શ્રી ઉમિયા યુવા ચેરીટેબલ ટ્રસ્ટ-રાજકોટ

૧૬૨ - બેકબોન શોપીંગ સેન્ટર, માયાણી ચોક,
ચંદ્રેશનગર મેઇન રોડ, રાજકોટ-૪. ફોન : ૨૩૬૭૧૩૬

(રજી.નં.ઈ-૧૧૯૨-રાજકોટ તા.૧૭-૫-૨૦૦૧)

કડવા પાટીદાર કુળદેવી શ્રી ઉમિયા માતાજી સિદસર

SPECIAL

HI-BOND™ cement

Bonding Nation...

WEBSITE :- www. hibondcement.com

EMAIL :- sales@hibondcement.com

INDEX

Name : Kaushal P. Shah Subject : Machine Learning

Std : Div. : — Roll No : —

School / College : —

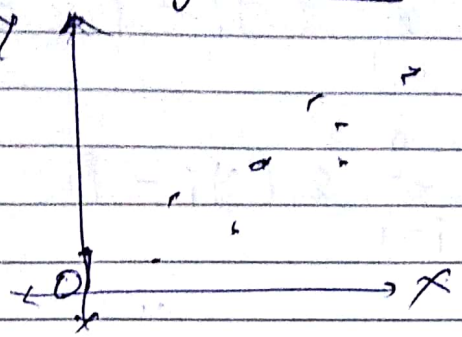
Sr. No.	Date	Title	Page No	Remark

Scanned by CamScanner

MACHINE LEARNING.

Page No.
Date

* Linear Regression: (Line fitting)



$h(x_i) = \beta_0 + \beta_1 x_i$, where $h(x_i)$ is predicted value.

so error (ε_i) = (y_i) − $h(x_i)$
original value.

→ Our aim is to minimize this error, (ε_i)

→ So mean squared error $(J(\beta_0, \beta_1))$ is

$$J(\beta_0, \beta_1) = \frac{1}{2n} \sum_{i=1}^{n} \varepsilon_i^2$$

$$= \frac{1}{2n} \sum_{i=1}^{n} (y_i - h(x_i))^2$$

→ To find Global minima (bcz we are trying to minimize error ε_i), we need to differentiate above eqn.

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_0} = \frac{\partial J(\beta_0, \beta_1)}{\partial \beta_1} = 0$$

Page No.
Date

$$J(\beta_0, \beta_1) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

$$\frac{\partial J(\beta_0, \beta_1)}{\partial (\beta_0)} = \frac{1}{n} \sum_{i=1}^{n} 2(y_i - \beta_0 - \beta_1 x_i)(-1) \rightarrow (1)$$

and

$$\frac{\partial J(\beta_0, \beta_1)}{\partial \beta_1} = \frac{-2}{n} \sum_{i=1}^{n} x_i (y_i - \beta_0 - \beta_1 x_i) \rightarrow (2)$$

→ Compering both to zero and solving both eq^n.

$$\rightarrow -\frac{2}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) = 0$$

$$-\frac{2}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) x_i = 0$$

$$\rightarrow \therefore \sum_{i=1}^{n} y_i - \beta_0 \sum_{i=1}^{n} - \beta_1 \sum_{i=1}^{n} x_i = 0 \rightarrow (1)$$

and $$\sum_{i=1}^{n} y_i x_i - \beta_0 \sum_{i=1}^{n} x_i - \beta_1 \sum_{i=1}^{n} x_i^2 = 0 \rightarrow (2)$$

$$\rightarrow \therefore \sum_{i=1}^{n} y_i - \beta_0 n - \beta_1 \sum_{i=1}^{n} x_i = 0$$

$$\sum_{i=1}^{n} y_i x_i - \beta_0 \sum_{i=1}^{n} x_i - \beta_1 \sum_{i=1}^{n} x_i^2 = 0$$

$$\Sigma y \Sigma x - n\beta_0 \Sigma x - \beta_1 (\Sigma x)^2 = 0$$

$$n\Sigma xy - n\beta_0 \Sigma x - n\beta_1 \Sigma x^2 = 0$$

$$\Sigma x \Sigma y - n\Sigma xy - \beta_1 (\Sigma x)^2 + n\beta_1 \Sigma x^2 = 0$$

$$\Sigma x \Sigma y - n\Sigma xy + \beta_1 \left(n\Sigma x^2 - (\Sigma x)^2\right) = 0$$

$$\beta_1 = \frac{n\Sigma xy - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2}$$

