Week 03

I. Logistic Regression

1. Classification and Representation

a. Classification

- Example: Email: Spam → 0 or 1; Transaction: fraudulent;
- $y = 0$ as negative class;
- $y = 1$ as positive class;

  ⇒ binary classification.
- Use linear regression and map prediction based on threshold comparison.
  - Threshold classifier $h_\theta(x)$ can be > 0 or < 0
  - $x^{(i)}$ as a feature and y as binary value;

  ⇒ Classification problem **is like** regression problem on a discrete values.

  ⇒ This method doesn't work well because classification is not actually a linear function.

b. Hypothesis Representation

- In some cases, y as 0 or 1 is not good. Sometimes we need real values between 0 and 1;
- To fix this as $0 \leq h_\theta(x) \leq 1$, add it into a Logisitic function.
  - $h_\theta(x) = g(\theta^T x)$ and $z = \theta^T x$ so $g(z) = \frac{1}{1+e^{-z}}$
  - $g(z)$ converge to 0 when z is negative
  - $g(z)$ converge to 1 when z is positive;
  - $g(0) = 0.5$;
  - __/¯¯
  - z good to have values between 0 and 1;
- So $h_\theta(x)$ will give the **probability**.
- $h_\theta(x) = P(y = 1|x;\theta) = 1 - P(y = 0|x;\theta)$
- $\sum P = 1$

c. Decision

- To get discrete 0 or 1 classification, translate the output of the hypothesis function as follows:
  - $h_\theta(x) \geq 0.5 \Rightarrow y = 1$
  - $h_\theta(x) < 0.5 \Rightarrow y = 0$
- The logistic function g behaves as that when its input $\geq$ 0, its output $\geq$ 0.5:
  - $g(z) \geq 0.5$ **when** $z \geq 0$
- Remember:
  - $z = 0, e^0 = 1 \Rightarrow g(z) = 0.5$
  - $z = +\infty, e^{+\infty} \to 0 \Rightarrow g(z) = 1$
  - $z = -\infty, e^{-\infty} \to \infty \Rightarrow g(z) = 0$
  - __/¯¯
- So if our input to g is θ^T, then that means:
  - $h_\theta(x) = g(\theta^T x) \geq 0.5$ **when** $\theta^T x \geq 0$
- So:
  - $\theta^T x \geq 0 \Rightarrow y = 1$
  - $\theta^T x < 0 \Rightarrow y = 0$
- Example:
  - choose parameters of θ
  - resolve θ in $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + ...)$ so g is not necessarily linear when use x_1^2 or else $power^3$;
  - $\theta = [5; 11; 0], y = 1$ if $5 - x_1 + 0x_2 \geq 0$ so it gives $x_1 \leq 5$
    $\Rightarrow$ Decision boundary is a straight vertical line placed on the graph where $x_1 = 5$; at left $y = 1$; at right $y = 0$;
  - note: sigmoid function g(z)

2. Logistic regression

a. Cost function

Cannot use the same cost function like in linear regression; Logistic function will cause the output to be wavy, causing many local optima so it will not be a convex function.

$J(\theta) = \dfrac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$ $Cost(h_\theta(x), y) = -\log(h_\theta(x)); \mathrm{if\ y} = 1$
$Cost(h_\theta(x), y) = -\log(1 - h_\theta(x)); \mathrm{if\ y} = 0$

> When y = 1, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:
>
> - 0 $\to +\infty$
> - 1 $\to \approx 0$

> When y = 0, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:
>
> - 0 $\to$ 0
> - 1 $\to +\infty$

$Cost(h_\theta(x), y) = 0 h_\theta(x) = y$ $Cost(h_\theta(x), y) \to \infty$ if $y = 0$ and $h_\theta(x) \to 1$
$Cost(h_\theta(x), y) \to \infty$ if $y = 1$ and $h_\theta(x) \to 0$

Note that writing the cost function in this way guarantees that J(θ) is convex for logistic regression.

b. Simplified Cost Function and Gradient Descent

i. Cost function

We can compress our cost function's two conditional cases into one case:

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

When $y = 0$ or $y = 1$ then one of the latter parts will be zero;

Full logistic regression cost function:

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

So minimize $J(\theta)$

A vectorized implementation is:

$h = g(X\theta)$

$J(\theta) = 1/m \cdot (-y^T log(h) - (1-y)^T log(1-h))$

ii. Gradient Descent

Remember that the general form of gradient descent is:

Minimize $J(\theta)$: Repeat { $\theta_j := \theta_j - \alpha \frac{d}{d\theta_j} J(\theta)$ }

$\frac{d}{d\theta_j} J(\theta) = 1/m \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

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

Repeat { $\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$ }

This algorithm is identical to the one we used in linear regression (where $h_\theta(x) = \theta^T x$). Simultaneously update all values in θ.

