

Resources : Book - Goodfellow et al  
 : Videos - Fifi Lee, Nando de Freitas  
 Ali Ghodsi

→ Machine learning classified into :

- 1) supervised learning
- 2) unsupervised learning
- 3) semi-supervised learning.

→  $x^{(i)}$  — data / feature vector  
 $y^{(i)}$  — output (mostly scalar)

→ In supervised learning, data is labelled (answer is known)

→ Based on training examples, we try developing relation between input and output

→ Methods (supervised learning techniques)

- 1) linear regression → Except this, all others are classification
- 2) logistic regression
- 3) Naive Baye's classification
- 4) linear discriminant analysis
- 5) quadratic discriminant analysis.
- 6) support vector machine
- 7) neural network classification

→ Classification — discrete outputs only (Identify digit from image)

→ In unsupervised learning, we try finding some coherent structure in the data. Here there is only input, no output. (unsupervised learning also called segmentation).

→ Technique — k-means clustering

→ principal component analysis

- 3) independent component analysis
- 4) Gaussian mixture model (GMM)

Disadvantage of S.L. - manually label data, which is tedious.

### Linear regression

EQ. Predicting price of flat. Following training set	
size (sq. feet)      price (in Rs)	
2000	80K
1400	50K
1500	60K
872	15K

suppose such so examples given

→ Hypothesis  $h_{\theta}(x) = \theta_0 + \theta_1 x$  (linear model)

→ To fit a model for given problem, we

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

w.r.t  $\theta$       model      given

$$\text{where } \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad \begin{matrix} \text{output} \\ (\hat{y}^{(i)}) \end{matrix}$$

- Regression - a measure to represent relation between input and output
- Linear regression - regression with linear relationship between input and output  
(Here  $y^{(i)}$  can be real, unlike classification)
- Here our aim is to find  $\theta_0$  and  $\theta_1$ . We do it as follows.

$$y^{(i)} = h_{\theta}(x^{(i)}) + \epsilon^{(i)}, \quad i = 1, \dots, m$$

$\epsilon^{(i)}, i = 1, \dots, m$  are errors.

where

$$\epsilon = \begin{bmatrix} \epsilon^{(1)} \\ \epsilon^{(2)} \\ \vdots \\ \epsilon^{(m)} \end{bmatrix} \quad \text{gives} \quad \|\epsilon\|^2 = (\epsilon^{(1)})^2 + (\epsilon^{(2)})^2 + \vdots + (\epsilon^{(m)})^2$$

so we need to minimize  $\frac{1}{2m} \|\epsilon\|^2$

$$\underline{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \underline{x} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(m)} \end{bmatrix} \quad \underline{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$J(\theta) = \frac{1}{2m} \|\underline{y} - \underline{x}\underline{\theta}\|^2 \rightarrow \text{loss function}$$

(objective function)

$$= \frac{1}{2m} \sum (y^{(i)} - \theta_0 - \theta_1 x^{(i)})^2 \quad \text{or cost function}$$

$$\rightarrow \text{Eg. } J(\theta) = \theta_1^2 + \theta_2^2$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = 2\theta_1 = 0 \Rightarrow \theta_1 = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_2} = 2\theta_2 = 0 \Rightarrow \theta_2 = 0$$

For min

$$\rightarrow \nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} = \underline{0} \quad (\text{zero vector})$$

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

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{2m} \sum 2(y^{(i)} - \theta_0 - \theta_1 x^{(i)}) \cdot (-1) = -\frac{1}{m} \sum (y^{(i)} - \theta_0 - \theta_1 x^{(i)})$$

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

→ Taking  $m = 2$

$$J(\theta) = \frac{1}{2(m)} \left[ (y^{(1)} - (\theta_0 + \theta_1 x^{(1)}))^2 + (y^{(2)} - (\theta_0 + \theta_1 x^{(2)}))^2 \right]$$

$$\frac{\partial J(\theta)}{\theta_0} = \frac{-1}{2} \left[ [y^{(1)} - (\theta_0 + \theta_1 x^{(1)})] + [y^{(2)} - (\theta_0 + \theta_1 x^{(2)})] \right]$$

$$= -\frac{1}{2} \left[ \sum_{i=1}^2 (y^{(i)} - \theta_0 - \theta_1 x^{(i)}) \right]$$

$$\frac{\partial J(\theta)}{\theta_1} = -\frac{1}{2} \left[ (y^{(1)} - (\theta_0 + \theta_1 x^{(1)})) x^{(1)} + (y^{(2)} - (\theta_0 + \theta_1 x^{(2)})) x^{(2)} \right]$$

$$= -\frac{1}{2} \left[ \sum_{i=1}^2 (y^{(i)} - \theta_0 - \theta_1 x^{(i)}) x^{(i)} \right]$$

In general,

$$\nabla J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m (y^{(i)} - (\theta_0 + \theta_1 x^{(i)})) \right. \\ \left. \sum_{i=1}^m (y^{(i)} - (\theta_0 + \theta_1 x^{(i)})) x^{(i)} \right]$$

$$= -\frac{1}{m} \begin{bmatrix} 1 & x^{(1)} & \dots & x^{(m)} \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} - (\theta_0 + \theta_1 x^{(1)}) \\ - (\theta_0 + \theta_1 x^{(2)})$$

$$\theta = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(m)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \end{bmatrix} = -\frac{1}{m} \begin{bmatrix} x^T (y - x\theta) \end{bmatrix} = 0$$

$\downarrow \quad \downarrow \quad \downarrow$   
 $m \times 1 \quad m \times 1 \rightarrow 2 \times 1$

Hence  $x^T y = x^T x \theta$  gives  $\theta = \underline{(x^T x)^{-1} x^T y}$   
pseudo inverse of  $x$

Here we have  $m$  equations and 2 unknowns  
 This is overdetermined set of equations

$y = x\theta \Rightarrow \theta = x^{-1} y$  but this works  
 only when  $x$  is a square matrix, but that's  
 not the case here. Hence use pseudo inverse

→ Suppose we had more number of features:  
 $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$ .

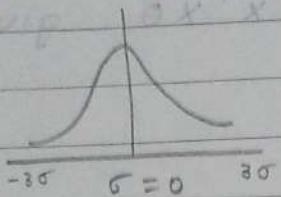
$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \quad x^{(i)} = \begin{bmatrix} 1 \\ x_{1(i)} \\ x_{2(i)} \\ x_{3(i)} \end{bmatrix} \quad h_\theta(x^{(i)}) = \theta^T x^{(i)}$$

### Probabilistic interpretation of LR

→ Earlier we considered  $e^{(i)}$  as error between  $y^{(i)}$  and  $h_\theta(x^{(i)})$ . Consider  $e^{(i)}$  as a R.V.  
 Assume that  $e^{(i)}$  is Gaussian distributed.  
 $N(0, \sigma^2)$  Here  $f(e^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{e^{(i)2}}{2\sigma^2}}$

$$\text{Consider } y^{(i)} = \theta_0 + \theta_1 x^{(i)} + \epsilon^{(i)}$$

Error is Gaussian distributed. So value drawn everytime is random but its closely in surrounding of the mean



EQ. Given  $\epsilon^{(i)}$  is gaussian. What is probability of  $y^{(i)}$  given  $x^{(i)}$

given  $\sigma$  constant

$$y^{(i)} = \theta_0 + \theta_1 x^{(i)} + \epsilon^{(i)}$$

$P(y^{(i)} | x^{(i)}; \theta)$  has the form of Gaussian

$$N(\theta^T x^{(i)}, \sigma^2).$$

shifted mean by  $\theta^T x^{(i)}$

(By adding only mean changes, variance same)

$$P(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}}$$

x parameterized by  $\theta$ .

→ We went to solve LR using probabilistic interpretation, one of the approach to formulate the problem is to seek those  $\theta$ 's ( $\theta_0, \theta_1$ ) such that  $L(\theta) = P(y^{(1)}, y^{(2)}, \dots, y^{(m)} | x^{(1)}, x^{(2)}, \dots, x^{(m)}; \theta)$  is maximum.

(Choose such  $\theta$  that maximizes the chance of getting these  $y_i$  for given  $x_i$ ).

$L(\theta)$  is called the likelihood function.

- Let's consider  $\epsilon^{(i)}$ 's as iid with  $N(0, \sigma^2)$
- $y^{(1)} = \theta_0 + \theta_1 x^{(1)} + \epsilon^{(1)}$  independent
- $y^{(2)} = \theta_0 + \theta_1 x^{(2)} + \epsilon^{(2)}$  of each other

$$\begin{aligned} \text{Hence } L(\theta) &= P(y^{(1)} | x^{(1)}; \theta) P(y^{(2)} | x^{(2)}; \theta) \dots \\ &\quad P(y^{(m)} | x^{(m)}; \theta) \\ &= \prod_{i=1}^m P(y^{(i)} | x^{(i)}; \theta) \\ &= \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}} \end{aligned}$$

- So the problem can be formulated as follows

$$\max_{\theta} L(\theta) \quad (\text{Max possible value is 1})$$

- This formulation is called maximum likelihood estimation of  $\theta$  (MLE).

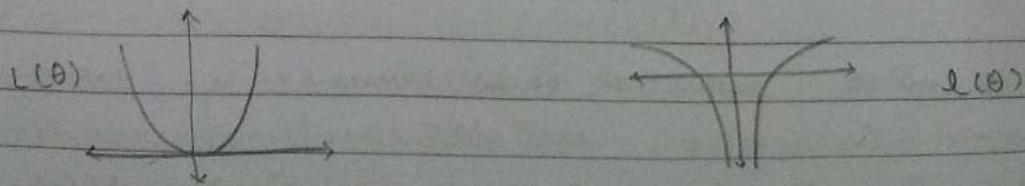
- Maximization can be simplified by taking  $l(\theta)$  instead of  $L(\theta)$  where  $l(\theta) = \log [L(\theta)]$ .

Reason for taking log

$$\rightarrow L(\theta) = \theta^2, \quad l(\theta)_{\min} = 0 \quad \text{at } \boxed{\theta = 0}$$

$$\rightarrow l(\theta) = \log(L(\theta)) \quad l(\theta)_{\min} = -\infty \quad \text{at } \boxed{\theta = 0}$$

Conclusion → we are only interested in  $\theta$  at which function maximizes/minimizes. This does not change by taking log. Hence fine.



$\theta^T x$  - line in linear regression  
(In general a hyperplane)

Page No.:

Date:

YOUNA

so here we will maximize  $l(\theta)$  in place of  $L(\theta)$  to simplify calculation.

$$l(\theta) = \sum_{i=1}^m \log \left[ \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}} \right]$$

$$= \sum_{i=1}^m \log \frac{1}{\sqrt{2\pi\sigma^2}} - \sum_{i=1}^m \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

↓

constant  $\propto$  independent of  $\theta$

→ so what we need is ...

$$\max_{\theta} l(\theta) = \max_{\theta} - \sum_{i=1}^m \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \quad \text{Equivalently}$$

$$\min_{\theta} l(\theta) = \min_{\theta} \sum_{i=1}^m \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

$$\rightarrow \text{Estimated } \theta^* = \min_{\theta} l(\theta) = \min_{\theta} \sum_{i=1}^m \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$$

This minimization leads to least square estimation of  $\theta$  which was obtained earlier by pseudo inverse approach. (Similar expression with just  $\sigma^2$  in place of  $m$ )  
Hence MLE method is equivalent to pseudo inverse method. (least square approach).

Note that the noise error here is Gaussian iid  $N(0, \sigma^2)$ . If this changes then the expression changes.

- White noise - mean of each element = 0, variance of each element is finite, all elements are uncorrelated
- iid noise - white + all elements independent also.

For non gaussian noise, MLE is not equivalent to least square. MLE depends on  $P(\epsilon_i)$ . Since least square doesn't work here, use MLE with pdf of non Gaussian  $\epsilon_i$

Page No.:	youva
Date:	

- ★ What happens if the noise error is not gaussian?

Typical non Gaussian noise are very and skewed Gaussian noise. We must note that Gaussian noise is a precondition for using least square method. Using this method on non Gaussian noise gives wrong estimation (especially when noise levels  $> 5\%$ )

Modifying least mean square method for non Gaussian noise is under study

What happens if errors are not iid?

If errors are not iid, then we will not be able to write joint probability  $p(y^{(1)} - y^{(m)} | x^{(1)} - x^{(m)}; \theta)$  as product of individual probabilities. The distribution here now is jointly Gaussian.

$$p(x_1, x_2) = p(x_1|x_2) p(x_2) \rightarrow \text{Taking just two jointly Gaussian RVs}$$

$$\text{Here } p(x_1|x_2) = N(x_1 | u_{12}, \Sigma_{12})$$

$$p(x_2) = N(x_2 | u_2, \Sigma_{22})$$

As number of non iid RVs increases, it becomes extremely difficult to calculate the estimation manually

$$(u_{12} = u_1 + \Sigma_{12} \Sigma_{22}^{-1} (x_2 - u_2))$$

$$\Sigma_{12} = \Sigma_1 \Sigma_{22} = \Sigma_1 - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21})$$

→ sometimes, some features are more important than other in  $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$ . Here, we can get better estimate of  $\theta$  if we use prior knowledge about  $\theta$ . This also gives better fit for given data. i.e. a better model / hypothesis.). This method is called regularisation in ML. In probabilistic terms it's called maximum a posteriori (MAP) estimation of  $\theta$ . For regression problem is referred to as ridge regression.