→ writing $\frac{\Sigma a}{n} = \bar{a}$

$$\beta_1 = \frac{\Sigma xy - \frac{\Sigma x \Sigma y}{n^2}(n)}{\Sigma x^2 - \frac{(\Sigma x)^2}{n}}$$

$$\boxed{\beta_1 = \frac{\Sigma xy - n\bar{x}\bar{y}}{\Sigma x^2 - n(\bar{x})^2}}$$

& $\boxed{\beta_0 = \bar{y} - \beta_1 \bar{x}}$

$\left(\frac{\Sigma x^2}{n} \cdot \frac{n^2}{n}\right)$

→ Similar for multiple features (Multiple Variables)

$$J(\beta_0, \beta_1, \beta_2 \dots \beta_m) = \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2$$

→ We can similarly derive $\beta_0, \beta_1 \dots \beta_m$ as done previously.

* But such formula are not used in practice.

* Gradient Descent with one variable

→ Have some cost function $J(\beta_0, \beta_1)$ or $J(\theta_0, \theta_1)$ & we want to minimize it.

→ start with some initial θ_0, θ_1

→ keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we get minimum

* Algo.

repeat until convergence {

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

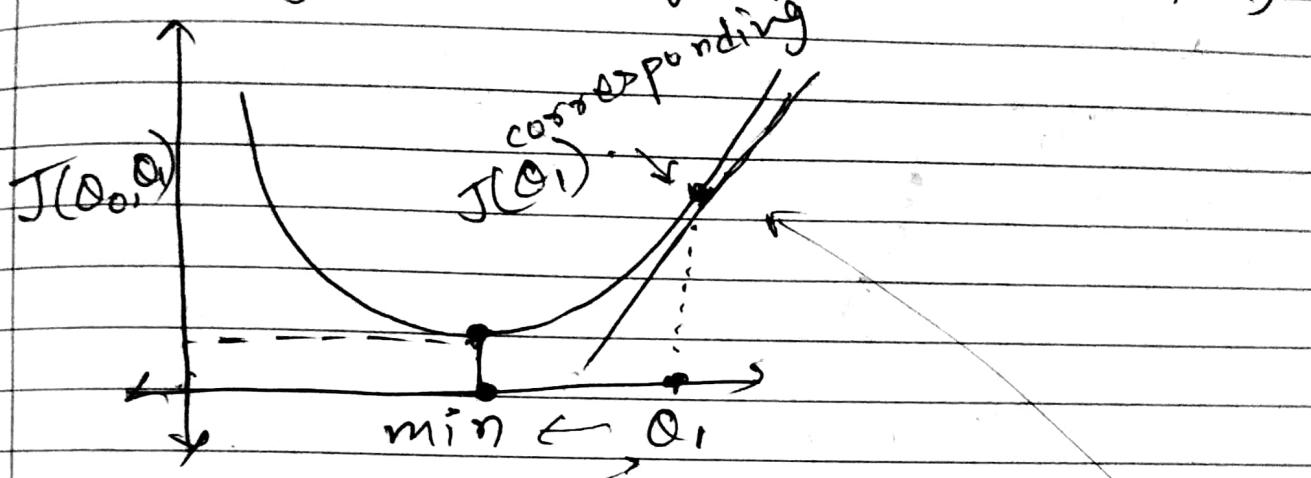
}

where, α is learning rate or step size.

Page No.
Date

* How this works?

→ Assume $\theta_0 = 0$ for simplysity (for getting 2-D graph of $J(\theta_0, \theta_1)$.



→ now, $\theta_1 = \theta_1 - \alpha \underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\text{slop at } \theta_1}$

→ as seen from figure, slop is +ve.

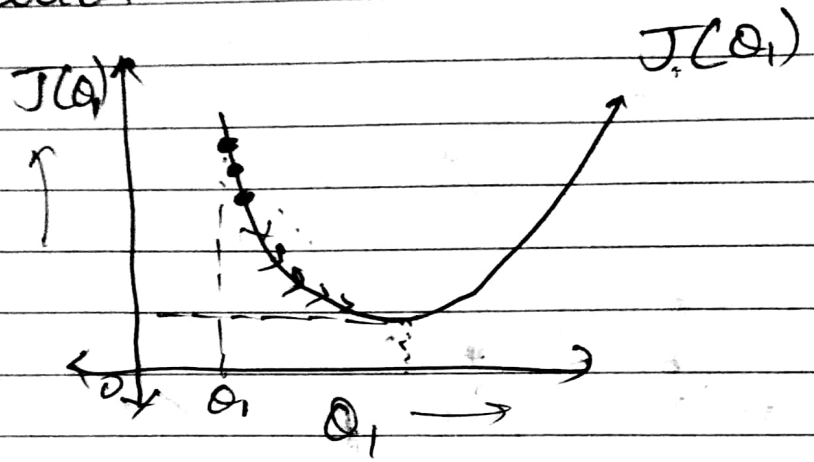
so,

$\theta_1 = \theta_1 - \alpha$ (+ve value)

