \vec{y}) \\
\nabla J(\Theta) = 0 \\
X^T X\Theta - X^T\vec{y} = 0 \\
\Theta = (X^T X)^{-1}X^T\vec{y}
\end{aligned}$$

3 Week 3

3.1 Classification

Classification: $y \in \{0, 1\}$, $h_\theta(x)$ can be > 1 or < 0.
Logistic Regression: $0 \leq h_\theta(x) \leq 1$

3.2 Logistic Regression

Sigmoid(logistic) function

$$g(t) = \frac{1}{1 + e^{-t}}$$

Logistic Regression Model:

$$h_\theta(x) = \frac{1}{1 + e^{-\Theta^T x}}$$

Interpretation:

- $h_\theta(x)$ =estimated probability that $y = 1$ on input x.
- $h_\theta(x) = P(y = 1|x; \theta)$ means probability that $y = 1$, given x, parameterized by θ.
- $P(y = 0|x; \theta) + P(y = 1|x; \theta) = 1$

3.3 Decision Boundary

Descion boundary is a property of the hypothesis and parameters.
Linear decision boundary $h_\theta(x) = g(\theta_0 + \theta_1 * x_1 + \theta_2 * x_2)$:

$$\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \sum_{i=0}^{n} = \Theta^T x$$

Non-linear decision boundary

3.4 Cost Function

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

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

$h_\theta(x)$ is hypothesis, y is training examples.
Simplified cost function

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^i \log h_\theta(x^i) + (1 - y^i)\log(1 - h_\theta(x^i))]$$

3.5 Gradient Descent

To fit parameters θ: $\min J(\theta)$
To make a prediction given new x:
output $p(y = 1|x; \theta) = h_\theta(x) = \frac{1}{1+e^{-\Theta^T x}}$
Gradient Descent: Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$
$$:= \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}, (j = 0 \cdots n)$$

}
Caution: Simulataneously update all θ_j.
Click here for detailed deduction.

3.6 Advanced Optimization

[x,fval,exitflag]=fminunc(fun,x0,options)

3.7 Multiclass Classification

one-vs-all

3.8 Regularization

Underfitting, just right, overfitting
Addressiong overfitting:

1. Reduce number of features:
    - Manually select which features to keep;
    - Model selection algorithm

2. Regularization:

    - Keep all features, but reduce magnitude/values of parameters θ_j;
    - Works well when there are a lot of features.

Regulariztion:
Small values for parameters $\theta_0, \theta_1, \cdots, \theta_n$

- "Simpler" hypothesis
- Less prone to overfitting.

Cost function

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

$$\min J(\theta)$$

3.9 Regularized linear/logistic regression

Cost function and gradient

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

3.10 Vectorization

$$X = \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ x_0^{(m)} & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{pmatrix}_{m \times (n+1)}, \Theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}_{(n+1) \times 1} y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix}_{m \times 1}$$

So,

$$A = X\Theta = \begin{pmatrix} x_0^{(1)}\theta_0 + x_1^{(1)}\theta_1 + x_2^{(1)}\theta_2 + \cdots + x_n^{(1)}\theta_n \\ x_0^{(2)}\theta_0 + x_1^{(2)}\theta_1 + x_2^{(2)}\theta_2 + \cdots + x_n^{(2)}\theta_n \\ \vdots \\ x_0^{(m)}\theta_0 + x_1^{(m)}\theta_1 + x_2^{(m)}\theta_2 + \cdots + x_n^{(m)}\theta_n \end{pmatrix}_{m \times 1}$$

We have,

$$E = h_\theta(x) - y = g(X\Theta) - y = \begin{pmatrix} g(A^{(1)}) - y^{(1)} \\ g(A^{(m)}) - y^{(m)} \\ \vdots \\ g(A^{(m)}) - y^{(m)} \end{pmatrix}_{m \times 1}$$

.
Now let's take a look at the update of θ:

$$\begin{aligned}
\theta_j &:= \theta_j - \alpha * \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \\
&:= \theta_j - \alpha * \sum_{i=1}^{m} E^{(i)} * x_j^{(i)} \\
&:= \theta_j - \alpha * X(:, j)^T * E \\
&\quad (j = 0, 1, 2, \cdots, n)
\end{aligned}$$

In summary, the matrix form of updating θ can be expressed below:

$$\Theta = \Theta - \alpha * X^T * (h_\theta(x) - y)$$

Thus will pave a convenient way to express in MATLAB. Also, we can express cost function and gradient in matrix form.