A vectorized implementation is: $\theta := \theta - \frac{\alpha}{m}X^T(g(X\theta) - y)$

c. Advanced optimization

Cost function $J(\theta)$ and want to min $J(\theta)$ so do gradient descent as we have $\frac{d}{d\theta_j}J(\theta)$.

Optimization algorithms:

1. Gradient descent
2. Conjugate gradient
3. BFGS
4. L-BFGS

With the 3 latter ones, no to α, fatser and more complex

Example:

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}; J(\theta) = (\theta_1 - 5)^2 - (\theta_2 - 5)^2$

$\frac{d}{d\theta_1}J(\theta) = 2(\theta_2 - 5); \frac{d}{d\theta_2}J(\theta) = 2(\theta_1 - 5)$

```
function [jVal, gradient] = costFn
    jVal = (theta(1) - 5)^2 + (theta(2) - 5)^2
    gradient = zeros(2, 1)
    gradient(1) = 2*(theta(1) - 5)
    gradient(2) = 2*(theta(2) - 5)

options = optimset('GradObj', 'on', 'MaxIter', 100);
initialTheta = zeros(2,1);
   [optTheta, functionVal, exitFlag] = fminunc(@costFn, initialTheta, options);
```

3. Multiclass classification

Instead of y = {0,1} we will expand our definition so that y = {0,1...n}.

Since y = {0,1...n}, we divide our problem into n+1 binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

$y \in 0, 1...n$

$h_\theta^{(0)}(x) = P(y = 0|x; \theta)$

$h_\theta^{(1)}(x) = P(y = 1|x; \theta)$

...

$h_\theta^{(n)}(x) = P(y = n|x; \theta)$

$prediction = max_i(h_\theta^{(i)}(x))$

We are basically choosing one class and then lumping all the others into a single second class. Repeat this. Apply binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.

II. Regularization

1. The problem of overfitting

Example: Housing prices using linear regression

- $\theta_0 + \theta_1 x$ called underfit or high bias
- $\theta_0 + \theta_1 x + \theta_2 x^2$ called just right
- $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ called overfit or high variance

  Plotted pass near the points

  ⇒ Overfit = many features + the learned hypothesis fit the training set ($J(\theta) \approx 0$) + fail to fit new examples.

  $j(\theta) = 1/2m \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

Example: Logistic regression

- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ called underfit
- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + ..)$

- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + .. + ..)$

  Plotted create a boundary between points of 2 or more classes

Address overfitting:

1. Reduce number of features:
   - Manually select which features to keep
   - Model selection algorithm
2. Regularization:
   - Keep all features but reduce values of θ_j
   - Regularization works well when we have a lot of slightly useful features.

2. Cost function

If we have overfitting from our hypothesis function, we can reduce the weight that some of the terms in our function carry by increasing their cost.

Say we wanted to make the following function more quadratic: $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

We want to eliminate the influence of $\theta_3 x_3 + \theta_4 x_4 \colon \Rightarrow$ modify our **cost function**

min $\theta 1/2m \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$

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

The lambda, is the **regularization parameter**. It determines how much the costs of our theta parameters are inflated.

3. Regularized Linear regression

Gradient Descent

We will modify our gradient descent function to separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0.

Repeat {

$$\theta_0 := \theta_0 - \alpha/m \sum_{i=1}^m h_\theta(x^{(i)} - y^{(i)})x_0^{(i)} \ \ \theta_j := \theta_j - \alpha[(1/m \sum_{i=1}^m h_\theta(x^{(i)} - y^{(i)})x_j^{(i)}) + \lambda/m.\theta_j] \ \ j \in 1..n$$

}

The term $\lambda/m.\theta_j$ performs the regulation. With some manipulation $\theta_j := \theta_j(1 - \alpha\lambda/m) - \alpha...$

The first part will be always less than 1. So reducing θ_j by some amount on every update gives as the same second part as before.

Normal Equation

Use the non-iterative normal equation. To add in regularization, the equation is the same as our original, except that we add another term inside the parentheses:

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

L is a matrix with 0 at the top left and 1's down the diagonal, with 0's everywhere else.

4. Regularized logistic regression

We can regularize logistic regression in a similar way that we regularize linear regression.

As a result, we can avoid overfitting.

Recall that our cost function for logistic regression was:

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

We can regularize this equation by adding a term to the end:

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

The sum in the added term means to explicitly exclude the bias term θ_0. So the gradient will be like:

Repeat {

$$\theta_0 := \theta_0 - \alpha/m\sum_{i=1}^{m} h_\theta(x^{(i)} - y^{(i)})x_0^{(i)}\ \ \theta_j := \theta_j - \alpha[(1/m\sum_{i=1}^{m} h_\theta(x^{(i)} - y^{(i)})x_j^{(i)}) + \lambda/m.\theta_j]\ \ j \in 1..n$$

}