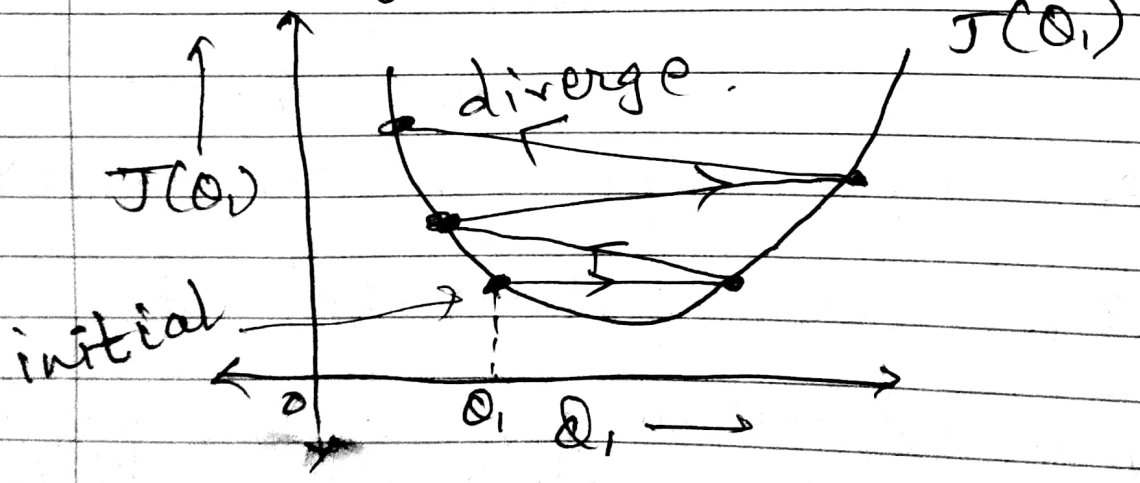
so θ_1 is decreasing, or going towards the minimum value.

→ θ_1 will increase if we had initialized it at LHS of min. value because slop will be -ve there & from above eq^n, θ_1 will increase or go to min. value.

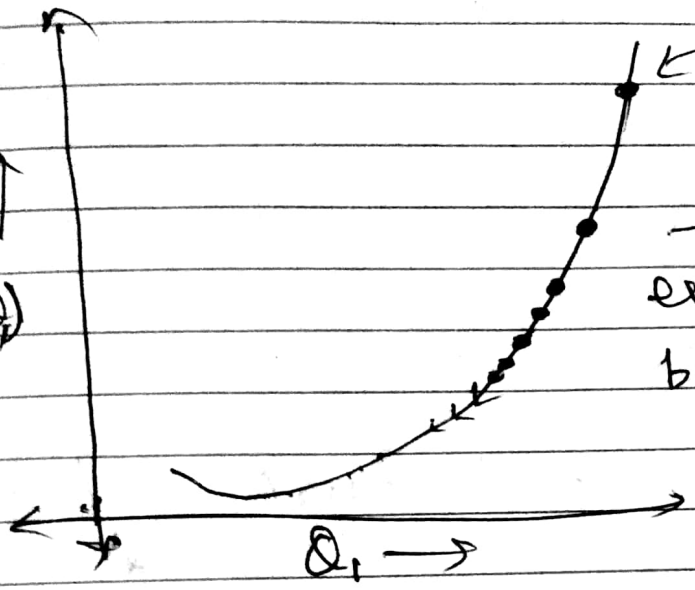
→ If α is too small, steps will be very small., so more iterations to reach minimum value.



→ And if α is too large, it may overshoot the min. It may fail to converge or even diverge.



→ Gradient descent can converge to local minima even with fixed α.



→ step size is decreasing even with fixed α. because of decreasing slop. so it will converge to minimum.

→ "Batch" Gradient Descent :- Each step of gradient descent uses all the training examples.

Lecture-2

Page No.
Date

* Linear Regression.

x_1 = size of house
x_2 = # of bedrooms, then

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

if we define $x_0 = 1$, then.

$$h_\theta(x) = \sum_{i=0}^{2} \theta_i x_i = \theta^T x.$$

if n is # of features, (here $n=2$), then

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x.$$

→ θ's are called # of parameters, which we are going to learn.

→ now error $\varepsilon = (h_\theta(x) - y)$

where $h_\theta(x)$ = predicted output or hypothesis
and y is actual output.

