Week 3

Classification

Classification is similar to regression, just that there are a smaller numbers of discrete values instead of real values

Ex:

Email: Spam / Not Spam?
Online transactions: Fraudulent (Yes/No)?
Tumor: Malignant / Bengn?

$y \in \{0, 1\}$ — 0: Negative class (eg. bengn tumors) ← absence of malignant tumor
1: Positive class (eg. malignant tumors) ← Presence of malignant tumor

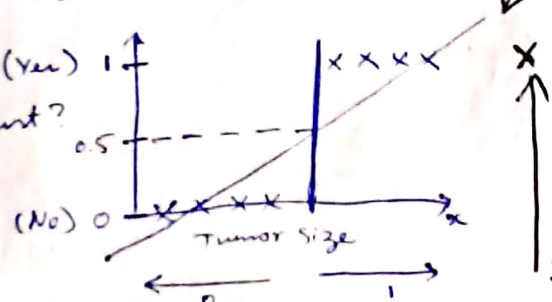
Binary classification (Only 2 classes - 0 & 1)

$y \in \{0, 1, 2 ...\}$ ∈ Multiclass classification

Problem with applying linear regression to classification problem

We can map all predictions > 0.5 as 1 and < 0.5 as 0.

linear regression line

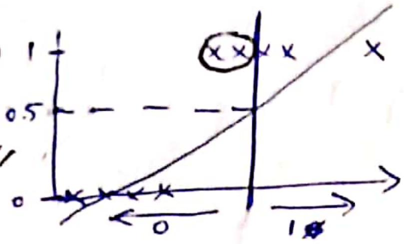


If $h_\theta(x) \geq 0.5$, predict $y = 1$

If $h_\theta(x) < 0.5$, predict $y = 0$

Suppose we add a new data point here. Then it gets screwed

Notice how that point changes the linear regression line, due to which 2 data points are classified wrong



Here $h_\theta(x)$ can be > 1 and $h_\theta(x)$ can be < 0 but y should $\in (0, 1)$ so this doesn't make sense.

We need $0 \leq h(x) \leq 1$

∴ We cannot use linear regression for classification problem. So we use

Logistic Regression

Logistic Regression

- we need to plot a better curve that fits the data better
- $0 \le h_\theta(x) \le 1$ since $y \in (0,1)$

$\therefore$ We use the Sigmoid Function (also called Logistic Function)

$h_\theta(x) = g(\theta^T x)$ ← hypothesis function in vector form on which we apply the sigmoid function

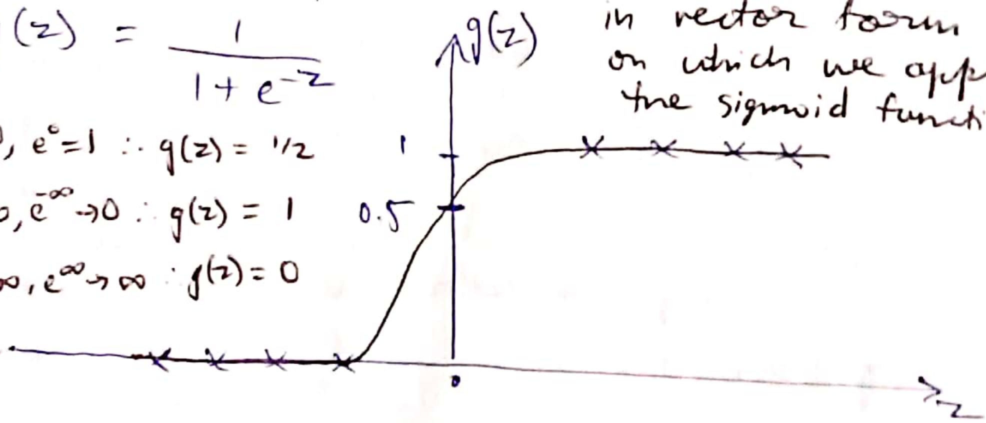
Let $z = \theta^T x$,

$g(z) = \dfrac{1}{1+e^{-z}}$

① $z = 0$, $e^0 = 1$ $\therefore g(z) = 1/2$

② $z \to \infty$, $e^{-\infty} \to 0$ $\therefore g(z) = 1$

③ $z \to -\infty$, $e^{\infty} \to \infty$ $\therefore g(z) = 0$



$h_\theta(x)$ will be probability that our output is 1

$h_\theta(x) = P(y=1 \mid x;\theta) = 1 - P(y=0 \mid x;\theta)$

Probability of $y=1$ given x parameterized by θ

Exa:

If $h_\theta(x) = 0.7$ in tumor example, it means the patient has a 70% probability of having a malignant tumor $(y=1)$.