Here, we need some prior knowledge of  $\theta$ . Generally we consider  $\theta$  also as a distribution mostly Gaussian.

Eg. Face recognition - Image as a matrix of pixels. Now we know that there is huge change in intensity when there is an edge. Around cheeks, same intensity. So we have some idea from beginning.

- When we know something about what could our answer  $\theta$  be, we will get better results. So clearly MAP is better than MLE
- For MLE we maximize  $L(\theta) = P(y^{(1)}, y^{(2)}, \dots, y^{(m)} | x^{(1)}, x^{(2)}, \dots, x^{(m)}; \theta)$

→ For MAP, problem is formulated as follows:

$$\text{Max } L(\theta) = \text{Max } P(y^{(1)}, y^{(2)}, \dots, y^{(m)}, x^{(1)}, x^{(2)}, \dots, x^{(m)} | \theta) P(\theta)$$

$$\textcircled{1} \quad \frac{\partial}{\partial \theta} \quad \text{Max } P(y^{(1)}, y^{(2)}, \dots, y^{(m)}, x^{(1)}, x^{(2)}, \dots, x^{(m)})$$

Using Bayes Rule  $P(A|B) P(B) = P(B|A) P(A)$   
we get --

$$\underset{\theta}{\operatorname{Max}} L(\theta) = \underset{\theta}{\operatorname{Max}} P(\theta | y^{(1)} \dots y^{(m)}, x^{(1)} \dots x^{(m)})$$

→ So this means we are trying to maximize the probability of getting a particular  $\theta$  given the data we have. so the  $\theta$  corresponding to the max value of probability is the answer.

→ From ①, we can ignore denominator as it is independent of  $\theta$  so maximisation is --

$$\underset{\theta}{\operatorname{Max}} L(\theta) = \underset{\theta}{\operatorname{Max}} p(y^{(1)} \dots y^{(m)}, x^{(1)} \dots x^{(m)} | \theta) \propto p(\theta)$$

$y$  — outputs (real numbers)

$x$  — feature (maybe a feature vector also)

→ Now  $y_i, x_i$  are dependent on each other but each pair of  $y_i, x_i$  are independent of one another.

$$L(\theta) = \left[ \prod_{i=1}^m p(y^{(i)}, x^{(i)} | \theta) \right] p(\theta)$$

$$= \left[ \prod_{i=1}^m p(y^{(i)} | x^{(i)}, \theta) p(x^{(i)} | \theta) \right] p(\theta)$$

Prior probability that we are trying to use

→ Note that  $\theta$  and  $x^{(i)}$ 's are independent so  $p(x^{(i)} | \theta) = p(x^{(i)})$  and this is now not dependent on  $\theta$ . so -

$$L(\theta) = \left[ \prod_{i=1}^m p(y^{(i)} | x^{(i)}, \theta) \right] p(\theta)$$

→ so here we consider  $\theta$  also as a RV and take it as Gaussian distributed  $N(0, \tau^2)$   
 Also we consider multiple  $\theta$  as iid.

$$P(\theta) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\theta^2/2\tau^2} \rightarrow \text{for each of } \theta_0, \theta_1, \dots, \theta_n$$

we actually need  $P(\theta_0, \theta_1, \dots, \theta_n)$  But  
 since iid, it is  $P(\theta_0) P(\theta_1) \dots P(\theta_n)$   
 which is  $P(\theta) = \left(\frac{1}{\sqrt{2\pi\tau^2}}\right)^{n+1} e^{-\|\theta\|^2/2\tau^2}$

(Here  $\theta$  is the entire vector and  $\|\theta\|^2 = \theta_0^2 + \theta_1^2 + \dots + \theta_n^2$ )

$$L(\theta) = \prod_{i=1}^m \left[ \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\tau^2}} \right] \left( \frac{1}{\sqrt{2\pi\tau^2}} \right)^{n+1} e^{-\frac{\|\theta\|^2}{2\tau^2}}$$

We ignore the constants which are independent of  $\theta$  and then take log before maximizing

$$L(\theta) = \prod_{i=1}^m \left[ e^{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\tau^2}} \right] e^{-\frac{\|\theta\|^2}{2\tau^2}}$$

$$l(\theta) = \sum_{i=1}^m \left[ -\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\tau^2} \right] - \frac{\|\theta\|^2}{2\tau^2}$$

$$\max_{\theta} l(\theta) = \max_{\theta} - \left[ \sum_{i=1}^m \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\tau^2} + \frac{\|\theta\|^2}{2\tau^2} \right]$$

$$\theta^* = \min_{\theta} \left[ \frac{1}{2\tau^2} \sum (y^{(i)} - \theta^T x^{(i)})^2 + \frac{1}{2\tau^2} \|\theta\|^2 \right]$$

Objective function

This means,  $\theta^*$  is that value of  $\theta$  that minimizes the objective function. This is regularization

→ Note that  $\frac{\|\theta\|^2}{2\tau^2}$  is the additional term for MAP from MLE

→  $\sigma^2$  - variance of errors

$\tau^2$  - variance of  $\theta$  (prior knowledge)

$$\rightarrow \theta^* = \min_{\theta} \frac{1}{2\sigma^2} \left[ \sum (y^{(i)} - \theta^T x^{(i)})^2 + \frac{\sigma^2}{\tau^2} \|\theta\|^2 \right]$$

regularisation parameter  $\lambda$

(Note that estimating  $\lambda$  is also a problem within itself. Here, instead we simply assume it)

→ so we minimize the following for MAP

$$L(\theta) = \frac{1}{2\sigma^2} \left[ \sum (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \|\theta\|^2 \right]$$

Hoping to get a close formed solution, we differentiate this w.r.t  $\theta$  and equate to 0.

partial differential w.r.t  $\theta_0, \theta_1, \dots, \theta_n$

$$\text{so we derive } \sum (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \|\theta\|^2$$

↓

This is equivalently  $\|y - x\theta\|^2$

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$x = \begin{bmatrix} 1 & x^{(1)} & \dots & (x^{(1)})^n \\ 1 & x^{(2)} & & (x^{(2)})^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x^{(m)} & \dots & (x^{(m)})^n \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

m × 1

m × (n+1)

(n+1) × 1

↓

Each row is  $x^{(i)}$

$$\star \quad \|y - x\theta\|^2$$

$\downarrow d$

$$-2x^T [y - x\theta] \quad (n+1) \quad \downarrow \lambda \theta \quad (n+1)$$

$$\rightarrow \text{Hence } -x^T y + x^T x \underline{\theta} + \lambda \underline{\theta} = \underline{0} \quad (\text{zero vector})$$

$$x^T y = (x^T x + \lambda I) \underline{\theta}$$

max unit vector

$$\rightarrow \text{Hence } \underline{\theta}^* = (x^T x + \lambda I)^{-1} x^T y \rightarrow \text{MAP}$$

additional term : better estimate of  $\theta$

$\lambda \Rightarrow$  better solution

In real life  $n$  is of the order of  $10^4 - 10^6$   
 In this case finding inverse is almost impossible due to computational inefficiency.

In such cases we use iterative approach —  
 gradient descent.

$$\star \quad J(\theta) = \sum (y^{(i)} - (\theta_0 + \theta_1 x_1^{(i)} + \theta_2 (x_2^{(i)})^2 + \dots + \theta_n (x_n^{(i)})^n))^2$$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} = \begin{bmatrix} -2 \sum ( ) \\ -2 \sum ( ) x_1^{(i)} \\ \vdots \\ -2 \sum ( ) (x_n^{(i)})^n \end{bmatrix}$$

Q Under what conditions MLE and MAP are equal?

Aw. When  $\theta$  is a uniform distribution i.e. when  $P(\theta)$  is constant, MAP will become equivalent to MLE as  $L(\theta)$  will no longer depend on  $P(\theta)$ .

→ Types of gradient descent : stochastic, moment based etc

→  $J(\theta)$  used while minimizing the objective func.  
was  $J(\theta) = \frac{1}{2m} \sum [y^{(i)} - (\theta_0 + \theta_1 x^{(i)})]^2$

→ By using G.D.  $\theta_0$  and  $\theta_1$  can be computed as follows :

$$\begin{aligned} \theta_0^{k+1} &= \theta_0^k - \frac{\alpha \frac{\partial J(\theta)}{\partial \theta_0}}{\quad \quad \quad \theta_0 = \theta_0^k} & \alpha \text{ is the learning rate (+ve)} \\ \theta_1^{k+1} &= \theta_1^k - \frac{\alpha \frac{\partial J(\theta)}{\partial \theta_1}}{\quad \quad \quad \theta_1 = \theta_1^k} \end{aligned}$$

(Here  $k$  represents the iteration number)

$$\begin{aligned} \frac{\partial J}{\partial \theta_0} &= \frac{1}{2m} \sum_{i=1}^m 2(y^{(i)} - \theta_0 - \theta_1 x^{(i)}) (-1) \\ &= -\frac{1}{m} \sum_{i=1}^m [y^{(i)} - (\theta_0 + \theta_1 x^{(i)})] \end{aligned}$$

$$\frac{\partial J}{\partial \theta_1} = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} - (\theta_0 + \theta_1 x^{(i)})] x^{(i)}$$

→ Assume some initial parameter values  $\theta_0^0, \theta_1^0$

→ keep using (1) iteratively. stop the iteration when algorithm converges at  $\|\theta^{k+1} - \theta^k\|^2 < \text{small value}$

- If we use entire set of examples, then it is called batch gradient descent. ( $\sum$  over  $i \in m$ )
- In stochastic GD we do not use summation we use randomly any one example in one iteration. Even then algorithm converges. This saves a lot of computation (SGD)
- Most important assumptions -  $J(\theta)$  is differentiable (However there are techniques to work with non-differentiable functions as well - genetic approach).
- Problem with GD - we might get stuck at local min and not reach global min. However stochastic GD helps solve this problem in majority cases.

### GD Algo --

Given  $J(\theta)$  which is differentiable

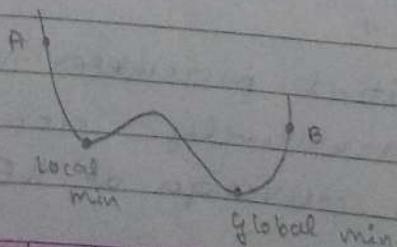
Write  $(k+1)^{th}$  iteration  $\theta$  values as,

$$\theta^{k+1} = \theta^k - \alpha \nabla J(\theta) \Big|_{\theta=\theta^k}$$

$$\alpha > 0$$

where  $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix}$$



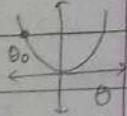
Initial A gives incorrect ans

Initial B give correct ans.

Algorithm stops when  $\nabla J(\theta)$  is  $\approx 0$   
 (This is what we usually do to find min  
 - derivate and equate with 0).

EQ  $J(\theta) = \theta^2$  Try to find  $\theta$  when  $J(\theta)$  is minimum.

$$J(\theta) \quad J(\theta_0 + s) = J(\theta_0) + \frac{dJ}{d\theta} \Big|_{\theta=\theta_0} s + \text{higher order terms.}$$



We try to approximate the function with a line at  $\theta_0$ . Take  $\theta = \theta_0 + s$ .

$$\begin{aligned} J(\theta) &= J(\theta_0) + 2\theta_0 (\theta - \theta_0) \quad \text{Ignore higher order terms.} \\ &= 4 + 2(2)(\theta - 2) \quad (\theta_0 = 2) \\ &= -4 + 4\theta \end{aligned}$$

so this is approximated by a line with slope = 4 and intercept = -4.

Choose  $s = -\alpha \frac{dJ}{d\theta} \Big|_{\theta=\theta_0}, \alpha > 0$

$$\frac{dJ}{d\theta} \Big|_{\theta=\theta_0} = \frac{dJ}{d\theta_0}$$

$$J(\theta_0 + s) = J(\theta_0) + \frac{dJ}{d\theta_0} \left( -\alpha \frac{dJ}{d\theta_0} \right).$$

always -ve  
 so  $J(\theta_0 + s) < J(\theta_0)$

As this continues, we go to smaller and smaller values and hence we are guaranteed to get a min.

$$J(\theta_0^{k+1}) = J(\theta_0^k) - \alpha \left( \frac{dJ}{d\theta} \Big|_{\theta=\theta_0^k} \right)^2$$

$$\alpha = 1. \quad \text{Eg 1} \quad \text{Take } k=0 \quad \theta^0 = 2. \quad J(\theta) = 4$$

$$k=1 \quad \theta^1 = -2 \quad J(\theta) = 4$$

$$k=2 \quad \theta^2 = 2 \quad J(\theta) = 4$$

$$\alpha = 10^2. \quad \text{Eg 2.} \quad \text{Take } k=0 \quad \theta^0 = 2 \quad J(\theta) = 4$$

$$k=1 \quad \theta^1 = -2.8 \quad J(\theta) = 7.64$$

$$k=2 \quad \theta^2 = 3.92 \quad J(\theta) = 15.3664$$

$$\alpha = 0.1. \quad \text{Eg 3} \quad \text{Take } k=0 \quad \theta^0 = 2 \quad J(\theta) = 4$$

$$k=1 \quad \theta^1 = 1.6 \quad J(\theta) = 2.56$$

$$k=2 \quad \theta^2 = 1.28 \quad J(\theta) = 1.6384$$

Here used :  $\theta^{k+1} = \theta^k - \alpha \frac{\partial J}{\partial \theta} \Big|_{\theta=\theta^k} = 2\theta^k$

- This shows that we must choose  $\alpha$  wisely
- Large  $\alpha$  - function diverges
- Small  $\alpha$  - function converges but slowly.  
(i.e. takes more number of iteration to converge. i.e. higher training time)
- Steepest descent choose such  $\alpha$  that we get faster convergence.