→ Mean squared error can be given as

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

→ our goal is $\min_\theta J(\theta)$ (minimize $J(\theta)$ over param θ)

→ Batch Gradient descent :- Update θ_i after iterating whole training set.

→ stochastic GD :- Update all θ_i for each training examples. So this is more faster then Batch Gradient Descent.

→ now, writing all terms in matrix (vector form),

$$\nabla_\theta J = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} \in R^{n+1}$$

→ so, Batch GD iterations can be given by

$$\theta = \theta - \alpha \nabla_\theta J$$

vector R^{n+1}

$(x^{(1)})^T$ is nothing but a row

Page No. Date

$$X = \begin{bmatrix} -(x^{(1)})^T- \\ -(x^{(2)})^T- \\ \vdots \end{bmatrix}_{n \times m}$$

X is matrix of training ex.

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}_{m \times 1}$$, $\vec{\theta}$ is vector of params to be learned.

so, $X\theta = \begin{bmatrix} (x^1)^T\theta \\ (x^2)^T\theta \\ \vdots \\ (x^m)^T\theta \end{bmatrix} = \begin{bmatrix} h_\theta(x^1) \\ \vdots \\ h_\theta(x^m) \end{bmatrix}$

this is entire row of X.

→ $\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}_{m \times 1}$, where $\vec{y}$ is vector of actual outputs.

$$X\theta - \vec{y} = \begin{bmatrix} h_\theta(x^1) - (y^1) \\ \vdots \\ h_\theta(x^m) - (y^m) \end{bmatrix}_{m \times 1}$$

vector of $m \times 1$

→ If $\vec{z}$ is any vector,

$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$, then

Page No.
Date

$z^T z = \begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$

$$\boxed{z^T z = \sum_{i=1}^{m} z_i^2}$$ for any vector $\vec{z}$

so, $\frac{1}{2}(X\theta - \vec{y})^T (X\theta - y) = \frac{1}{2}\sum_{i=1}^{m} (h(x^i) - y^{(i)})^2$

so, $J(\theta) = \frac{1}{2}(X\theta - y)^T (X\theta - y)$

→ Now to minimize $J(\theta)$, w.r.t θ, we need to take derivative & compare it to zero.

so,

$\nabla_\theta J(\theta) = \vec{0}$

$\therefore \nabla_\theta \frac{1}{2}(X\theta - y)^T (X\theta - y) = 0$

$\therefore \frac{1}{2}\nabla_\theta (X\theta - y)^T (X\theta - y) = 0$

$\therefore \frac{1}{2}\nabla_\theta \left((X\theta)^T (X\theta) - (X\theta)^T y - y^T (X\theta) + y^T y\right) = 0$

$\therefore \nabla_\theta \left(\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta - y^T y\right) = 0$

→ now $J(\theta)$ is a real number, & trace or $tr(a)$ of real no. is just a real number, so

Page No.
Date

$A = tr\ A^T$ & taken transpose bcz it's a real number

$$\therefore \nabla_\theta tr\left(\theta^T x^T x\theta - \theta^T x^T y - y^T x\theta + y^T y\right) = 0$$

$$\therefore \nabla_\theta \left[tr\,\theta\theta^T x^T x - tr(y^T x\theta) - tr(y^T x\theta)\right]$$

now, $\nabla_\theta tr\,\underbrace{\theta}_{A}\underbrace{I}_{B}\underbrace{\theta^T}_{A^T}\underbrace{x^T x}_{C}$

$$= \underbrace{x^T x}_{C}\underbrace{\theta}_{A}\underbrace{I}_{B} + \underbrace{x^T x}_{C^T}\underbrace{\theta}_{A}\underbrace{I}_{B^T}$$

also, $\nabla_\theta tr\,\underbrace{(y^T x}_{B}\underbrace{\theta)}_{A} = \underbrace{x^T y}_{B^T}$

→ so,

$$\nabla_\theta J(\theta) = 0 = \left[x^T x\theta + x^T x\theta - x^T y - x^T y\right]$$

