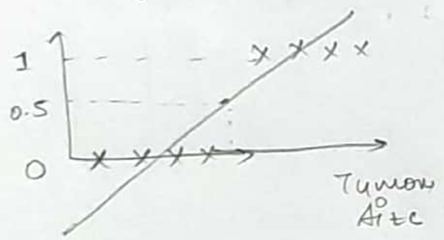
Week - 3

⇒ classification → variable y is discrete valued
↳ we'll use logistic regression for same.

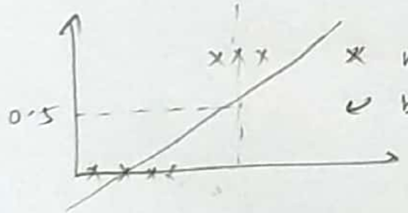
example :- email : spam | not spam?
Online Transactions: fraud (Yes | No)?
Tumor malignant | Benign?

$y \in \{0,1\}$ 0: "Negative class" 1: "Positive class" binary class
↳ Right now, we'll deal with classification with $y \in \{0,1\}$ only

Training set



we can try to fit linear regression.
$h_\theta(x) = \theta^T x$
Threshold classifier output $h_\theta(x)$ at 0.5:
If $h_\theta(x) \geq 0.5$, predict "$y=1$"
If $h_\theta(x) < 0.5$, predict "$y=0$"



✱ not good because

classification $y = 0$ or 1
$h_\theta(x)$ can be > 1 or < 0

logistic regression : $0 \leq h_\theta(x) \leq 1$
classification.

⇒ Hypothesis representation
logistic regression → want $0 \leq h_\theta(x) \leq 1$

$h_\theta(x) = g(\theta^T x)$ $\quad g(z) = \frac{1}{1+e^{-z}}$ } sigmoid function / logistic function

Scanned with CamScanner

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \quad \nearrow \underline{\underline{\theta's}}.$$

Now, we need to fit parameters to our data.

Interpretations of Hypothesis output

$h_\theta(x)$ = estimated probability that $y=1$ on input x.

Example :- $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorsize} \end{bmatrix}$

$h_\theta(x) = 0.7 \Rightarrow$ 0.7 chance that $y=1$

So, it's 70% chance that tumour is malignant.

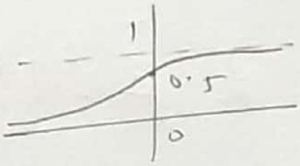
$h_\theta(x) = P(y=1 \mid x;\theta) \Rightarrow$ "Probability that $y=1$, given x, parametrised by θ".

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

$\Rightarrow$ **Decision Boundary**

$h_\theta(x) = g(\theta^T x)$

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



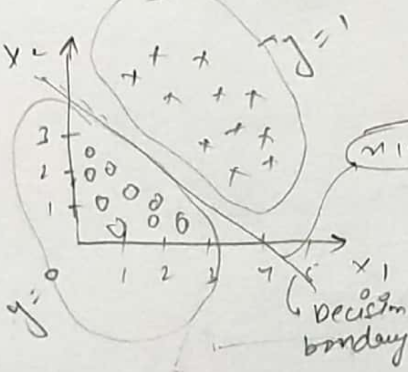
$\rightarrow$ Suppose predict

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

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

Now $g(z) \geq 0.5$ for $z \geq 0 \Rightarrow \theta^T x \geq 0$

Thus "$y=1$" when $\theta^T x \geq 0$, "$y=0$", when $\theta^T x < 0$



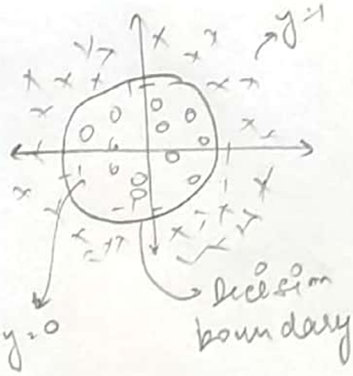
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \rightarrow$ we will see how to find θ later.

let us say $\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

"$y=1$" if $\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0 \Rightarrow x_1 + x_2 \geq 3 \nearrow$ equation of straight line

Non linear decision boundary



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

let's say after procedure, we have

$\theta_0 = -1 \quad \theta_1 = 0 \quad \theta_2 = 0 \quad \theta_3 = 1 \quad \theta_4 = 1$

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

$\Rightarrow \underbrace{x_1^2 + x_2^2 \geq 0}_{\text{equation of circle}}$

⇒ Cost function

Training set : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \dots x^{(m)}, y^{(m)})$

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} R^{n+1} \quad x_0 = 1, \; y \in \{0, 1\}$

$h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$, How to choose θ?