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

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{2m} \sum_{i=1}^m [y^{(i)} - (\theta_0 + \theta_1 x^{(i)})] (-1).$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{2m} \sum_{i=1}^m [y^{(i)} - (\theta_0 + \theta_1 x^{(i)})] (-x^{(i)})$$

and so many derivatives go on if we have a lot no. of parameters. Not practically feasible.

→ In steepest descent optimisation, we take  $\alpha^*$  (this is  $\alpha$  at  $k^{th}$  iteration). Choose  $\alpha$  such that  $J(\theta^* - \alpha \nabla J(\theta))$  is minimized over  $\alpha$ . next value evaluated at  $\theta = \theta^*$  of  $\theta^*$  i.e.  $\theta'$  and so on for every successive  $\theta^k$ .

→ choose some initial  $\theta^0$  (i.e.  $\theta_0^0$  and  $\theta_1^0$ ) and find  $\nabla J(\theta)|_{\theta=\theta^0} = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} & | \theta_0 = \theta_0^0 \\ \frac{\partial J}{\partial \theta_1} & | \theta_1 = \theta_1^0 \end{bmatrix}$  and put

in  $\theta^0 - \alpha \nabla J(\theta)|_{\theta=\theta^0}$  to then get  $J$  as a function of  $\alpha$  and minimize it over  $\alpha$

$$\alpha^{k+1} = \alpha^k - \frac{J'(\alpha)}{J''(\alpha)} \quad | \quad \alpha = \alpha^k \quad \text{--- (1)}$$

Note that though here we have iterations for finding  $\alpha$  for each iteration to find  $\theta$ , still this is faster than GD.

→ This is called Newton's method. (and we do not have any learning rate here)

The expression (1) comes as

→ Use Taylor series expression of  $J(\alpha)$

$$J(\alpha) = J(\alpha_0) + \frac{d}{d\alpha} J(\alpha) \Big|_{\alpha=\alpha_0} (\alpha - \alpha_0) + \frac{1}{2!} \frac{d^2}{d\alpha^2} J(\alpha) \Big|_{\alpha=\alpha_0} (\alpha - \alpha_0)^2 \quad \text{(use 2nd order approximation)}$$

$\frac{d J(\alpha)}{d\alpha} = 0$  gives minimum

$$J'(\alpha) \Big|_{\alpha=\alpha_0} + \frac{J''(\alpha)}{2} \Big|_{\alpha=\alpha_0} 2(\alpha - \alpha_0) = 0.$$

$$\alpha = \alpha_0 - \frac{J'(\alpha_0)}{J''(\alpha_0)}$$

Q. When  $\theta$  has multiple parameter, lets say  $\theta_0$  and  $\theta_1$ , why do we do simultaneous update and not minimize each of  $\theta_0$  and  $\theta_1$  separately?

In case of  $J(\theta) = \theta^2$ , we approximated the function with a line because we had just one parameter  $\theta$ .

Now, if we had something of the form  $J(\theta_0, \theta_1, \dots, \theta_n)$ , then to approximate this at a point, we would need a hyperplane.

In earlier case, the line was completely defined by its slope, i.e. only one parameter was changing that one parameter we reach a new point.

But a hyperplane is defined by multiple parameters. So to go from a point  $\theta^k$  to  $\theta^{k+1}$ , we need to change all of  $\theta_0, \dots, \theta_n$ , only then it makes sense of shifting.

Minimizing each of  $\theta_0, \dots, \theta_n$  does not make sense in terms of point shifting on hyperplane and hence gives no correct result. So we need to use SIMULTANEOUS UPDATE.

→ Newton's method for single dimensional case  
 $\alpha$  is as follows

$$\alpha^{k+1} = \alpha^k - \frac{f'(\alpha)}{f''(\alpha)} \Big|_{\alpha=\alpha^k}$$

→ Newton's method for multi dimensional case  
 $\alpha$  is as follows:

$$\text{Let } \alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix}$$

$$\alpha^{k+1} = \alpha^k - [Hf(\alpha)]^{-1} \nabla f(\alpha) \Big|_{\alpha=\alpha^k}$$

(Replace derivative by gradient and reciprocal of double derivative with inverse of Hessian matrix.)

$$\nabla f(\alpha) = \begin{bmatrix} \frac{\partial f}{\partial \alpha_0} \\ \frac{\partial f}{\partial \alpha_1} \end{bmatrix} \quad \text{at } \alpha_0 = \alpha_0^k, \alpha_1 = \alpha_1^k$$

$$Hf(\alpha) = \begin{bmatrix} \frac{\partial^2 f}{\partial \alpha_0^2} & \frac{\partial^2 f}{\partial \alpha_0 \partial \alpha_1} \\ \frac{\partial^2 f}{\partial \alpha_1 \partial \alpha_0} & \frac{\partial^2 f}{\partial \alpha_1^2} \end{bmatrix} \quad \text{at } \alpha_0 = \alpha_0^k, \alpha_1 = \alpha_1^k$$

EQ Minimize  $x^4 + 2x^2y^2 + y^4$ .

E  $x = \begin{bmatrix} x \\ y \end{bmatrix}$

$$x^{k+1} = x^k - [H_f(x)]^{-1} \nabla f(x)$$

$$-\nabla f(x) = H_f(x) (x^{k+1} - x^k)$$

Assume start point  $(a, a)$

$$\nabla f(x) = \begin{bmatrix} 4x^3 + 4xy^2 \\ 4x^2y + 4y^3 \end{bmatrix}$$

$$H_f(x) = \begin{bmatrix} 12x^2 + 4y^2 & 8xy \\ 8xy & 4x^2 + 12y^2 \end{bmatrix}$$

$$\begin{bmatrix} -8a^3 \\ -8a^3 \end{bmatrix} = \begin{bmatrix} 16a^2 & 8a^2 \\ 8a^2 & 16a^2 \end{bmatrix} \begin{bmatrix} a-a \\ a-a \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{2}{3} \begin{bmatrix} a \\ a \end{bmatrix} \quad \text{is the solution for first iteration}$$

In general for  $k^{\text{th}}$  iteration  
it is  $\left(\frac{2}{3}\right)^k \begin{bmatrix} a \\ a \end{bmatrix}$

## Polynomial Regression

line ( $n=1$  so just 2 parameters)

$$\rightarrow h_0(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \dots + \theta_n x_n^{(i)}$$

hyperplane =  $[\theta_0 \ \theta_1 \ \theta_2 \ \dots \ \theta_n] \begin{bmatrix} 1 \\ x_1^{(i)} \\ 1 \\ \vdots \\ x_n^{(i)} \end{bmatrix}$

$$= \Theta^T \underline{x^{(i)}} \quad (\text{For linear regression})$$

→ Now, for polynomial regression . . .

$$h_0(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 \sqrt{x} + \dots$$



If  $x$  is size,  $h_0(x)$  is price depending on size

→ Here also, we aim to find the  $\theta$ s.

→ Dimension of  $X$  matrix in general is  
 $m \times (n+1)$

$$X = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \dots & (x_1)^n \\ 1 & x_2 & (x_2)^2 & \dots & (x_2)^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & (x_m)^2 & \dots & (x_m)^n \end{bmatrix} \quad m \times (n+1)$$

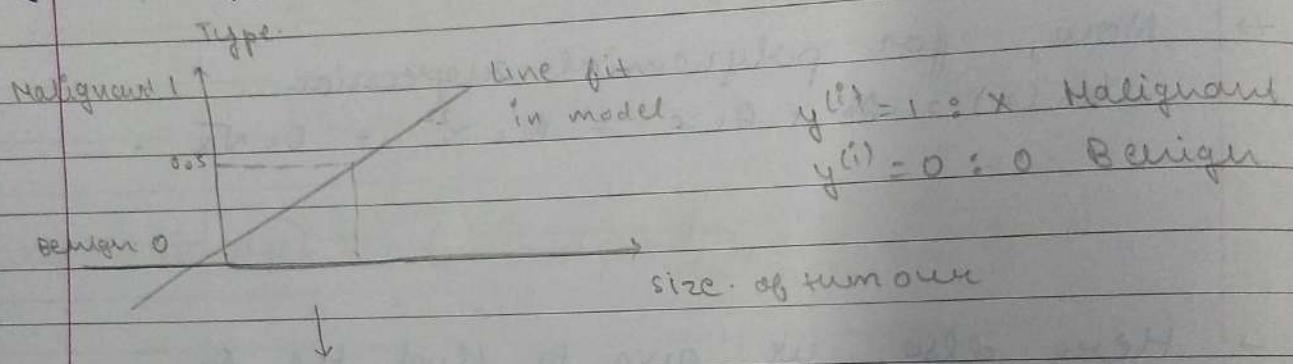
(Fit a model through data - fitting problem)

## Logistic Regression

- This is a supervised learning algorithm where  $\{x^{(i)}, y^{(i)}\}_{i=1}^{10m}$  are given.
- This is a binary classifier so  $y^{(i)}$  takes only 2 values 0 or 1

- $x^{(i)}$  could be scalar or vector.
- Thus we have linear regression, where we could have output from  $-\infty$  to  $\infty$ , here  $y$  is not a real number. It takes value from a fix set. Thus this is discrete whereas linear regression is continuous.

Eg. classify tumour as malignant or benign



Any point on right of model line denotes malignant tumor while on left of it denotes benign.  
(Fixed  $y$  as the midway point between 0 and 1).

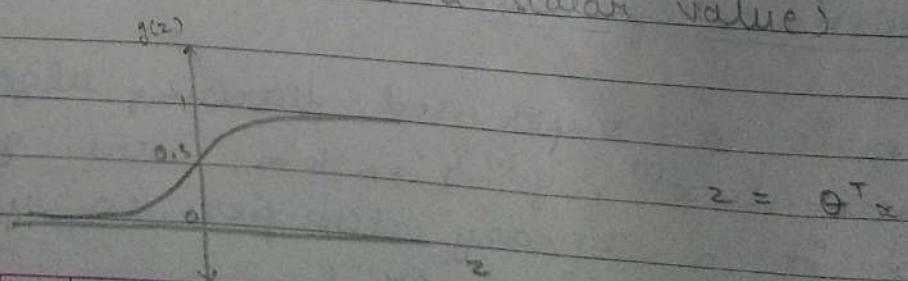
- We can use linear regression to plot this model fit line, but not a good idea as we do not need multiple possible outputs here. So the model is modified.

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

(Hypothesis)

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

$(\text{must be a scalar value})$  Here  $m=1$



- Let us consider  $h_\theta(x)$  as the estimated probability that  $y=1$  on input  $x$ .
  - For example --  $x = \begin{bmatrix} 1 \\ x_1 \end{bmatrix}$  → Tumor size (test example)
  - Eg  $h_\theta(x) = 0.7$  — This means patient has malignant cancer with probability 0.7 and has benign cancer with probability 0.3
  - We know that  $y=0$  or 1. So
- $$\underline{P(y=1 | x; \theta)} = 1 - P(y=0 | x; \theta)$$
- $h_\theta(x)$
- $$\rightarrow h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$
- $\theta^T x$

Suppose we know  $\theta_0 = -3$ ,  $\theta_1 = 1$ ,  $\theta_2 = 1$ .

$-3 + x_1 + x_2 \geq 0 \rightarrow$  belongs to one class  
(like malignant)

Decision boundary  $\leftarrow$  One class  $[\theta^T x \geq 0 : \text{ans} = 1]$

Other class  $[\theta^T x < 0 : \text{ans} = 0]$

$$-3 + x_1 + x_2 = 0 \quad (\theta^T x = 0)$$

- Note that here  $y$  takes only one of the 2 values : 0 or 1. Hence  $y$  is a Bernoulli Random Variable.
- If  $\theta_0 = 0$ , the decision boundary passes through  $(0, 0)$ . ( $y$  intercept = 0)
- If  $\theta^T x$  turns up to be = 0, toss a coin and decide what to classify it as
- Decision boundary may not always be linear.

class 1

class 2

→ Here we had supposed  $\theta_0, \theta_1, \theta_2$ . Now we will see how to estimate these values actually. We will use maximum likelihood estimation.