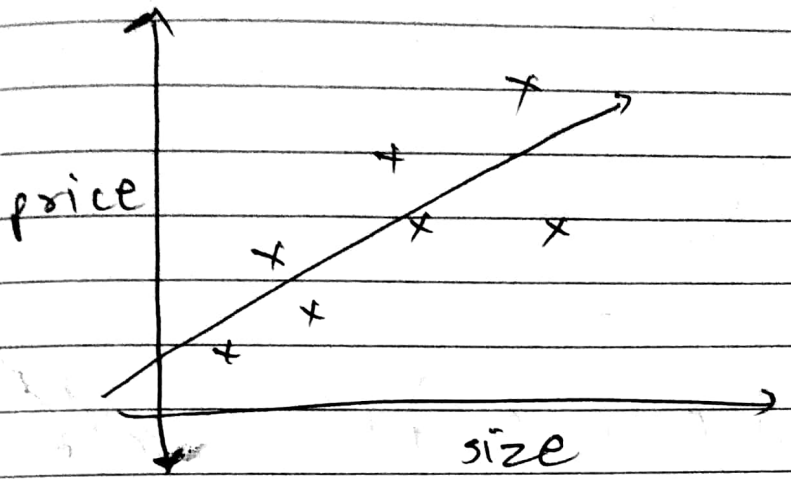
$\therefore x^T x\theta = x^T y \rightarrow$ Normal eq^n & solve it for θ

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

Lecture-3

Page No.	
Date	

→ If there is only one feature, for ex. for predicting prices of houses, if there is only one feature 'size', then

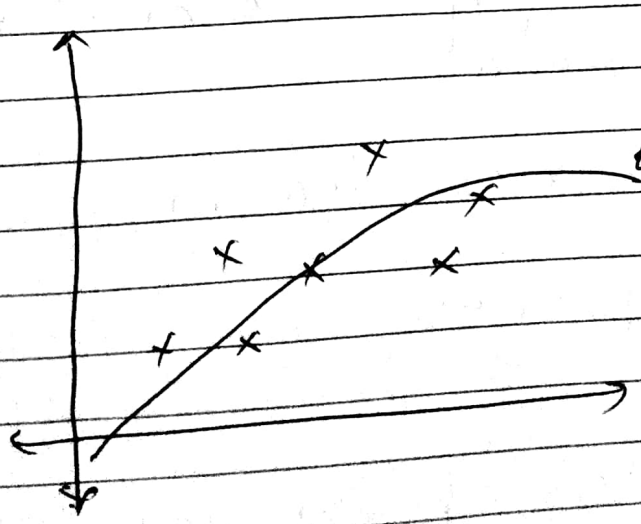


$\theta_0 + \theta_1 x_1$

(underfitting)

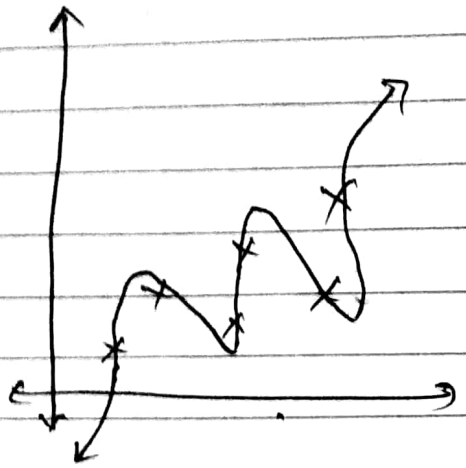
→ but if there are two features, size & $(size)^2$, then our $h_\theta(x)$ would be $\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ ↑ x_2

so,



quadratic, so best fits to model.

→ but if we take too many features, then curve will overfit to data.



$$\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \cdots + \theta_6 x_1^6$$

→ So we need to choose features very carefully.

* Two types of learning algorithms:

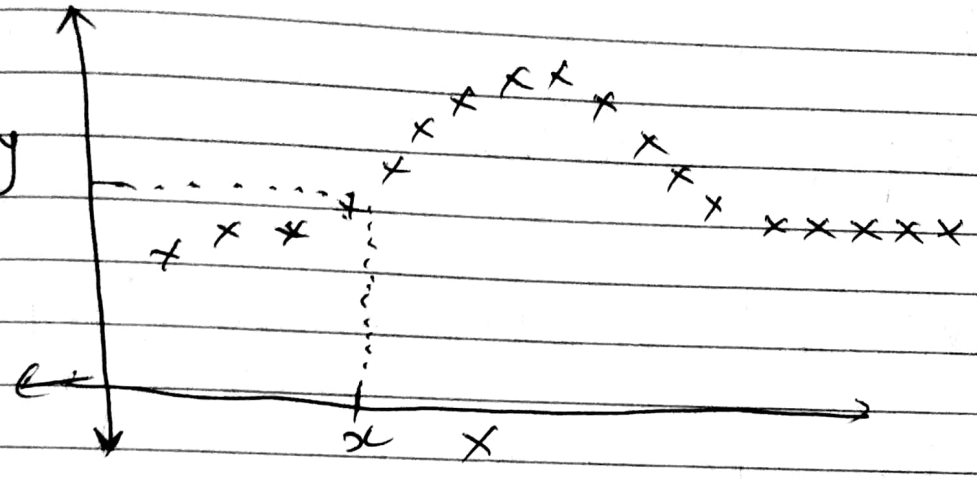
1) "Paramatric" learning algo.
   - It's defined as an algo. which has fixed no. of parameters, to be fitted to data.