→ linear regression : $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2}_{cost(h_\theta(x^{(i)}), y^{(i)})}$

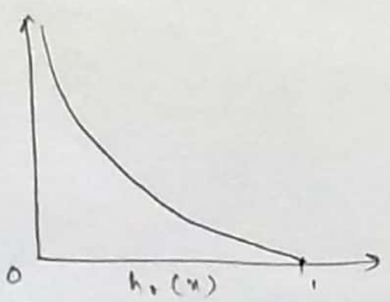
let's define

$\left[cost(h_\theta(x), y) = \frac{1}{2} (h_\theta(x) - y)^2 \right]$

logistic regression cost function

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

If $y = 1$

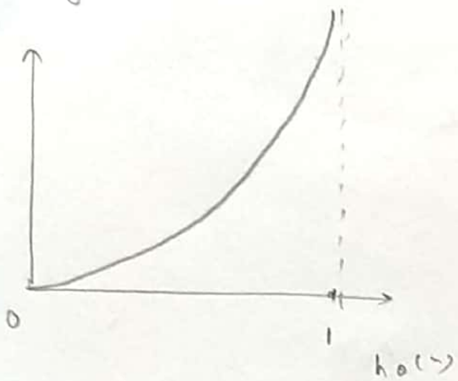


cost = 0 if $y = 1$, $h_\theta(x) = 1$

But as $h_\theta(x) \to 0$
cost $\to \infty$

That means that if $h_\theta(x) = 0$ and $y = 1$, we'll have a very large cost.

If $y = 0$



→ cost function in this way guarantees that $J(\theta)$ is convex for logistic regression.

⇒ Simplified cost function and Gradient Descent

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

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

why is always either 0 or 1

Now, let's make a combined cost fn.

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

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

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

→ So, this cost fn. works

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

To fit θ such that $J(\theta)$ is minimised.

To make a prediction given new x:

output $h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$ } → $P(y = 1 | x; \theta)$

Gradient descent

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

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0 x_0 + \theta_1 x_1)}}$$

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

$$= -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log\left(\frac{1}{1+e^{-(\theta_0 x_0 + \theta_1 x_1)}}\right) + (1-y^{(i)}) \log\left(1 - \frac{1}{1+e^{-(\theta_0 x_0 + \theta_1 x_1)}}\right)$$

let's take 1 term.

$$\frac{dJ(\theta)}{d\theta_1} = y \cdot \cancel{(1+e^{-(\theta_0 x_0 + \theta_1 x_1)})} \left(\frac{e^{-(\theta_0 x_0 + \theta_1 x_1)} \cdot x_1}{(1+e^{-(\theta_0 x_0 + \theta_1 x_1)})^2}\right)$$

$$= y \frac{e^{-\theta^T x}}{1+e^{-(\theta^T x)}} x_1 + (1-y) \cdot \frac{1}{e^{-\theta^T x}} \cdot \frac{\cancel{e^{-(\theta^T x)}} \cdot -x_1}{(1+e^{-\theta^T x})}$$

$$= y \frac{e^{-\theta^T x}}{1+e^{-\theta^T x}} \cdot x_1 + \frac{(y-1)\, x_1}{1+e^{-\theta^T x}}$$

$$= \frac{y x_1 (\cancel{1+e^{-\theta^T x}})}{} - \frac{x_1}{1+e^{-\theta^T x}}$$

$$\boxed{= (y - h_\theta(x))\, x_1}$$

$6 - x_2 = 0$

$x_2 = 6$

so GD.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

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

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

↳ This is exactly the same as linear regression, just $h_\theta(x)$'s definition is changed; ⇒ linear → $h_\theta x = \theta^T x$

logistic → $h_\theta x = \frac{1}{1+e^{-\theta^T x}}$

vectorised implementation

$$\boxed{h = g(X\theta)}$$

$$X = \begin{bmatrix} x_0^1 & x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_0^2 & x_1^2 & \cdots & & x_n^2 \\ \vdots \end{bmatrix}_{m \times n} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}_{n \times 1} = \underbrace{\begin{bmatrix} x_0^1\theta_0 + x_1^1\theta_1 \cdots \\ x_0^2\theta_0 + x_1^2\theta_1 \cdots \\ \\ \end{bmatrix}}_{X\theta}$$