The probability of having a benign tumor is $1 - 0.7 = 0.3$ $\therefore$ 30%.

Scanned with CamScanner

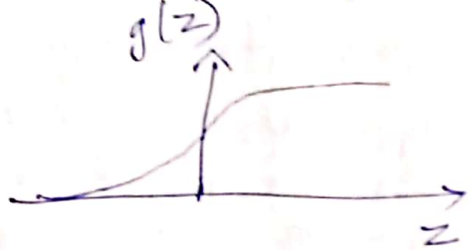
Decision Boundary

In order to get discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows,

$h_\theta(x) \geq 0.5 \rightarrow y = 1$

$h_\theta(x) < 0.5 \rightarrow y = 0$

~~when~~



From the diagram,

when $z \geq 0$, $g(z) \geq 0.5$

$z < 0$, $g(z) < 0.5$

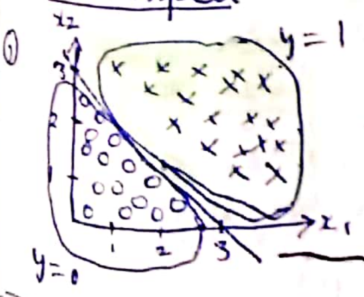
since $z = \theta^T x$,

- predict $y = 1$ if $h_\theta(x) \geq 0.5$ or $\theta^T x \geq 0$
- predict $y = 0$ if $h_\theta(x) < 0.5$ or $\theta^T x < 0$

Decision boundary is the line that seperates the area where $y=0$ and $y=1$. It is created by the hypothesis function.

Examples

①



$h_\theta(x) = g(\theta^T x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

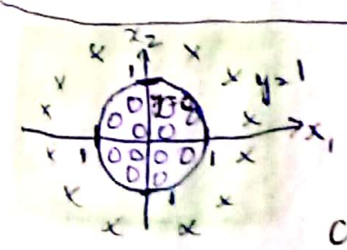
Let $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$ i.e $\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

$\therefore$ Predict $y = 1$ if $-3 + x_1 + x_2 \geq 0$

$= x_1 + x_2 \geq 3$

equation of line (Decision boundary) $\rightarrow$ At these points $h_\theta(x) = 0.5$

②



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$

$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ $\therefore$ Predict y if $-1 + x_1^2 + x_2^2 \geq 0$

$= x_1^2 + x_2^2 \geq 1$

equation of Circle of radius 1 $\rightarrow$ $(x_1^2 + x_2^2 = 1)$

Logistic Regression Model

Cost Function

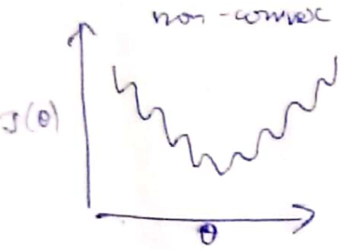
For linear regression $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

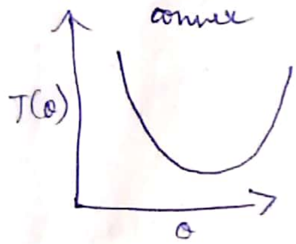
The cost the learning algorithm needs to pay if it predicted $h_\theta(x^{(i)})$ instead of y. — $\mathrm{cost}(h_\theta(x^{(i)}), y)$

$$\mathrm{Cost}(h_\theta(x), y) = \frac{1}{2}(h_\theta(x) - y)^2$$

Since we are using a logistic function for $h_\theta(x)$ we will get a non-convex function for the cost function if we try to use this for logistic regression

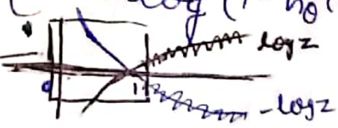
- Since $\frac{1}{1+e^{-\theta^T x}}$ is non-linear this happens