2) "Non paramatric" learning algo.
   - # of param grows with m (size of training set).

* Locally weighted Regression is one example of "non-param." Algorithm.

(LWR)

* Locally Weighted Regression / Loess / lowess.



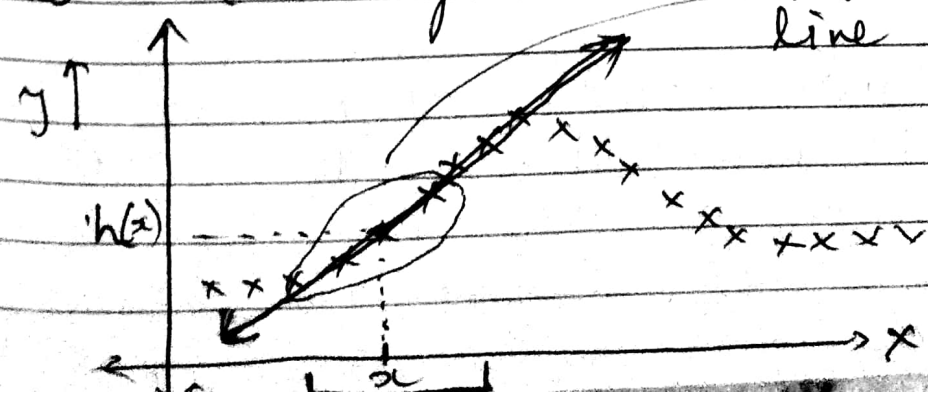
→ In above data linear or quadratic function can't fit perfectly.

→ To evaluate h at a certain x,

In Linear Regression we do following

- fit θ to minimize $\sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$
& then return $\theta^T x$.

→ But in LWR, we apply Linear Regression locally & fit a line locally.



* LWR :- fit θ to minimize

$$\sum_i \omega^{(i)} \left(y^{(i)} - h_\theta(x^{(i)}) \right)^2$$

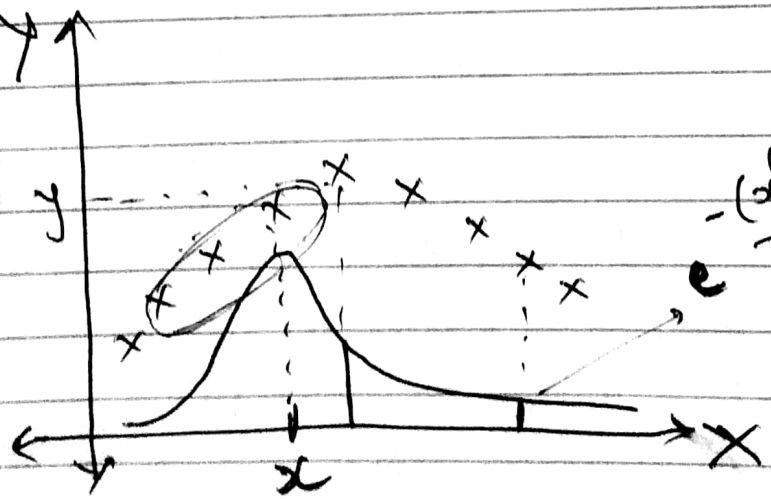
$$\omega^{(i)} = \exp\left(- \frac{(x^{(i)} - x)^2}{2} \right).$$

If $|x^{(i)} - x|$ small, then $e^0 = 1$
$\therefore \omega^{(i)} = 1$

$x^{(i)}$ is close to x.

→ And if $|x^{(i)} - x||$ is large, then $e^{-\text{large}} \approx 0$.

→ so,



$e^{-\frac{(x^{(i)} - x)^2}{2}}$, height of curve at each x gives it's weight.

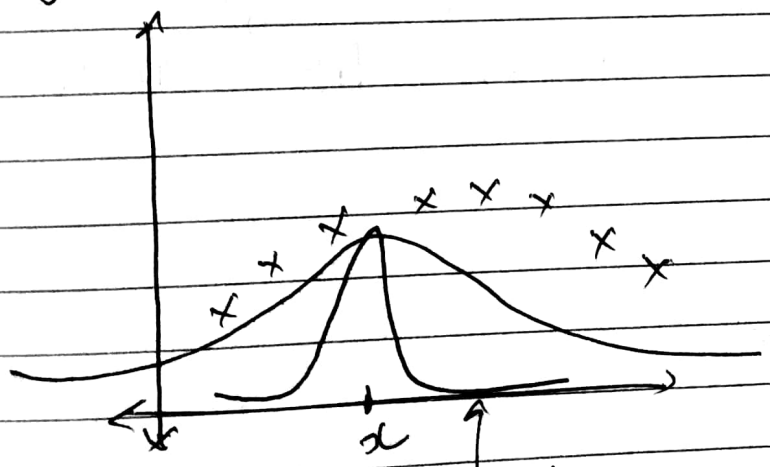
→ so, distant points from x does not contribute much to the above formula because $\omega^{(i)}$ is near to zero.