$m \times 1$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{m \times 1} \quad y^T = 1 \times m$$

$$y^T = [1 - y_1 \; 1 - y_2 \; \cdots \; y_m]$$

$$\log(h) = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_m \end{bmatrix}_{m \times 1}$$

$$J(\theta) = \frac{1}{m} \left(y_1 h_1 + y_2 h_2 \cdots y_m h_m + (1-y_1)(1-h_1) \right]$$

$$= \sum yh + (1-y)(h)$$

$$\boxed{J(\theta) = \frac{1}{m} \left(-y^T \log(h) - (1-y)^T \log(1-h) \right)}$$

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

$$(h_1 - y_1)x_0^1 + (h_2 - y_2)x_0^2 \cdots$$

$$\begin{bmatrix} h_1 - y_1 \\ h_2 - y_2 \\ \vdots \\ h_m - y_m \end{bmatrix}_{m \times 1}$$

$X^T \rightarrow n \times m.$

$$\begin{bmatrix} x_0^1 & x_0^2 & x_0^3 & \cdots & x_0^n \\ x_0^2 & x_1^2 & \cdots & & x_n^2 \\ & & \vdots \end{bmatrix}_{n \times m} \begin{bmatrix} h_1 - y_1 \\ \vdots \\ h_m - y_m \end{bmatrix}_{m \times 1} = \begin{bmatrix} \\ \\ \end{bmatrix}$$

Scanned with CamScanner

⇒ Advanced optimisation

optimisation Algorithm : → Gradient descent

$$\begin{bmatrix} \text{conjugate gradient} \\ \text{BFGS}, \quad \text{L-BFGS} \end{bmatrix}$$

So, Given θ, we have code to compute

$J(\theta)$

$\frac{\partial J(\theta)}{\partial \theta_j}$ (for $j = 0, 1, \dots n$)

Now, we can supply these to various optimisation algorithms

Advantages :- No need to give α, faster than gradient descent

Disadvantages :- More complex

theta = $\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ function [Jval, gradient] = costFunction(theta)

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

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

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

$\vdots$

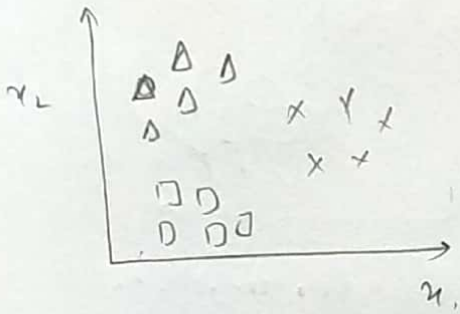
⇒ Multiclass classification : One vs All.

ex. Email foldering : works, friends, Family, Hobby

$y=1$ $y=2$ $y=3$ $y=4$

Medical diagrams : Not ill, cold, Flu

$y=1$ $y=2$ $y=3$



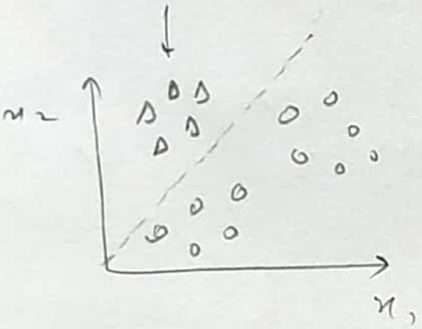
→ class 1 : Δ

class 2 : □

class 3 : X

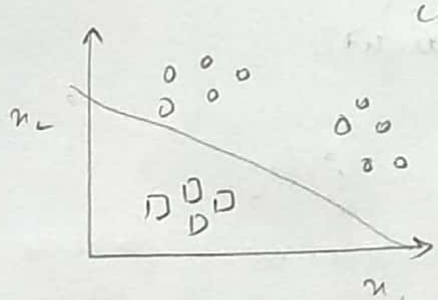
One vs All.