→ Assume that  $P(Y=1 | x; \theta) = h_\theta(x)$

$$P(Y=0 | x; \theta) = 1 - h_\theta(x)$$

→ This means  $P(Y=y | x; \theta) = (h_\theta(x))^y (1-h_\theta(x))^{1-y}$

$$\rightarrow L(\theta) = P(y^{(1)}, y^{(2)}, \dots, y^{(m)} | x^{(1)}, x^{(2)}, \dots, x^{(m)}; \theta)$$

$$\rightarrow \theta^* = \max_{\theta} L(\theta) = \max_{\theta} P(y^{(1)}, y^{(2)}, \dots, y^{(m)} | x^{(1)}, x^{(2)}, \dots, x^{(m)}; \theta)$$

Here  $y_1, \dots, y_m$  are independent of each other once  $x_1, \dots, x_m$  are given.

$$\rightarrow \theta^* = \max_{\theta} P(y^{(1)} | x^{(1)}; \theta) \cdots P(y^{(m)} | x^{(m)}; \theta)$$

$$\rightarrow L(\theta) = \prod_{i=1}^m P(y^{(i)} | x^{(i)}; \theta) \quad \text{m training examples}$$

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

$$\rightarrow L(\theta) = \log(L(\theta))$$

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

Note : When there are probability terms in likelihood function, it's better to use  $\log$  because since probability is between 0 and 1, multiplying m such values gives a very small number, and hence difficult to handle by computer. So take  $\log$  and convert multiply into addition.

→ Maximization can be done using gradient ascent

→ We instead take a minus sign and minimize only.

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



This is called binary cross entropy



$$H = -\sum p_i \log p_i$$

(max when  $p_i = 0.5$ )

→ It's called cross entropy because here we are using 2 probabilities: one actual  $y^{(i)}$  and other estimated of  $h_\theta(x^{(i)})$ .

→ Entropy basically means randomness. Hence, we are essentially trying to minimize this randomness between actual and expected output.

$$\rightarrow g(z) = \frac{1}{1+e^{-z}} \quad \text{(A)}$$

$$\rightarrow g'(z) = -\left(\frac{1}{1+e^{-z}}\right)^2 (-e^{-z}) = \left(\frac{1}{1+e^{-z}}\right)^2 (e^{-z})$$

$$\begin{aligned} g'(z) &= \left(\frac{1}{1+e^{-z}}\right) \left(1 - \frac{1}{1+e^{-z}}\right) \\ &= g(z) (1 - g(z)) \end{aligned}$$

$$\text{Now } g'(\theta) = g(z) (1 - g(z)) \frac{dz}{d\theta}$$

→ We use gradient descent to find  $\theta_s$

$$\theta_0^{(k+1)} = \theta_0^{(k)} - \alpha \frac{\partial J(\theta)}{\partial \theta_0} \Big|_{\theta=\theta^{(k)}}$$

$$\theta_1^{(k+1)} = \theta_1^{(k)} - \alpha \frac{\partial J(\theta)}{\partial \theta_1} \Big|_{\theta=\theta^{(k)}}$$

∴ so on till  $\theta_n^{(k+1)}$

Simultaneous update is important because here we will have a hyperplane. Each point on hyperplane is defined by all parameters. So to go from one point to other, we need to change all parameters at a time).

If we use just one training example instead of  $\sum_{i=1}^m$ , we come to stochastic gradient descent

$$J(\theta) = - \left[ y \log(h_\theta(x)) + (1-y) \log(1-h_\theta(x)) \right]$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = - \left[ y \frac{1}{h_\theta(x)} - (1-y) \frac{1}{1-h_\theta(x)} \right] \frac{\partial h_\theta(x)}{\partial \theta_j} \quad \text{from (A)}$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = -(y(1-h_\theta(x)) - (1-y)h_\theta(x)) x_j$$

$$\begin{aligned} &= -(y - yh_\theta(x) - h_\theta(x) + yh_\theta(x)) x_j \\ &= -(y - h_\theta(x)) x_j. \end{aligned}$$

$$\rightarrow \theta_0^{(k+1)} = \theta_0^{(k)} + \alpha \sum_{i=1}^m (y^{(i)} - h_\theta(x^{(i)})) x_j \quad \text{for } \theta_0$$

at  $\theta = \theta^{(k)}$

Here we evaluate at  $\theta = \theta^k$  and not  $\theta = \theta_0$ ,  
because  $h_\theta(x^{(i)})$  depends on all  $\theta_0, \theta_1, \dots$   
as it is  $\frac{1}{1 + e^{-\theta^T x}}$

$$\theta_j^{(k+1)} = \theta_j^{(k)} + \alpha [y^{(i)} - h_\theta(x^{(i)})] x_{j(i)}$$

! and so on  $\theta_j^{(k+1)} = \theta_j^{(k)} + \alpha (y^{(i)} - h_\theta(x^{(i)})) x_j$

→ This was all about binary classification.  
Now we use multiclass classification.

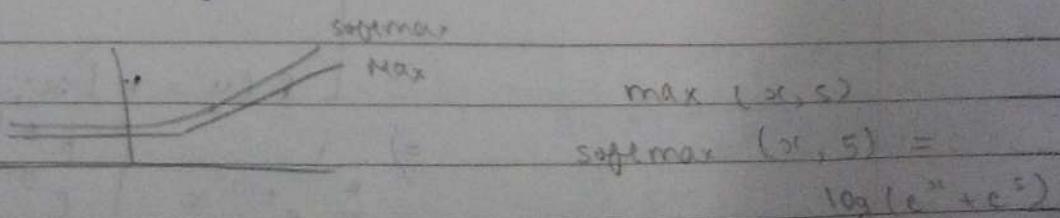
### Multiclass classification

Ex. Consider 3 classes : 1, 2, 3 ...

Combination	1 2, 3	$h_{\theta_1}(x) = 0.9$	→ highest so classified in class 1
	2 1 3	$h_{\theta_2}(x) = 0.5$	
	3 1, 2	$h_{\theta_3}(x) = 0.8$	

This is essentially binary classification of  
one vs all

→ Sometimes we also use softmax classifier  
 $\text{Max}(R_1, \dots, R_n) : R^n \rightarrow R$  (Not differentiable)  
 Softmax  $\rightarrow$  differentiable version of Max



$$\text{softmax}(x_1, x_2) = \log(e^{x_1} + e^{x_2})$$

(s)

$$\frac{ds}{dx_1} = \frac{e^{x_1}}{e^{x_1} + e^{x_2}}$$

$$\frac{ds}{dx_2} = \frac{e^{x_2}}{e^{x_1} + e^{x_2}}$$

- Addition of all these = 1 hence we can use this as probability.
- softmax is also called multinomial logistic regression / generalised logistic regression
- In logistic regression  $y_i \in \{0, 1\}$ . We learn  $n+1$  parameters
- For softmax classification,  $y$  can take  $K$  different values ( $K = \text{no. of classes}$ )
- Eg: MNIST data set.  $K = 10$  to recognise 10 digits. Each digit is an image of size  $28 \times 28 = 784$ . Thus we have  $\theta_0$  to  $\theta_{784}$ , i.e. 785 parameters.
- Given a test example  $x$ , we want our hypothesis to estimate  $P(y=k|x)$  for each  $k=1, 2, \dots, K$  and classify  $K$  as the one with highest probabilities.
- Hypothesis has to be such that sum of probabilities = 1.
- Hence  $h_\theta(x) = \begin{bmatrix} P(y=1|x; \theta) \\ P(y=2|x; \theta) \\ \vdots \\ P(y=K|x; \theta) \end{bmatrix}$ 
 $= \frac{1}{\sum_{j=1}^K e^{\theta^{(j)\top} x}} \begin{bmatrix} e^{\theta^{(1)\top} x} \\ e^{\theta^{(2)\top} x} \\ \vdots \\ e^{\theta^{(K)\top} x} \end{bmatrix}$

$$\rightarrow P(y=1 | x; \theta) = \frac{e^{\theta^{(1)\top} x}}{e^{\theta^{(1)\top} x} + e^{\theta^{(2)\top} x}} \quad (\text{considering only 2 classes})$$

Here  $\theta^{(1)\top} = [\theta_0^{(1)} \dots \theta_n^{(1)}]$

$$x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$\rightarrow$  For digit classification,  $K = 10$  and 785 parameters for each class. so we need to learn 7850  $\theta$ s.

$$\rightarrow \theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)} \in \mathbb{R}^{n+1}$$

$$\rightarrow \theta = (n+1) [\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)}]$$

(length of each column)

$\rightarrow$  For binary classifier, we learnt  $n+1$  parameters. Here  $K = 2$ . So we learnt  $(n+1)(K-1)$  number of parameters. (So actually for digit classification we'll need  $(785)(9)$  parameters only).

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

we need to maximize this.

$$\star \rightarrow l(\theta) = \sum_{i=1}^m \sum_{k=1}^K I[y^{(i)} = k] \log \frac{e^{\theta^{(k)\top} x^{(i)}}}{\sum_{j=1}^K e^{\theta^{(j)\top} x^{(i)}}}$$

if true i.e.  $y^{(i)} = k$  then  $P(y^{(i)} = k | x; \theta)$

else  $= 0$ .

(Here  $K = 2$ )

- we'll maximize this function using gradient descent to chose those 7850 Os.

After training, output 10000000000 when 0 is applied, 01000000000 when 1 is applied and so on i.e. only one value quite high.

- Logistic regression - supervised learning (classification)
- $P(Y|X)$  - we model this as  $h_0(x)$

### Linear Discriminant Analysis (LDA)

- Also called density estimation technique
- Here we estimate  $P(X = x | Y = 0)$  and  $P(X = x | Y = 1)$  from  $x_i$  with  $i = 1 \dots m$  (given set of examples)
- Considering a K class problem in LDA
- $$P(Y=1 | X=x) = \frac{P(X=x | Y=1) P(Y=1)}{P(X=x)}$$

$P(X=x | Y=1)$  - class conditional probability  
 $P(Y=1)$  - prior probability  
 $P(X=x)$  - marginal probability.

(Consider example of only one class  $Y = k$ )

- In logistic regression, we were directly modelling  $P(Y=k | X=x)$ , but here we are modelling the data  $P(X=x | Y=k)$  itself as a gaussian.

Principle Component Analysis -  
Unsupervised learning - used for  
clustering

Page No.:	
Date:	youva

$$\rightarrow P(x = \infty) = P(x = \infty, y = 0) + P(x = \infty, y = 1)$$

(similar to  $f(x) = \int f(x, y) dy$ )

$$\rightarrow P(x = \infty) = P(x = \infty | y = 0) P(y = 0) + P(x = \infty | y = 1) P(y = 1).$$

(for a binary class).

$f_{ik}(x)$  - class conditional probability for  $k^{\text{th}}$  class

$\pi_k$  - prior probability of  $k^{\text{th}}$  class

( $K$  - Total number of classes)

$k$  - of particular class)

$\rightarrow$  generalising for  $K$  class classification.

$$P(y = k | x = \infty) = \frac{f_{ik}(x) \pi_k}{\sum_{k=1}^K f_{ik}(x) \pi_k}$$

$\rightarrow$  On decision boundary, we have

$$P(y = 1, x = \infty) = P(y = 2 | x = \infty), \text{ if } K = 1, 2.$$

(binary classification)

$$\rightarrow f_0(x) = P(x = \infty | y = 0)$$

$$f_1(x) = P(x = \infty | y = 1).$$

We model using multivariate Gaussian with a covariance matrix.

$$\rightarrow f_0(x) = \frac{1}{\sqrt{2\pi \sigma_x^2}} e^{-(x - m_0)^2 / 2\sigma_x^2}$$

$$\rightarrow f_{1,2}(x_1, x_2) = \frac{1}{(2\pi)^{1/2} |\Sigma|^{1/2}} e^{-\frac{1}{2} [(x - M) \Sigma^{-1} (x - M)^T]}$$

$$(x = [x_1, x_2], x - M = [x_1 - m_1, x_2 - m_2])$$

$$M = [m_1, m_2]$$

- $\Sigma$  is the covariance matrix
- $$\Sigma = \begin{bmatrix} \sigma_{x_1}^2 & \text{cov}(x_1, x_2) \\ \text{cov}(x_2, x_1) & \sigma_{x_2}^2 \end{bmatrix}$$
- symmetric
- with  $\text{cov}(x_1, x_2) = E[(x_1 - \mu_{x_1})(x_2 - \mu_{x_2})]$   
(A single value)
- Consider digit classification problem, so  
here 10 classes :  $k = 0$  to 9. Take an  
image and make a  $28 \times 28$  grid.  
Put all rows vertically to have a column  
with 784 entries. There are  $x_1, \dots, x_{784}$ ,  
so now we can calculate joint density given  
we know  $\Sigma$  matrix.

$$f(x_1, \dots, x_{784} | x_1, \dots, x_{784}) = \frac{1}{(2\pi)^{\frac{784}{2}} |\Sigma|^{1/2}} e^{-\frac{1}{2} [(x - \mu) \Sigma^{-1} (x - \mu)^T]}$$

$$\text{where } x = [x_1 \dots x_{784}]$$

$$\mu = [\mu_{x_1} \dots \mu_{x_{784}}]$$

- Suppose we take just digits 0 and 1,  
and suppose we have 50 zeroes written  
(the greater than 50, the better learning)

$x_1$	$x_2$		$x_{784}$
$x_{11}$	$x_{21}$		$x_{7841}$
$x_{12}$	$x_{22}$		$x_{7842}$
⋮	⋮		⋮
$x_1^{50}$	$x_2^{50}$		$x_{784}^{50}$

The first pixel value from all the images. This gives us a random variable. This way, all pixel values (pixel 1 to 784) are individually an RV

$$\Sigma = \begin{bmatrix} \sigma_{x_1}^2 \text{ cov}(x_1, x_2) & \dots & \text{cov}(x_1, x_{10}) \\ \vdots & \ddots & \vdots \\ \text{cov}(x_{10}, x_1) & \dots & \sigma_{x_{10}}^2 \end{bmatrix}_{784 \times 784}$$

Note  $\sigma_{x_i}^2 = E[(x_i - \bar{x}_i)^2]$  with  $\bar{x}_i$  as the average of first column.