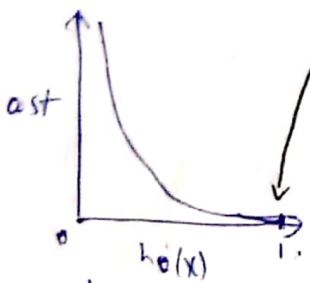
Therefore we will use the following cost function:

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



graph of log z & -log z
consider only 0-1

If $y=1$

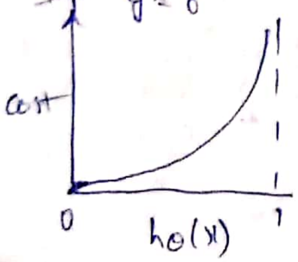


Cost $= 0$ if $y=1$, $h_\theta(x) = 1$

As $h_\theta(x) \to 0$, cost $\to \infty$

- ∞ at 0 since if $y=1$ & it predicts $h_\theta(x) = 0$, we need to penalise it by a very large cost

If $y=0$



Cost $= 0$ if $y=0$, $h_\theta(x) = 0$

As $h_\theta(x) \to 1$, cost $\to \infty$

Simplified Cost Function

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

Combine the two cases in one equation,

$$\therefore \text{cost}(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

If $y=1$, $\text{cost}(h_\theta(x), y) = -\log h_\theta(x) \cancel{- 0 \cdot \log(1-h_\theta(x))}$

If $y=0$, $\text{cost}(h_\theta(x), y) = -\log(1-h_\theta(x)) \cancel{+ 0 \cdot \log(h_\theta(x))}$

$\therefore$ Logistic Regression Cost Function

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

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

Gradient Descent

Repeat {

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \quad \text{(simultaneously update all } \theta_j)$$

}

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

$\therefore$ Repeat {

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

}

This algorithm looks identical to linear regression however note that here $h_\theta(x^{(i)}) = \frac{1}{1+e^{-\theta^T x}}$ instead of $\theta^T x$

Vectorised implementation:

$$\theta := \theta - \frac{\alpha}{m}X^T(g(X\theta) - \vec{y})$$

We can use feature scaling in logistical regression too to make gradient descent converge faster.

Advanced Optimization

Apart from gradient descent, there are other advanced optimization functions:

- Conjugate Gradient
- BFGS
- L-BFGS

Advantages: - No need to manually pick α
- Often faster than gradient descent

Disadvantages: - More complex

Example

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

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

Here $\min_\theta J(\theta)$ is at $\theta_1 = 5$, $\theta_2 = 5$

We find this how:

```
function [jVal, gradient] = costFunction(theta)
    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(@costFunction, initialTheta, options);
```

optTheta - values of θ

functionVal - value of $J(\theta)$ at $\min J(\theta)$ → Should ≈ 0

exitFlag - 1 - means converged
0 - not converged

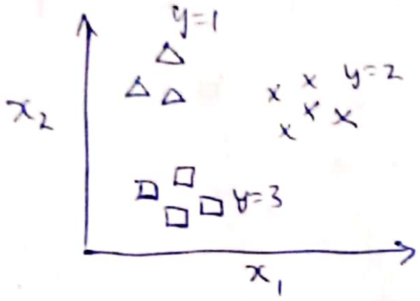
Multiclass Classification

When there are multiple classes

eg. Email foldering/tagging: Work ($y=1$), Friends ($y=2$), Family ($y=3$), Hobby ($y=4$)