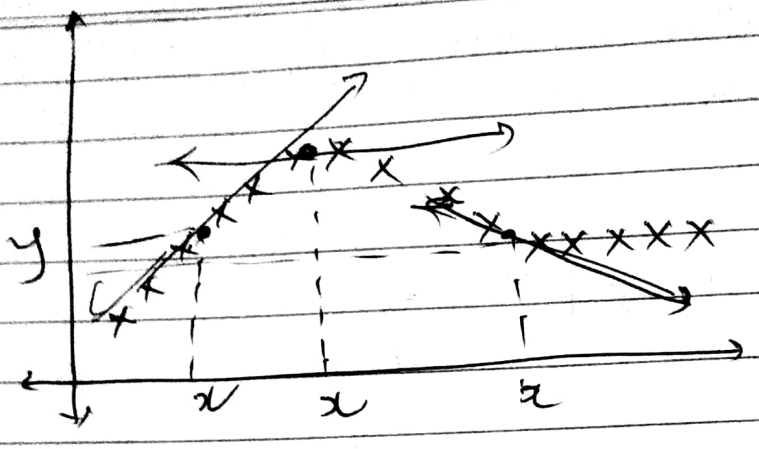
→ More general form of $w^{(i)}$ can be given by

$$w^{(i)} = exp\left(-\frac{(x_i - x)^2}{2\tau^2}\right)$$

where τ is bandwidth., which controls how fast weights of x_i fall with distance from x.

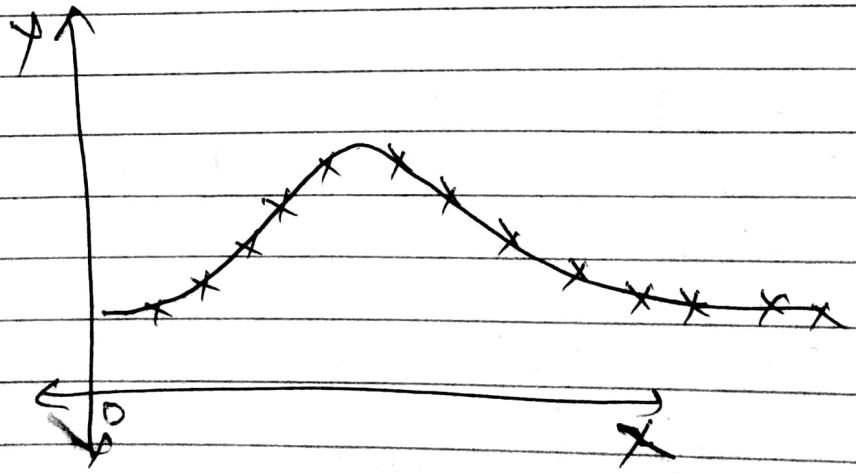
→ So, if τ is small, then it will give narrow bell shape.





→ So, for every new ~~position~~ prediction, we need to run fitting procedure at that point.

→ So, if we do this at every point of x-axis then we get a full non-linear curve like following.



→ Eq^n of $w^{(i)}$ looks like gaussian distribution, but it has nothing to do with it.

Page No.
Date

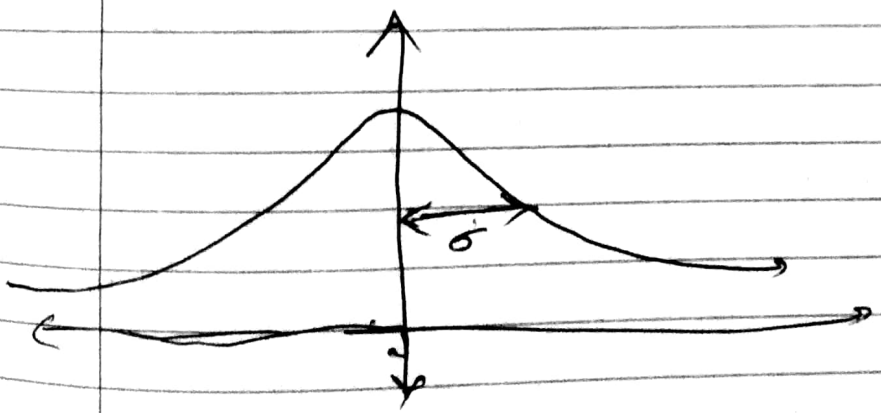
* Probabilistic Interpretation :-

→ Let assume $y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$,

where $\varepsilon^{(i)}$ = error

$\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ => That means $\varepsilon^{(i)}$ are distributed as per gaussian distribution. And $\mathcal{N}$ is for Normal distribution.