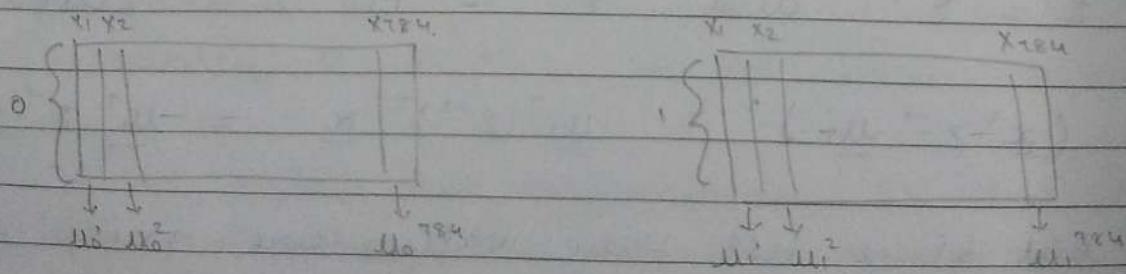
→ To find the decision boundary, we use the equiprobability  $P(Y=1 | X=x) = P(Y=0 | X=x)$

$$\rightarrow f_k(x) = P(X=x | Y=k) = \frac{1}{(2\pi)^{n/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2} [(x - \bar{u}_k)^T \Sigma_k^{-1} (x - \bar{u}_k)]}$$

→ Considering binary classification, for decision boundary.

$$\frac{f_0(x) \pi_0}{\sum_{k=0}^1 f_k(x) \pi_k} = \frac{f_1(x) \pi_1}{\sum_{k=0}^1 f_k(x) \pi_k}$$

→ Let  $\bar{u}_0$  represent the mean vector of class 0 and  $\Sigma_0$  represent the covariance matrix of class 0



(Mean of entire column)

$$f_1(x) \pi_1 = f_0(x) \pi_0$$

$$\frac{1}{(2\pi)^{n/2} |\Sigma_1|^{1/2}} e^{-\frac{1}{2} [(x - \bar{u}_1)^T \Sigma_1^{-1} (x - \bar{u}_1)]} \pi_1 = \frac{1}{(2\pi)^{n/2} |\Sigma_0|^{1/2}} e^{-\frac{1}{2} [(x - \bar{u}_0)^T \Sigma_0^{-1} (x - \bar{u}_0)]} \pi_0$$

Here 784 features, so decision boundary is a hyperplane

Page No.:  
Date: youva

- We assume  $\Sigma_0 = \Sigma_1 = \Sigma$  (not the case always).
- Using this and taking log both sides.

$$\begin{aligned} -\frac{1}{2} [\underline{x} - \underline{\mu}_1]^\top \Sigma^{-1} [\underline{x} - \underline{\mu}_1] + \log(\pi_1) &= \\ -\frac{1}{2} [\underline{x} - \underline{\mu}_0]^\top \Sigma^{-1} [\underline{x} - \underline{\mu}_0] + \log(\pi_0). \end{aligned}$$

Decision boundary is of the form  $\underline{B}^\top \underline{x} = 0$

$$\begin{aligned} -\frac{1}{2} [\underline{x} - \underline{\mu}_1]^\top \Sigma^{-1} [\underline{x} - \underline{\mu}_1] + \frac{1}{2} [\underline{x} - \underline{\mu}_0]^\top \Sigma^{-1} [\underline{x} - \underline{\mu}_0] + \log\left(\frac{\pi_1}{\pi_0}\right) &= 0 \end{aligned}$$

Equal so can be combined

$$\begin{aligned} \frac{-1}{2} \underline{x}^\top \Sigma^{-1} \underline{x} + \frac{1}{2} \underline{x}^\top \Sigma^{-1} \underline{\mu}_1 + \frac{1}{2} \underline{\mu}_1^\top \Sigma^{-1} \underline{x} - \frac{1}{2} \underline{\mu}_1^\top \Sigma^{-1} \underline{\mu}_1 \\ + \frac{1}{2} \underline{x}^\top \Sigma^{-1} \underline{x} - \frac{1}{2} \underline{x}^\top \Sigma^{-1} \underline{\mu}_0 - \frac{1}{2} \underline{\mu}_0^\top \Sigma^{-1} \underline{x} + \frac{1}{2} \underline{\mu}_0^\top \Sigma^{-1} \underline{\mu}_0 \\ + \log\left(\frac{\pi_1}{\pi_0}\right) = 0 \end{aligned}$$

Each term is a scalar here.

Covariance matrix is symmetric  
 $\Sigma = \Sigma^\top$  and  $\Sigma^{-1} = (\Sigma^{-1})^\top$ , so

using  $\Sigma \Sigma^{-1} = I$

$$(\underline{x}^\top \Sigma^{-1} \underline{\mu}_1)^\top = \underline{\mu}_1^\top (\Sigma^{-1})^\top \underline{x} = \underline{\mu}_1^\top \Sigma^{-1} \underline{x}$$

But since all these values are  $1 \times 1$ , we get  $(\underline{x}^\top \Sigma^{-1} \underline{\mu}_1)^\top = \underline{x}^\top \Sigma^{-1} \underline{\mu}_1 = \underline{\mu}_1^\top \Sigma^{-1} \underline{x}$

$$\begin{aligned} \underline{x}^\top \Sigma^{-1} \underline{\mu}_1 - \frac{1}{2} \underline{\mu}_1^\top \Sigma^{-1} \underline{\mu}_1 - \underline{x}^\top \Sigma^{-1} \underline{\mu}_0 + \frac{1}{2} \underline{\mu}_0^\top \Sigma^{-1} \underline{\mu}_0 \\ + \log\left(\frac{\pi_1}{\pi_0}\right) = 0. \end{aligned}$$

(Assumption  $\Sigma_1 = \Sigma_0$  works in linear case as there will be no square terms here. Case is different in quadratic discriminant analysis).

$$\mathbf{x}^T \Sigma^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \frac{1}{2} \left( \boldsymbol{\mu}_0^T \Sigma^{-1} \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 \right) + \log \left( \frac{\pi_1}{\pi_0} \right) = 0$$

- Now this is in the form of decision boundary  $\mathbf{x}^T \mathbf{B} + b = 0$ .
- Considering both classes as equiprobable,  $\pi_1 = \pi_0$ , so  $\log(\pi_1/\pi_0) = 0$ .
- All parameters  $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \Sigma$  are given. Now we can use the decision boundary to test a new example.

$$DB: \mathbf{x}^T \Sigma^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) + \frac{1}{2} \left( \boldsymbol{\mu}_0^T \Sigma^{-1} \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 \right) = 0$$

- For testing, use some  $\mathbf{x}$  and based on this expression,  $> 0$  or  $< 0$  we classify.
- When we consider  $\Sigma_0 = \Sigma_1$ , the decision boundary is linear. (Hyperplane when multiple parameters)
- Usually  $\Sigma_1 \neq \Sigma_0$ . so we take  $\Sigma = \frac{\Sigma_0 + \Sigma_1}{2}$
- If we relax the assumption  $\Sigma_k$  are not identical, then the classifier results in Quadratic Discriminant Analysis (QDA).

$$\begin{aligned} \mathbf{x}^T \Sigma \mathbf{x} &= (x_1, x_2) \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= ax_1^2 + cx_2^2 + 2bx_1x_2. \end{aligned}$$

When  $\Sigma_k$  are assumed to be  $I$ , LDA becomes Naive Based classification.

$$f(x_1, x_2) = \frac{1}{(2\pi)^{2/2}} e^{-\frac{1}{2} [(x - \mu)^T I^{-1} (x - \mu)]}$$

$$(x - \mu)^T = [x_1 - \mu_{x_1}, x_2 - \mu_{x_2}]$$

$$(x - \mu) = \begin{bmatrix} x_1 - \mu_{x_1} \\ x_2 - \mu_{x_2} \end{bmatrix}$$

$$f(x_1, x_2) = \frac{1}{\sqrt{2\pi} \sqrt{2\pi}} e^{-\frac{1}{2} (x_1 - \mu_{x_1})^2} e^{-\frac{1}{2} (x_2 - \mu_{x_2})^2}$$

$$= f(x_1) f(x_2)$$

- Hence independent. Thus for Gaussian RV only, we can say that uncorrelatedness implies independence. (Uncorrelated because  $\Sigma_k = I$ ) The reverse independence  $\rightarrow$  uncorrelated is true for any RV.
- Thus for  $\Sigma_k$  as  $I$ ;
- $f(x_1, \dots, x_n | Y=k) = f(x_1 | Y=k) \dots f(x_n | Y=k)$
- so all  $x_1, \dots, x_n$  are independent
- For Naive Bayes classifier, the features of a class are assumed independent which is not true in practice.
- LDA - for multiclass classification.  $k = 1$  to  $K$  where  $K$  is the number of classes.
- Compute  $f_k(x) \pi_k$  for all classes, given some  $x$ . Classify as that digit which gives max value for  $f_k(x) \pi_k$ . (Note: for finding  $f_k(x) \pi_k$ , we have  $k$  covariance and mean matrix vector)

$$\rightarrow f_k(x) \pi_k = \frac{1}{(2\pi)^{n/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)} \pi_k$$

$\rightarrow$  We can equivalently check  $\max (\log (f_k(x) \pi_k))$

$$\delta_k = \log (f_k(x) \pi_k) = \log (1 \Sigma_k^{-1}) - \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log \pi_k$$

$\mu_k$  - average image - Pixel  $i$ , average of all pixel  $i$  in training example and so on.  
(each pixel is a RV)

$\rightarrow$  So given a test example  $x$ , compute  $\delta_k$  for all  $k = 1$  to  $K$  and choose the class with maximum  $\delta_k$ .

$\rightarrow$  Taking  $\Sigma_k = I$

$$\delta_k = -\frac{1}{2} \underbrace{(x - \mu_k)^T (x - \mu_k)}_{+} + \log (\pi_k)$$

Can ignore this term if we take equiprobable classes

So in a way, the distance between given example and average image is used for classifying. Minimum distance gives maximum  $\delta_k$ .

Here we have erroneously modelled the RV as Gaussian, however it's not actually so when we plot and check. It works moderately when the values are drawn from actual Gaussian distribution.

$\rightarrow \Sigma_k$  - covariance matrix of  $K^m$  days, so its symmetric

→ If  $\Sigma_k$  is not identity, we write it using matrix diagonalisation.

$$\Sigma_k = \begin{matrix} nxn & nxn \\ U & S & U^{-1} \\ nxn & nxn \end{matrix}$$

(U is symmetric)

$$[U_1 \dots U_n] \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}$$

Note :  $S^T = S$

$S^{-1} = S^{-1/2} S^{-1/2}$

Here  $U_1 \dots U_n$  each is a  $n \times 1$  vector

(eigenvectors) corresponding to eigenvalues

$$\lambda_1 \dots \lambda_n$$

Here  $U^{-1} = U^T$  (Unitary matrix)

↓ F.T. of eigenvectors

Matrix can be diagonalised if the eigenvectors are independent because only then inverse is possible

$$\Sigma_k^{-1} = (U S U^T)^{-1} = U S^{-1} U^T = U S^{-1} U$$

$$S_k = -\frac{1}{2} (\underline{x} - \underline{u}_k)^T \Sigma_k^{-1} (\underline{x} - \underline{u}_k) + \log(\pi_k)$$

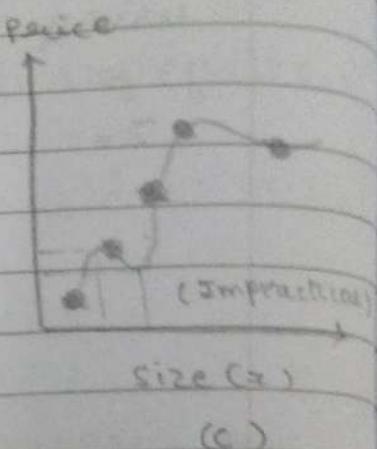
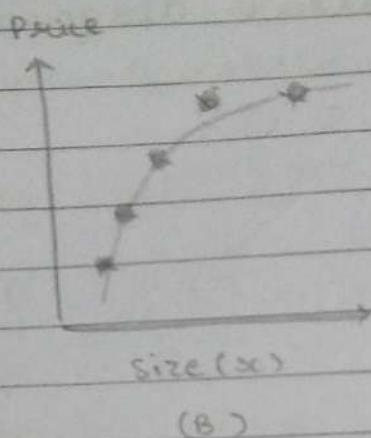
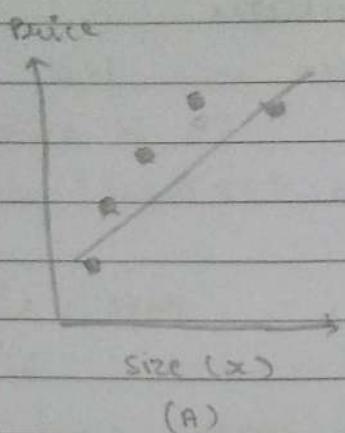
$$= -\frac{1}{2} (\underline{x} - \underline{u}_k)^T U S^{-1} U^T (\underline{x} - \underline{u}_k) + \log(\pi_k)$$

$$= -\frac{1}{2} (U^T \underline{x} - U^T \underline{u}_k)^T S^{-1} (U^T \underline{x} - U^T \underline{u}_k) + \log(\pi_k)$$

$$= -\frac{1}{2} (\underbrace{S^{-1/2} U^T \underline{x}}_{\downarrow} - \underbrace{S^{-1/2} U^T \underline{u}_k}_{\downarrow})^T (\underbrace{S^{-1/2} U^T \underline{x}}_{\downarrow} - \underbrace{S^{-1/2} U^T \underline{u}_k}_{\downarrow}) + \log(\pi_k)$$