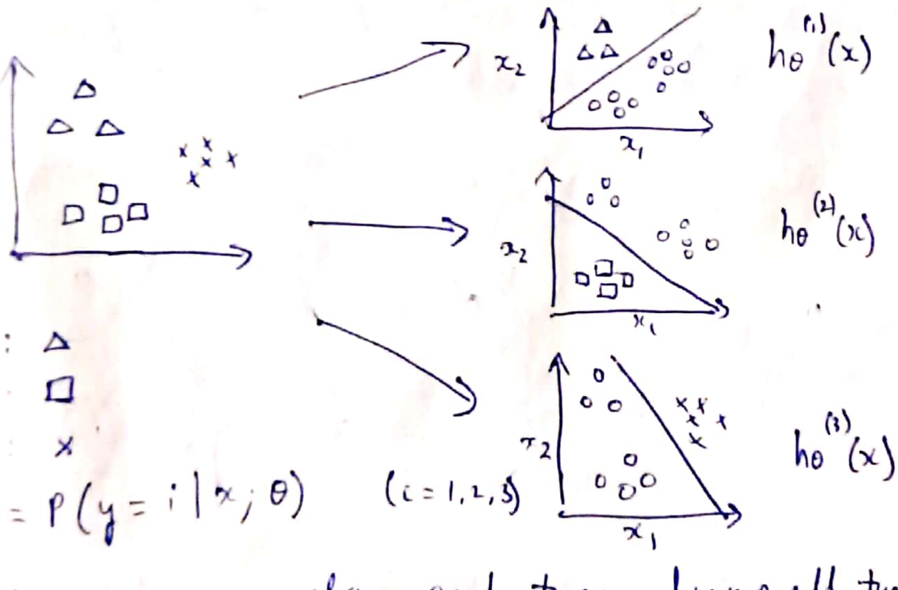
Medical diagrams: Not ill ($y=1$), Cold ($y=2$), Flu ($y=3$)

Weather: Sunny ($y=1$), Cloudy ($y=2$), Rain ($y=3$), Snow ($y=4$)



$y \in \{0, 1, 2 \ldots n\}$

We can classify into different classes using One-vs-all / One-vs-rest Method



Class 1: △

Class 2: □

Class 3: ×

$h_\theta^{(i)}(x) = P(y=i \mid x;\theta) \quad (i = 1, 2, 3)$

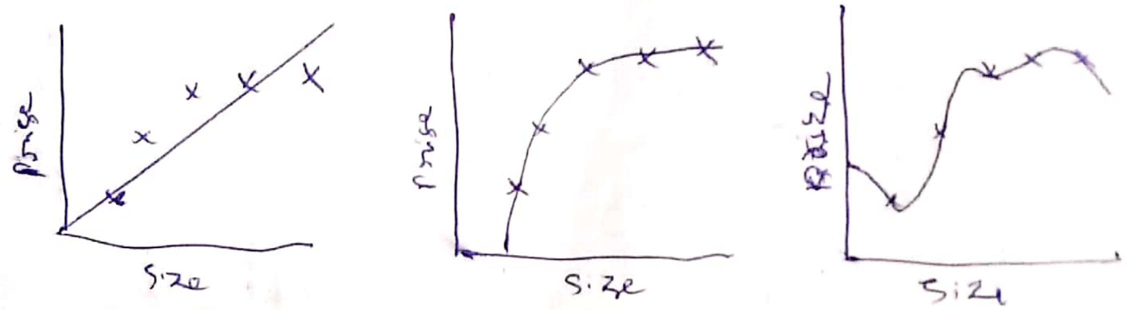
We choose one class and then lump all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.

$$h_\theta^{(0)}(x) = P(y=0 \mid x;\theta)$$
$$h_\theta^{(1)}(x) = P(y=1 \mid x;\theta)$$
$$\cdots$$
$$h_\theta^{(n)}(x) = P(y=n \mid x;\theta)$$
$$\text{prediction} = \max_i \left(h_\theta^{(i)}(x)\right)$$

Overfitting

If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalise to new examples (predict prices on new examples

Linear Regression (housing prices)



$\theta_0 + \theta_1 x$ | $\theta_0 + \theta_1 x + \theta_2 x^2$ | $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 \ldots$

Underfitting

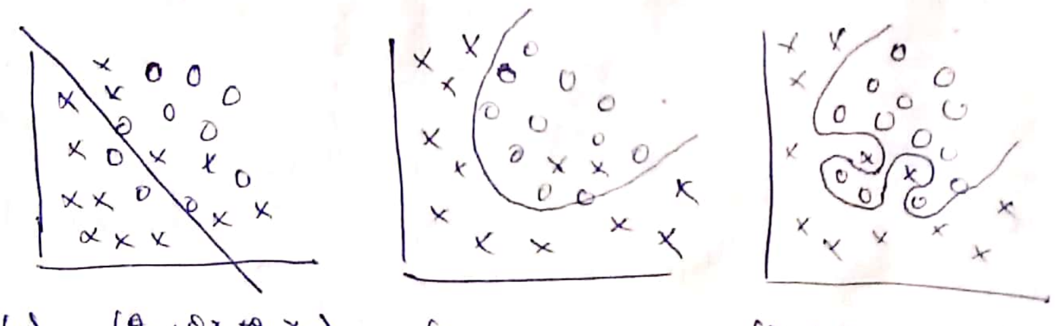
High bias ← Has a preconception or bias that the price will increase linearly even though data doesn't agree