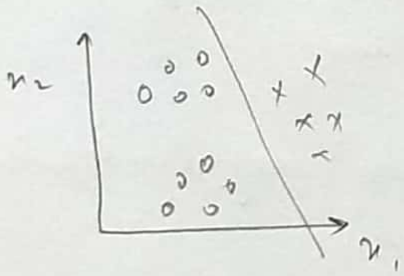
Now, we turn this into 3 binary classification problem



→ $h_\theta^{(1)}(x)$ class 1

class 2 $h_\theta^{(2)}(x)$





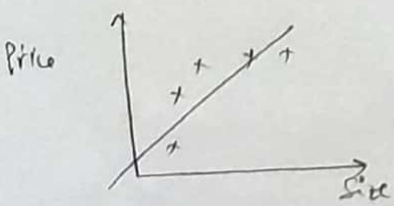
$h_\theta^{(3)}(x)$, so, we fit 3 classifiers

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class (i) to predict probability that $y=i$.

On a new input x, to make a prediction, pick class i that maximises, $\max\limits_i h_\theta^{(i)}(x)$

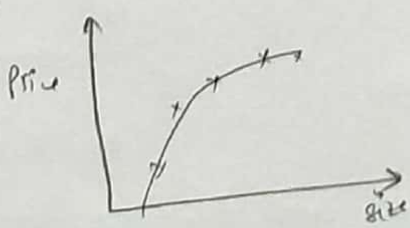
⇒ The Problem of overfitting

Housing price



$\theta_0 + \theta_1 x$

'Underfit', ("High bias")



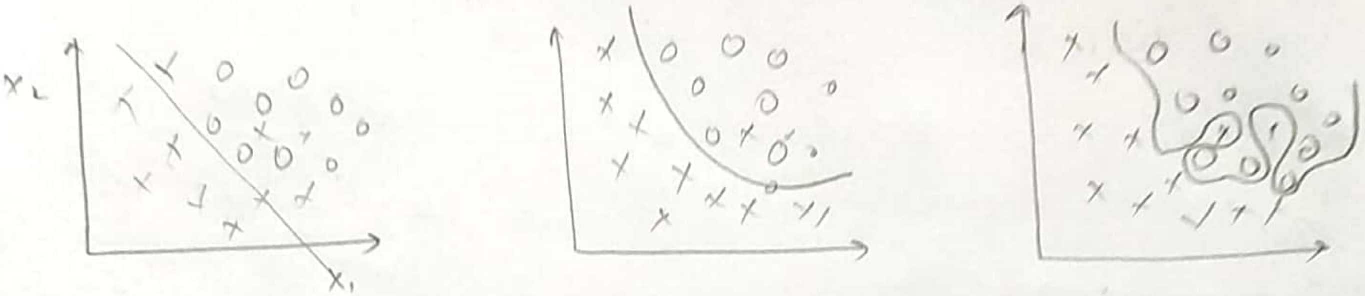
$\theta_0 + \theta_1 x + \theta_2 x^2$

'just right'

"overfit", "High variance"

Overfitting : when we use too many features, the curve fits very well to training set, $J(\theta) \approx 0$ maybe, but now, we fail to generalize over a new input x.

For logistic regression



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

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$ (right)

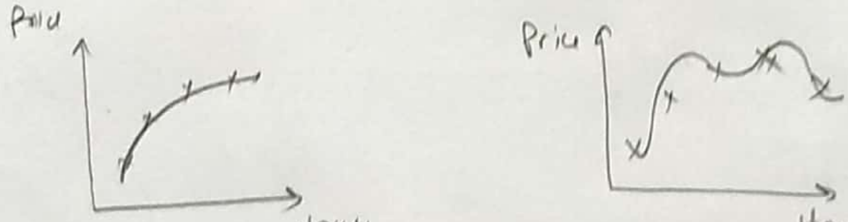
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 \ldots \ldots)$ (overfit)

→ Address overfitting

1. Reduce no. of features
   - ↳ (a) Manually select which features to keep.
   - ↳ (b) Model selection algorithm (later)
2. Regularization.
   - ↳ (a) keep all the features, but reduce magnitude/values of parameters θ_j (less useful ones)
   - (b) works when we have lot of features, each contributes to predicting y.