Even here we take difference between given given and averages but after taking a linear transformation with  $S^{-1/2} U^T$

## Problem of overfitting and regularization



A -  $h_0(x) = \theta_0 + \theta_1 x$  (high bias).  
[Underfit]

B -  $h_0(x) = \theta_0 + \theta_1 x + \theta_2 x^2$   
[Just right fit]

C -  $h_0(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$   
(high variance) [Overfit]

- Here, given data points are  $y^{(i)}$
- The fitted function is  $\hat{f}(x^{(i)})$  with  $i=1 \text{ to } m$

→ Regularisation takes care of choosing the right  $h_0(x)$ . This is done by choosing the correct  $\lambda$  in

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (y^{(i)} - h_0(x^{(i)}))^2 + \lambda \|\theta\|^2$$

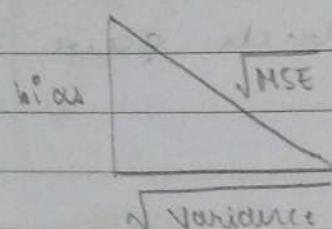
Problem of overfitting — If we have too many features the learned hypothesis may fit very well to training data but fails to generalize to new example.

- This concept applies to regression and classification both.
  - For every problem, we need to estimate the true function which gave the training data points
  - $\theta$  - true value - unknown  
 $\hat{\theta}$  - random value
  - If  $E[\hat{\theta}] = \theta$ , then the estimator is unbiased.
  - If  $-E[\hat{\theta}] + \theta \neq 0$ , then estimator is biased.  
 bias value
  - $\text{var}(\hat{\theta}) = E[(\hat{\theta} - E[\hat{\theta}])^2]$  (variance of estimator)
  - Ideal estimator has bias = 0 and variance = 0  
 (Not practically possible)
  - $(\text{bias})^2 + \text{variance} = \text{mean squared error}$   
 between true and estimated function
  - $(-E[\hat{\theta}] + \theta)^2 + E[\hat{\theta} - E[\hat{\theta}]]^2$   
 $= \theta^2 + (E[\hat{\theta}])^2 - 2\theta E[\hat{\theta}] + E[\hat{\theta}^2] - (E[\hat{\theta}])^2$   
 $= \theta^2 - 2\theta E[\hat{\theta}] + E[\hat{\theta}^2] - A$
  - Also  $E[(\theta - \hat{\theta})^2] = E[\theta^2] + E[\hat{\theta}^2] - 2E[\theta \hat{\theta}]$   
 $= \theta^2 + E[\hat{\theta}^2] - 2\theta E[\hat{\theta}]$
- Hence  $A = B$  — B

### Regularisation

- In ML, the task is to estimate a function. This is done based on available data and

- making assumptions on true function
- We are choosing a function from a class  
(estimated function)



$$\text{bias} \propto \frac{1}{\text{variance}} \quad (\text{MSE constant})$$

- Given set of examples:  $T = \{(x_i, y_i)\}_{i=1}^n$   
observed value (RV depending on RV  $x_i$ )
- $y_i = f(x_i) + \epsilon_i$ . Here  $\epsilon_i$  is assumed to be Gaussian  $N(0, \sigma^2)$

$$\hat{y}_i = \hat{f}(x_i) = h_{\theta}(x) = \theta_0 + \theta_1 x_i \dots$$

(a class of different functions)

- We can reduce the empirical error i.e.  $E[\hat{y}_i - y_i]^2$  (for a finite set of values)
- We cannot actually calculate the true error i.e.  $E[T_{\hat{f}} - f] = E[-\beta_i + \hat{y}_i]$   
(for infinite set of values, i.e. the whole function)

$$\begin{aligned}\text{Bias} &: f - E[\hat{f}] \\ \text{Variance} &: E[\hat{f} - E[\hat{f}]]^2 \\ \text{MSE} &: E[(f - \hat{f})^2]\end{aligned}$$

- High bias means estimated function is not able to cope up with actual function
- Consider MSE between data points and fitted function for a point  $(x_0, y_0)$
- $$E(\hat{y}_0 - y_0)^2 = E[(\hat{f} - f)^2]$$

$$\mathbb{E}[(\hat{y}_0 - y_0)^2] = \mathbb{E}[(\hat{f}_0 - f_0)^2] + \mathbb{E}[e_0^2] \\ - 2\mathbb{E}[e_0(\hat{f}_0 - f_0)]$$

$$\text{Var}(e_0^2) = \mathbb{E}[e_0^2] - (\mathbb{E}[e_0])^2 \\ \sigma^2 = \mathbb{E}[e_0]^2 - 0 \\ \sigma^2 = \mathbb{E}[e_0]^2$$

$$\mathbb{E}[(\hat{y}_0 - y_0)^2] = \mathbb{E}[(\hat{f}_0 - f_0)^2] + \sigma^2 - 2\mathbb{E}[(y_0 - f_0)(\hat{f}_0 - f_0)]$$

(Estimated error)      (True error)

This is kind of covariance between  $y_0$  and  $\hat{f}_0$ . Now, this  $\hat{f}_0$  is obtained from the training data. So, if we choose a testing example  $(x_0, y_0)$  not from the training set, then this covariance will be zero.

$$y_0 = f(x_0) + e_0$$

$$\downarrow \qquad \downarrow \\ \text{true } N(0, \sigma^2)$$

Using this and assuming a  $\hat{f}(x_0)$ , we arrive at  $\hat{f}(x_0) = \hat{y}_0$

→ Considering  $n$  training points, we can write --

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (f_i - \hat{f}_i)^2 + \text{noise}$$

(Empirical)                  (True)

→ Empirical error in a way represents the true error (except the noise) provided we use a point outside the training set.

- for 'validation'. (cross validation test)
- If our fitted function works well for these new points outside training set, then the true error will be minimized.
  - Thus empirical error is a good estimate of true error, when considering a point outside training set. This is the justification of using cross validation test.
  - This is not the case when we consider a point in the training set. i.e.  $(x_0, y_0)$  belongs to training set.
  - Then  $E[\epsilon_0 (\hat{f}_0 - f_0)] \neq 0$
- \* → Consider Stein's Lemma :
- If  $x$  is a R.V.  $N(\theta, \sigma^2)$  and  $g(x)$  is a differentiable function, then
- $$E[g(x)(x-\theta)] = \sigma^2 E\left(\frac{\partial g(x)}{\partial x}\right)$$
- $\epsilon_0 \sim N(0, \sigma^2)$
- $x = N(\theta, \sigma^2)$
- $x - \theta = N(0, \sigma^2) = \epsilon_0$  (mean subtracted)

$E[g(x)(x-\theta)]$  compared with  $E[\epsilon_0 (\hat{f}_0 - f_0)]$   
 we get  $g(x) = \hat{f}_0 - f_0 = \hat{y}_0 - f_0$   
 (so this is actually a function of  $\epsilon_0$ )  
 because  $y_0 = x_0 + \epsilon_0$  and  $\hat{y}_0$  is obtained from  $y_0$ .

$$\begin{aligned} E[\epsilon_0 (\hat{f}_0 - f_0)] &= \sigma^2 E\left(\frac{\partial}{\partial \epsilon_0} (\hat{f}_0 - f_0)\right) \\ &= \sigma^2 E\left(\frac{d \hat{f}_0}{d \epsilon_0}\right) \end{aligned}$$

(constant so derivative zero)

$$\frac{df^{\hat{o}}}{d\epsilon_0} = \frac{df^{\hat{o}}}{dy_0} \frac{dy_0}{d\epsilon_0} = \frac{df^{\hat{o}}}{dy_0} \quad (y_0 = x_0 + \epsilon_0)$$

$$E[\epsilon_0 (f^{\hat{o}} - f_0)] = \sigma^2 E\left(\frac{df^{\hat{o}}}{dy_0}\right) \quad E \neq D_0 \}$$

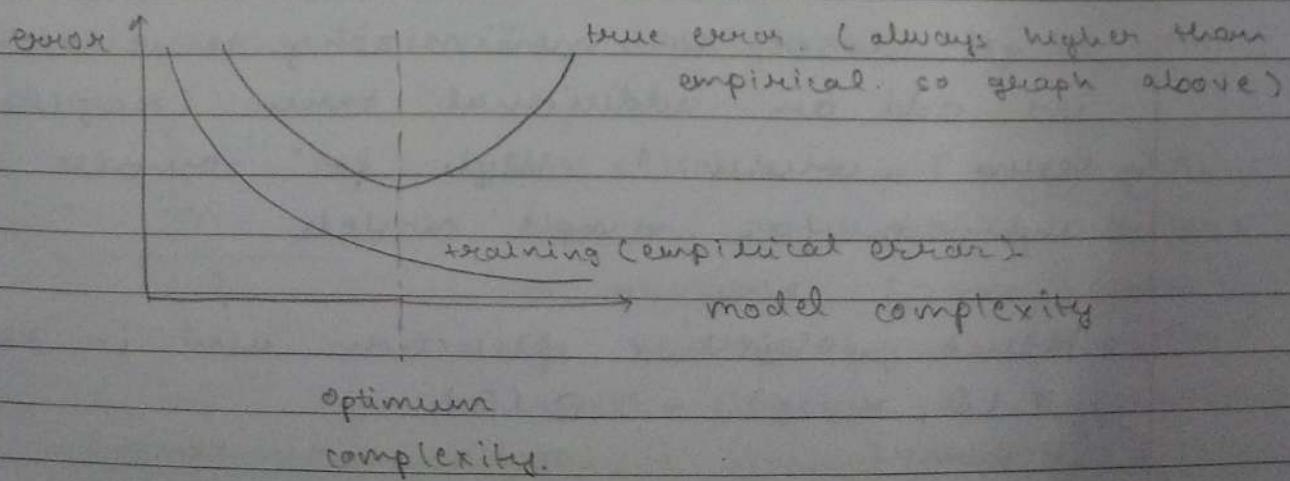
↓

This term represents complexity of model; small when complexity is small.  
Large when complexity is high

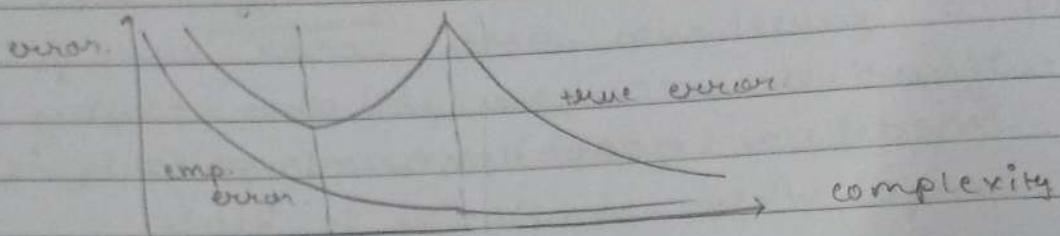
- $\frac{df^{\hat{o}}}{dy_0}$  is high when small perturbation in  $y_0$  leads to large change in  $f^{\hat{o}}$ , hence a complex model.
- For a simple model,  $f^{\hat{o}}$  does not change a lot with small perturbation in  $y_0$ .

$$\begin{aligned} \text{True error} &= \text{Empirical error} - n\sigma^2 \\ &\quad + 2\sigma^2 \sum_{i=1}^n \delta_i \end{aligned}$$

- For a complex model, overfitting occurs and hence true error will increase



- Neural networks are extremely complex models (classical ML theory). Here the graph of error can changes when complexity is increased tremendously.



- When the points are in the training set, empirical error is not a good estimate of true error. It's always less than true error.
- so to get true error, we add a term that takes care of model complexity.
- If model complexity is less, true error is less; if mode complexity is more, true error is high.
- That is the reason for overfitting. This happens because we are minimizing empirical error while we should have been minimizing true error (which we can't).
- To get closer to minimizing true error, we add an additional term (regularisation term) which is high for complex model and low for simple models.
- Hence, objective function used in regularisation
 
$$J(\theta, \alpha, y) + R(\theta)$$

(empirical error term) (Regularisation term).

$$\text{Eq. } \sum (y^{(i)} - h_{\theta}(x^{(i)})) + \lambda \|\theta\|^2$$

↓  
empirical error↓  
regularisationSupport Vector Machine (SVM).