Doesn't fit training data well

Overfitting

High variance

Logistic regression



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

Underfitting

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots)$

overfitting

Addressing overfitting

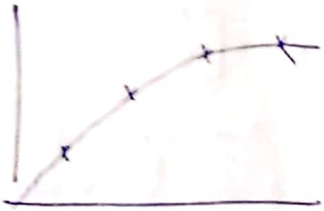
1. Reduce no. of features
   - Manually select which features to keep
   - Model selection algorithm
2. Regularization
   - keep all features, but reduce magnitude/values of θ_j
   - works well when we have a lot of features, each contributing a bit to predicting y

Regularisation

Suppose we have overfitting.

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

If we eliminate the influence of $\theta_3 x^3$ & $\theta_4 x^4$ we get a quadratic function $\theta_0 + \theta_1 x + \theta_2 x^2$ that fits our data well



we modify our cost function to

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$$

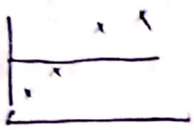
Here we have added 2 extra terms to inflate the cost of θ_3 & θ_4. So for the cost function to get close to 0, we have to reduce θ_3 & θ_4 to 0, and this will reduce $\theta_3 x^3$ & $\theta_4 x^4$ in the ~~equation~~ hypothesis function.

Similarly we regularise all our θ parameters,

$$\boxed{\min_\theta \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2}$$

where λ - regularization parameter this is θ_j square

Note: If λ is too large, it can result in underfitting since all the θ values are reduced eventually making the hypothesis function $h(x^{(i)}) = \theta_0$



Note: we Don't penalise θ_0. Here j is from 1 to n

Regularized Linear Regression

Gradient descent:

Repeat {

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

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right]$$

$j \in \{1, 2 \ldots n\}$

}

Note that we took θ_0 seperately since its not regularised

If we take θ_j common for the second equation,

$$\theta_j := \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) \right]$$

Here $(1 - \alpha \frac{\lambda}{m})$ will be less than 1, Intuitively we can see the equation as same as normal gradient descent but θ_j is reduced by some amount (eg: $0.99\theta_j$) every iterati

Normal equation:

$$\theta = (X^T X + \lambda \cdot L)^{-1} X^T y$$

where $L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ 0 & \cdots & 1 & 0 \\ 0 & \cdots & & 1 \end{bmatrix}$

$(n+1) \times (n+1)$

So for $n=2$,

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Before we had a problem with the normal equation

$$(X^T X)^{-1} X^T y$$

Here if $m > n$, then $X^T X$ would be non-invertable

no. of training examples, no. of features

However $(X^T X + \lambda \cdot L)$ makes it invertible and solves this problem.

Regularized Logistic regression

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

$+ \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$ ← This is square

From $\theta_1, \theta_2 \ldots$ Not θ_0

Gradient descent:

← same derivation,

Repeat {

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

$\theta_j := \theta_j(1-\alpha\frac{\lambda}{m}) - \alpha\left[(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)})\right]$

}

only difference is that $h_\theta(x^{(i)}) = \frac{1}{1+e^{-\theta^T x}}$ here

Advanced optimization:

function [jVal, gradient] = costFunction(theta)

jVal = [code to compute $J(\theta)$];

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

$+ \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

gradient(1) = [code to compute $\frac{\partial}{\partial\theta_0} J(\theta)$];

$\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$

gradient(2) = [code to compute $\frac{\partial}{\partial\theta_1} J(\theta)$];

$(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}) + \frac{\lambda}{m}\theta_1$

gradient(3) = [code to compute $\frac{\partial}{\partial\theta_2} J(\theta)$];

$(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}) + \frac{\lambda}{m}\theta_2$

⋮

gradient(n+1) = [code to compute $\frac{\partial}{\partial\theta_n} J(\theta)$];

This is same as before but we have updated the equations to add the regularisation term. We later run this into fminunc

fminunc (@costFunction, initialTheta, options);