⇒ Cost function, regularisation

Intuition



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

Suppose, we make θ_3 and θ_4 very small, it would be as if we are getting rid of θ_3 and θ_4.

Regularisation

↳ Small values for $\theta_0, \theta_1 \dots \theta_n$

- "simpler hypothesis"
- less prone to overfitting

Housing

- Features: $x_0, x_1 \dots x_{100}$
- Parameters: $\theta_0, \theta_1 \dots \theta_{100}$

} Here, we do not know which feature will give higher order terms or which features are irrelevant.

Now, we modify the cost function

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \underbrace{\lambda \sum_{i=1}^{n} \theta_j^2}_{\text{regularisation term}}\right]$$

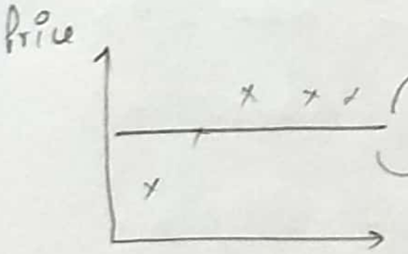
So, in regularised linear regression, we choose θ to minimise

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

What if λ is very large, say 10^{10}

↳ $\theta_1, \theta_2, \theta_3 \dots \theta_n \approx 0$

So $h_\theta(x) \approx \theta_0$



$$h_\theta(x) = \theta_0 + \cancel{\theta_1} x_1 + \cancel{\theta_2} x_2 + \cancel{\theta_3} x_3 \dots$$

Thus, we also say that $h_\theta(x)$ has high bias for θ_0.

So, λ needs to be chosen wisely.

⇒ Regularised Linear Regression

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

↓

Regularised cost function

We need to find $\min_\theta J(\theta) \Rightarrow \theta$ corresponding to $\min J(\theta)$.

Gradient Descent

Repeat {

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

$j = (1, 2, 3 \dots n)$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_0^{(i)}$$ → writing θ_0 case seperately

}

↓ because we don't penalise θ_0.

Now, we need to modify GD.

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

$\frac{\partial J(\theta)}{\partial \theta_j}$ for New $J(\theta)$

($j = 1, 2, 3 \dots n$)

update rule now becomes:

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

→ $\left(1 - \frac{\alpha\lambda}{m}\right) < 1$, this would be slightly less than 1, like 0.99.

So, $\theta_j := \underbrace{0.99\,\theta_j} - (GD)$ ↳ original update

↓

So, basically, we are shrinking the θ_j at every iteration, so that eventually, it becomes smaller.

Normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}_{m \times n+1} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \mathbb{R}^m$$

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

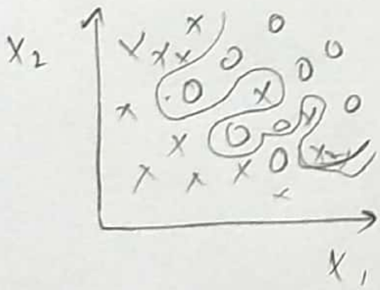
Now, for regularised - $J(\theta)$

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & \ddots 1 \end{bmatrix}_{(n+1) \times (n+1)} \right)^{-1} X^T y$$

Non invertibility issue

If $\lambda > 0$, then $\underbrace{X^T X + \lambda \begin{bmatrix} 0 & & 0 \\ & 1 & \\ 0 & & \ddots 1 \end{bmatrix}}$ is invertible / non singular.

→ Regularised logistic Regression



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

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

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

$\theta_1, \theta_2 \ldots \theta_n$

So, even with a lot of features regularisation can help with overfitting problem.

GD

Repeat {

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

Deal differently with θ_0.

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

$(j = 1, 2, 3 \ldots n)$

$\theta_1, \theta_2 \ldots \theta_n$

$\frac{dJ(\theta)}{d\theta_j}$

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

Advanced optimisation → (see again)

function [jval, gradient] = costFunction(theta)

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

gradient 1 = [code to compute $\frac{dJ(\theta)}{d\theta_0}$]

gradient 2 = [code to compute $\frac{dJ(\theta)}{d\theta_1}$]

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