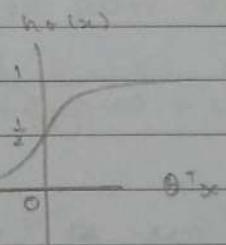
→ supervised learning.

→ binary linear classifier.

→ In logistic regression;

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

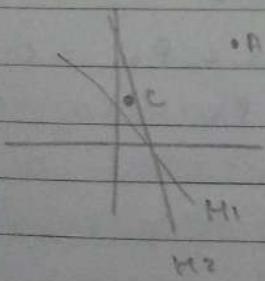
$$\underline{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad \underline{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$



$$\theta^T x > 0 \rightarrow y = 1$$

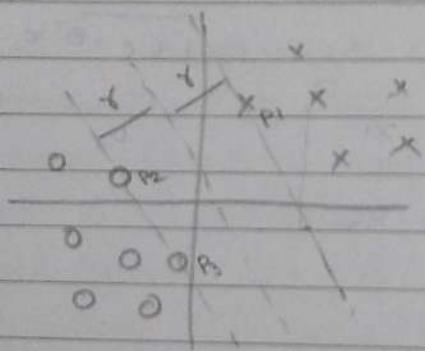
$$\theta^T x < 0 \rightarrow y = 0$$

→ For better accuracy, we train the model such that  $\theta^T x \gg 0$  when  $y = 1$  and  $\theta^T x \ll 0$  when  $y = 0$ . This gives more confidence in classification.



We want the hyperplane decision boundary as far as possible from even the closest point. Else a small change in hyperplane would change the classification of this close point.

- Distance between hyperplane and the closest point is called margin.
- The objective of SVM is to maximize the margin.
- There are infinitely many separating hyperplanes. The best hyperplane as far as SVM is concerned is the one that maximizes the distance between closest data points from both classes. (i.e. the margin).
- Hence, SVM is a maximum margin classifier



$$\text{Margin} = \gamma$$

Support vector - The closest data points ( $p_1, p_2, p_3$ ) which may be N-dimensional

$$\text{hyperplane} = \beta^T x + \beta_0 = \beta_1 x_1 + \beta_2 x_2 + \beta_0 = 0$$

→ Let's label +ve examples as +1 and -ve examples as -1

→ Consider an example of Hyperplane ..

$$x_1 + x_2 = 3$$

$$x_1 + x_2 - 3 = 0$$

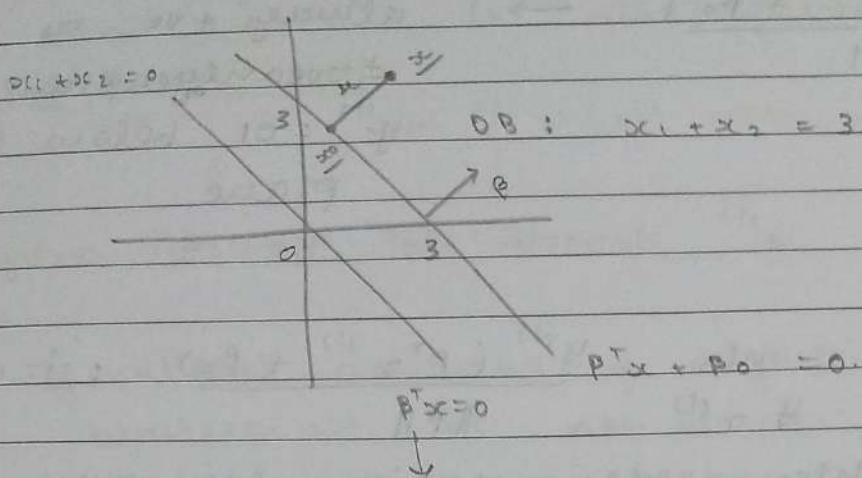
→ Compare with  $\beta^T x + \beta_0 = 0$

and get  $\beta_0 = -3$ ,  $\beta_1 = 1$ ,  $\beta_2 = 1$ .

$$\beta_1 x_1 + \beta_2 x_2 - 3 = 0$$

/ \ / \ / \ /

1x1 2x1 3x1 4x1 5x1 6x1



$p$  is  $\perp$  to  $\infty$  and hence  $p$  is  $\perp$  to the hyperplane

$$\rightarrow (\underline{x} - \underline{x}_0) = (\underline{x}) \frac{\underline{p}}{\|\underline{p}\|}$$

magnitude =  $\infty$   
direction = unit  
vector in direction  
of  $\underline{p}$

$$\rightarrow (\underline{x} - \underline{x}_0) \cdot \underline{p} = \underline{x} \cdot \underline{p}$$

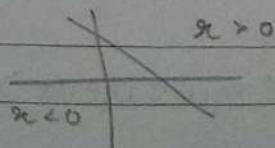
$$\rightarrow r = \frac{\underline{p}^T \underline{x} - \underline{p}^T \underline{x}_0}{\|\underline{p}\|} \quad [\underline{x} \cdot \underline{p} = \underline{p}^T \underline{x}]$$

Now  $\underline{x}_0$  lies on hyperplane so

$$\underline{p}^T \underline{x}_0 + p_0 = 0.$$

$$r = \frac{\underline{p}^T \underline{x} + p_0}{\|\underline{p}\|}$$

$\rightarrow$  Now, we can use this formula of  $r$  to calculate distance of any  $\underline{x}$  from hyperplane.



\*\* Constraint optimisation problem

Page No.:

YUVRAJ  
Date:

$\rightarrow \frac{y^{(i)} [\beta^T x^{(i)} + \beta_0]}{\|\beta\|} \rightarrow$  always +ve as  $y^{(i)} = 1$  above hyperplane and  $y^{(i)} = -1$  below hyperplane  
(Hence  $y^{(i)}$  chosen to be  $\pm 1$  only)

$\rightarrow$  Margin:  $\min_{\beta, \beta_0} \frac{y^{(i)} (\beta^T x^{(i)} + \beta_0)}{\|\beta\|}; i=1, \dots, m$

$\rightarrow$  The problem formulation for SVM is to maximize the margin.

$$\max_{\beta, \beta_0} \left[ \min_{x^{(i)}} \frac{y^{(i)} (\beta^T x^{(i)} + \beta_0)}{\|\beta\|} \right]$$

such that  $y^{(i)} (\beta^T x^{(i)} + \beta_0) > 0$

cannot be  $= 0$  as then we end up lies on decision boundary.

\*  $\rightarrow$  Since  $\beta$  is not dependent on  $x$ , we have

$$\max_{\beta, \beta_0} \frac{1}{\|\beta\|} \left[ \min_{x^{(i)}} \frac{y^{(i)} (\beta^T x^{(i)} + \beta_0)}{\|\beta\|} \right] \quad \text{min possible value} = 1.$$

such that  $y^{(i)} (\beta^T x^{(i)} + \beta_0) > 0$ .

inequality constraint

$\rightarrow$  Taking  $y^{(i)} (\beta^T x^{(i)} + \beta_0) \geq c$  will give.

$$y^{(i)} \left( \frac{\beta^T x^{(i)}}{c} + \frac{\beta_0}{c} \right) \geq 1 \quad \text{+ve constant}$$

$$y^{(i)} (\beta^{*T} x^{(i)} + \beta_0^*) \geq 1$$

Just a scaled version of IP

so hyperplane  $\beta^T x + \beta_0 = 0$

will not change.

$\beta^*$ : new set of parameters

\* By taking that as min = 1, we get.

$$\max_{\beta, \beta_0} \frac{1}{\|\beta\|} \text{ such that } y^{(i)} (\beta^T x^{(i)} + \beta_0) \geq 1$$

$$\rightarrow \min_{\beta, \beta_0} \|\beta\| \quad (\text{equivalent})$$

(Not  $> 0$ )

→  $\min_{P, P_0} \frac{1}{2} \|P\|^2$  is also equivalent  
 $\downarrow$   
so and it cancels when we take derivative.  
Objective function

- If atleast any one of objective function or constraint is non linear, then its called NLP (Non linear programming)
- Non linear problems are usually difficult to solve.
- Lagrange Multiplier Approach is used to solve for  $P, P_0$ .

Ex Optimize  $2x_1 + 3x_2$  such that  $x_1^2 + x_2^2 = 1$

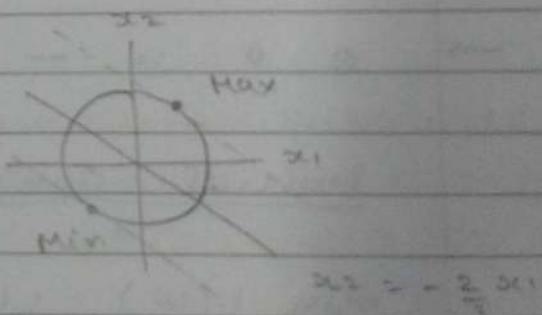
Linear objective  
function

Non linear  
constraint

$$2x_1 + 3x_2 = 0$$

$$3x_2 = -2x_1$$

$$x_2 = -\frac{2}{3}x_1$$



$$x_2 = -\frac{2}{3}x_1$$

What is  $\rightarrow$  Lagrange method....

$$L(x_1, x_2, \lambda) = (2x_1 + 3x_2) + \lambda (x_1^2 + x_2^2 - 1)$$

Lagrange multiplier  $f(x)$   $g(\lambda)$

→ Differentiate with  $x_1, x_2, \lambda$ .

$$2 + 2x_1\lambda = 0 \Rightarrow x_1 = -\lambda$$

$$3 + 2x_2\lambda = 0 \Rightarrow x_2 = -\frac{3}{2}\lambda$$

$$x_1^2 + x_2^2 - 1 = 0$$

$$\frac{1}{\lambda^2} + \frac{9}{4\lambda^2} = 0 \Rightarrow \frac{13}{4\lambda^2} = 1 \Rightarrow \lambda = \pm \frac{\sqrt{13}}{2}$$

[Constraint  $y^{(i)} (\underline{\beta}^T \underline{x}^{(i)} + \beta_0) \geq 1$  is equivalent to  $1 - y^{(i)} (\underline{\beta}^T \underline{x}^{(i)} + \beta_0) \leq 0$ ]

$$g^{(i)}(\underline{\beta}, \beta_0)$$

→  $\lambda$  is called the Lagrange multiplier.

$$\rightarrow \lambda = \frac{\sqrt{13}}{2} \Rightarrow x_1 = -\frac{2}{\sqrt{13}}, x_2 = \frac{-3}{\sqrt{13}}$$

$$\lambda = -\frac{\sqrt{13}}{2} \Rightarrow x_1 = \frac{2}{\sqrt{13}}, x_2 = \frac{3}{\sqrt{13}}$$

Here, the second pair of  $(x_1, x_2)$  will give the max  $f(x_1, x_2)$  [because for first pair  $(x_1, x_2)$ ,  $f(x_1, x_2)$  is -ve]

?  $\lambda$  +ve — pure min

$\lambda$  -ve — pure max

→  $\Phi, \beta, w$  — all notations equivalent.

### Primal Optimisation Problem

→  $\min f(w)$  s.t.  $g_i(w) \leq 0$  and  $h_i(w) = 0$   
 for  $i = 1 \dots k$  for  $g_i$   
 for  $i = 1 \dots l$  for  $h_i$

To solve this problem, define generalised Lagrangian  $L(w, \underline{\alpha}, \underline{\beta}) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$

$\alpha_i \quad i = 1 \dots k$

$\beta_i \quad i = 1 \dots l$

→ Lagrange multipliers

$$\Theta_p(\underline{w}) = \max_{\substack{\underline{\alpha}, \underline{B}, \alpha_i > 0 \\ \downarrow}} L(\underline{w}, \underline{\alpha}, \underline{B})$$

equivalent to 1 used earlier

- Let us say some  $\underline{w}$  is given.
- If  $\underline{w}$  violates any primal constraint, i.e.  $g_i(\underline{w}) > 0$  and/or  $n_i(\underline{w}) \neq 0$ , then  $\Theta_p(\underline{w}) = \infty$ .
- When constraints are satisfied, we get some  $\Theta_p(\underline{w})$ .
- So the original primal problem is written as  $\min_{\underline{w}} (\Theta_p(\underline{w}))$  i.e.

$$\min_{\underline{w}} \left[ \max_{\substack{\underline{\alpha}, \underline{B}, \alpha_i > 0}} L(\underline{w}, \underline{\alpha}, \underline{B}) \right]$$

- This will give the required parameters  $\underline{w}^*, \underline{\alpha}^*, \underline{B}^*$  and on putting those values back we get  $p^*$ .

- Suppose we interchange the max, min order, then we land into the dual problem.  $\max_{\substack{\underline{\alpha}, \underline{B}, \alpha_i > 0}} \left[ \min_{\underline{w}} L(\underline{w}, \underline{\alpha}, \underline{B}) \right]$

Note :  $\underline{w}$  - primal variable  
 $\underline{\alpha}, \underline{B}$  - dual variables.

→ Now, we try understanding primal problem with an example ...

Take some  $\underline{w}_1 = \begin{bmatrix} \dots \\ \dots \\ \dots \end{bmatrix}_{(n+1) \times 1}$

→ So now  $\underline{w}_1$  is fix. Take different pairs of  $\underline{\alpha}, \underline{\beta}$  and obtain values of function. Choose the max out of these and let that be associated with  $\underline{w}_1$ .

→ We do the same with multiple  $\underline{w}_s$ .

$$\underline{w}_1 = 25$$

$$\underline{w}_2 = 26$$

$$\underline{w}_3 = 27$$

$$\underline{w}_4 = 88 \text{ and so on ...}$$

(A)

Now  $p^*$  will be the min of all these (corresponding to all  $\underline{w}_s$ ). So here  $p^* = 25$

→ Now if try using dual approach we have some  $\underline{\alpha}, \underline{\beta}$  as constant and obtain functional value with different  $\underline{w}_s$ . Then choose min of those for each  $\underline{\alpha}, \underline{\beta}$ .

$$\underline{\alpha}_1, \underline{\beta}_1 = 2$$

$$\underline{\alpha}_2, \underline{\beta}_2 = 3$$

$$\underline{\alpha}_3, \underline{\beta}_3 = 4$$

$$\underline{\alpha}_4, \underline{\beta}_4 = 1$$

→ Intuitively these values must be lower than those in (A)

Now  $d^*$  is the max of these i.e.  $d^* = u$

→ Clearly  $d^* \leq p^*$

→ We get the correct solution by primal method, not dual (when  $d^* < p^*$ )

→ However, under some condition,  
we get  $d^* = p^*$

→ There are conditions called Karush-Kuhn-Tucker (KKT) under which  $d^* = p^*$  so that we can get  $w$  either by using primal or dual formulation of problem

→ KKT conditions:

$$\textcircled{1} \quad \frac{\partial}{\partial w_i} L(w^*, \alpha^*, \beta^*) = 0 \quad i = 0, \dots, n$$

$$\textcircled{2} \quad \frac{\partial}{\partial \beta_i} L(w^*, \alpha^*, \beta^*) = 0 \quad i = 1, \dots, l$$

$$\textcircled{3} \quad \alpha_i^* g_i(w^*) = 0 \quad i = 1, \dots, k$$

$$g_i(w^*) \leq 0$$

$$\alpha_i^* \geq 0.$$

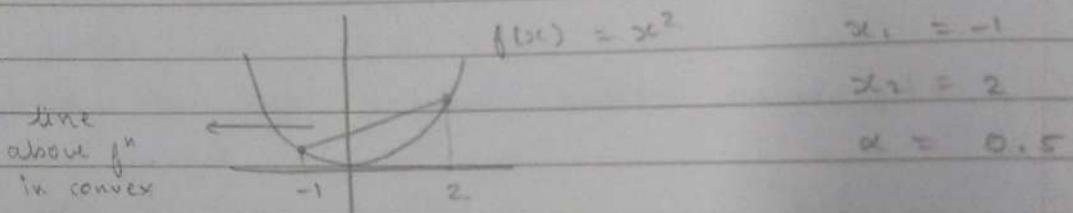
Only for s.v.  $w_i$ 's exist.

(Condition true only when  $\alpha \neq 0$ )

→ The conditions will be satisfied. for  
 $f$  and  $g$  are convex function  
 $m$  are affine.

→ Function  $f$  is convex if  
 $f(x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$ ,  
 where  $\alpha \in [0, 1]$ ,  $x \in \mathbb{R}^n$

Eg.  $f(x) = x^2$  is a convex function



$$\begin{aligned} & \rightarrow f((0.5)(-1) + (1-0.5)(2)) = f(0.5) = 0.25 \\ & \rightarrow (0.5)f(-1) + (1-0.5)f(2) = 2.5 \\ & \text{Hence convex.} \end{aligned}$$

$$\rightarrow f(w) \text{ here is } \frac{1}{2} \|P\|^2$$

$$g(w) \text{ here is } 1 - y^{(i)} (P^T x^{(i)} + P_0)$$

$$L(P, P_0, \lambda) = \frac{1}{2} \|P\|^2 + \sum_{i=1}^m [\lambda_i (1 - y^{(i)} (P^T x^{(i)} + P_0))]$$

$$P_1^2 + P_2^2 + \dots + P_n^2$$

→ We solve using dual approach i.e.  
 $\max_{\lambda} \left[ \min_{P, P_0} L(P, P_0, \lambda) \right]$

↓  
 gives  $P^*$  and  $P_0^*$ . Put that in actual  
 function to get a function of  $\lambda$  and  
 then maximize over  $\lambda$ .

\* These always appear as dot product  
(while using dual approach)

Page No.:	youva
Date:	

→ so consider minimizing  $L(\underline{\beta}, \beta_0, \lambda)$   
w.r.t.  $\beta_0, \underline{\beta}$

→ Differentiate  $L$  w.r.t. each  $\beta_i$ ;  $i=0$  to  $n$   
and equate to 0. i.e.

$$\frac{\partial}{\partial \beta_i} L(\underline{\beta}, \beta_0, \lambda) = 0$$

$$\frac{\partial}{\partial \beta_2} L(\underline{\beta}, \beta_0, \lambda) = 0 \quad \text{and so on ...}$$

$$\text{which is } \nabla_{\underline{\beta}} L(\underline{\beta}, \beta_0, \lambda) = 0. \quad \boxed{}$$

$$\beta_i - \sum_{i=1}^m \lambda_i (y^{(i)} x^{(i)}) = 0.$$

$$\underline{\beta} - \sum_{i=1}^m \lambda_i (y^{(i)} | x^{(i)}) = \underline{0} \quad \leftarrow$$

$\downarrow \quad \downarrow$   
 $n \times 1 \quad n \times 1$

$$\text{Hence } \underline{\beta}^* = \sum_{i=1}^m \lambda_i (y^{(i)} | x^{(i)}) \rightarrow \text{Derivative w.r.t. to } \beta_1 \text{ to } \beta_n$$

$$0 = \sum_{i=1}^m \lambda_i y^{(i)} \rightarrow \text{Derivative w.r.t. } \beta_0$$

→ Using  $\|x\|^2 = x^T x$

$$L = \frac{1}{2} \left[ \sum_{i=1}^m \lambda_i y^{(i)} x^{(i)} \right]^T \left[ \sum_{j=1}^m \lambda_j y^{(j)} x^{(j)} \right]$$

$$+ \sum_{i=1}^m \lambda_i \left[ 1 - y^{(i)} \left( \sum_{j=1}^m \lambda_j y^{(j)} x^{(j)} \right)^T x^{(i)} + \beta_0 \right]$$

$$= \sum_{i=1}^m \lambda_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j y^{(i)} y^{(j)} | x^{(i)} |^2 x^{(i)}$$

(Combine like terms and use  $\sum_{i=1}^m \lambda_i y^{(i)} = 0$ )

- \* Quadratic objective function with linear constraints  
(this is a convex optimisation problem).

for  $\lambda_i \in$

Page No.:	_____
Date:	_____

round

- \* This has to now maximised (because this is now a function of  $\lambda$ ) on  $\lambda_i \geq 0$  for  $i = 1, \dots, m$ . with constraint  $\sum \lambda_i y_i = 0$  for  $i = 1, \dots, m$



Formulate problem to solve for  $\lambda_i$   
using Lagrangian approach —  
Lagrangian multiplier ( $\mu, \lambda$ )

Q Once we get  $\lambda_i$ , find  $B = \sum_{i=1}^m \lambda_i y_i b_i^{(1)}$

Q-7 Convex problem — We will have a unique solution. (do not worry about multiple minima)

- This is a quadratic optimisation problem (with quadratic objective function and linear constraints).
- This is a convex optimisation problem (so we do not have to worry about multiple minimas). we will have a unique solution.
- Once we get this, find  $\underline{\beta} = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$   
 $\beta_1, \dots, \beta_n$
- Now our target is to find value of  $\beta_0$  from  $\underline{\beta}^T x + \beta_0 = 0$ .
- For this we use the KKT conditions
 
$$\begin{aligned}\alpha_i^* g(\underline{\beta}^*, \beta_0^*) &= 0 \quad i=1, \dots, m \\ g(\underline{\beta}^*, \beta_0^*) &= 1 - y^{(i)} (\underline{\beta}^{*T} x^{(i)} + \beta_0^*)\end{aligned}$$

(This is  $< 0$ )
- We have  $\alpha_i^* \geq 0$
- so if  $\alpha_i^* > 0$ , then  $y^{(i)} (\underline{\beta}^{*T} x^{(i)} + \beta_0^*) = 1$
- This shows that  $\alpha_i > 0$  only for support vectors i.e.  $\alpha_i$  exist i.e.  $\alpha_i \neq 0$  for support vector only
- For all the other examples,  $\alpha_i = 0$

So far solving our initial problem,

$$\min_{\underline{\beta}, \beta_0} \frac{1}{2} \|\underline{\beta}\|^2 \text{ s.t. } y^{(i)} (\underline{\beta}^T x^{(i)} + \beta_0) \geq 1$$

only a few data points are used (i.e. only the support vectors are used because they have non zero  $\alpha_i$ .)

$\underline{\beta} = (\dots) + \sum_{i=1}^s \alpha_i (\dots)$  if there are  $s$  support vectors

- The research problem here is identifying these support vectors.
- Once we know the S.V. and also know whether it belongs to class 1 or class -1 we have -

For +ve example :  $\underline{\beta^T x^{(i)}} + \beta_0^* = 1$   
 $\beta_0^* = 1 - \underline{\beta^T x^{(i)}}$

For -ve example :  $\beta_0^* = -1 - \underline{\beta^T x^{(i)}}$ .

Here all  $i$  is correspond to support vector.

$$L = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} \underline{x^{(i)T} x^{(j)}}$$

The dimension of solution of this problem is  $R^m$  ( $\alpha_1 \dots \alpha_m$ ) irrespective of the size of data ( $x_0 \dots x_n$ ), because  $\underline{x^{(i)T} x^{(j)}}$  is always a single value

Advantage of dual approach - Even if the data is not linearly separable, we move from  $x \rightarrow \phi(x)$  but the problem still remains the same.

?

- Here we are imposing a strict constraint that no data point is closer than 1 from the D.B. This is called hard margin classifier.
- With primal approach, if data is not linearly separable by hyperplane, then this method gives no solution.

$\begin{array}{c} + \\ - \\ - \end{array}$

$\begin{array}{c} + \\ + \\ + \end{array}$

linearly separable

$\begin{array}{c} + \\ + \\ - \\ + \end{array}$

Not linearly separable



At least for some points, the constraint  $y^{(i)}(\beta^T x^{(i)} + \rho_0) \geq 1$  is not satisfied.

- In order to solve this problem when data is not linearly separable, we use softmax classifier.

$$z = x c \leftarrow \begin{matrix} \downarrow \\ \text{channel behavior} \end{matrix}$$

↑      ↑  
 data of    received  
 sent signal    data

(needed a priori for determining  $c$  but once  $c$  is obtained  $z$  is calculated from  $x$  and  $c$ )

M1 analog)

$$\rightarrow y = x \theta$$

$$x = \begin{bmatrix} n_{11} \\ n_{12} \\ \vdots \\ n_{1n} \end{bmatrix}$$

$$MSE = (y - x \theta)^T (y - x \theta)$$

$$= (y^T - \theta^T x^T)(y - x \theta)$$

$$= y^T y - y^T x \theta - \theta^T x^T y + \theta^T x^T x \theta$$

$$\frac{\partial MSE}{\partial \theta} = 0$$

$$= -y^T x - y^T x + \theta^T x^T x + \theta^T x^T x \theta = 0$$

$$y \rightarrow z$$

$$\theta \Rightarrow c$$

$$x \rightarrow x$$

$$y^T x = \theta^T x^T x$$

$$x^T y = x^T x \theta$$

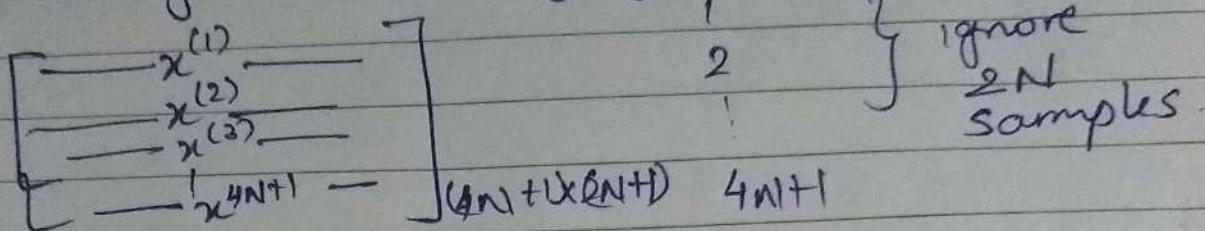
$$\theta = (x^T x)^{-1} x^T y$$

$$\mathbf{C} = (x^T x)^{-1} x^T z$$

$$= R_{xx}^{-1} R_{xy}$$

↑  
 autocorrelation

zero forcing  $\rightarrow$  is like saying, we ignore



$X(k-N)$  to  $X(k+N)$  ← features

$R_{xx}$  and  $R_{xz}$  are correlations between the samples.

But  $X$  is actually data that we ~~send~~ received so we can calculate  $X$  only if the samples are ergodic in nature.

(Not sure what I mean by this but the analogy says so)