$$P(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}\right) \longrightarrow (1)$$



→ So, eq^ (1) means, that the probability distribution of the price of the house or $(y^{(i)})$, for given $x^{(i)}$ (features), and parameters θ, will be gaussian

So, $P(y^{(i)} / x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$

or

$$y^{(i)} | x^{(i)}; \theta \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$$

→ Here θ is not random variable but ϵ is.

→ Now, assume that error terms $\epsilon^{(i)}$s are IID (independently and identically distributed).

→ So,

$$L(\theta) = P(\vec{y} | X; \theta) = \prod_{i=1}^{m} P(y^{(i)} | x^{(i)}; \theta)$$

Likelyhood of θ.

probability of $\vec{y}$ given x, parameterized by θ

$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

(1)

→ We need to choose θ (parameters) to mixmize likelyhood $L(\theta)$.

* Maximum likelihood :-

Choose θ to maximize $L(\theta)$
$= P(\vec{y} | X; \theta)$

Page No.
Date

→ $l(\theta) = \log L(\theta)$ — same as (1)

$$= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp(\cdots)$$

$$= \sum_{i=1}^{m} \log \left[\frac{1}{\sqrt{2\pi}\sigma} \exp(\cdots) \right]$$

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

→ So, maximizing $l(\theta)$ is the same as minimizing

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

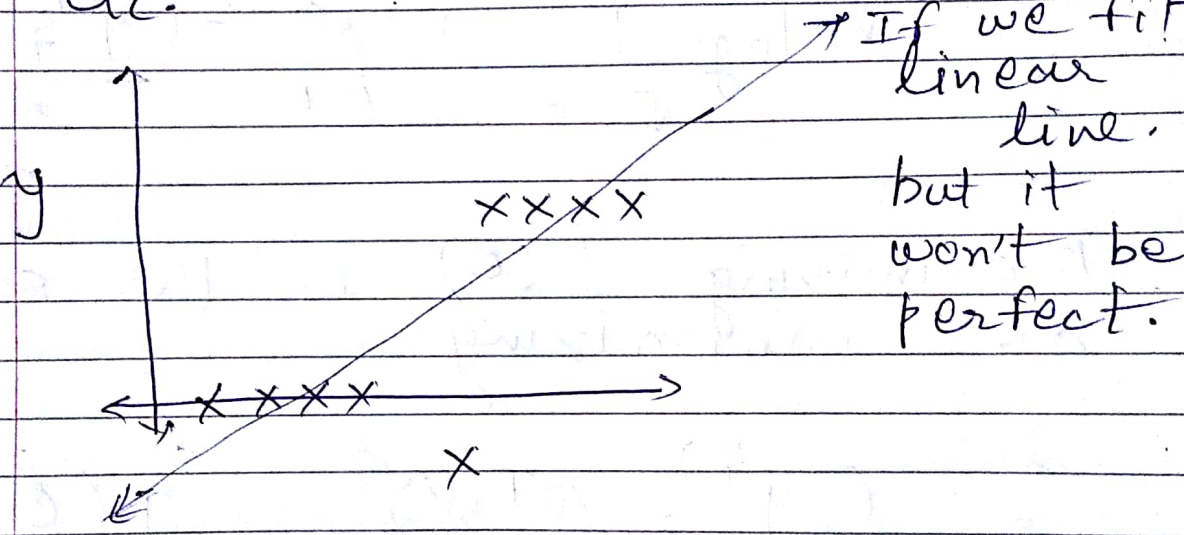
→ value of σ^2 or other constants doesn't matter in minimizing the value of θ.

Page No.
Date 23 4 18

(Bernouli Distribution).

* Classification Algo:- In this, the value of y or outcome is descrete values instead of continuous values as it was in regression problems.

→ for ex. $y \in \{0, 1\}$ or cancer type like (benign or melignant) etc.



→ So, if $y \in \{0, 1\}$, then we can define hypothesys as

$$h_\theta(x) \in [0, 1]$$

→ So, now instead of choosing linear function, we should choose non-linear function.

So,

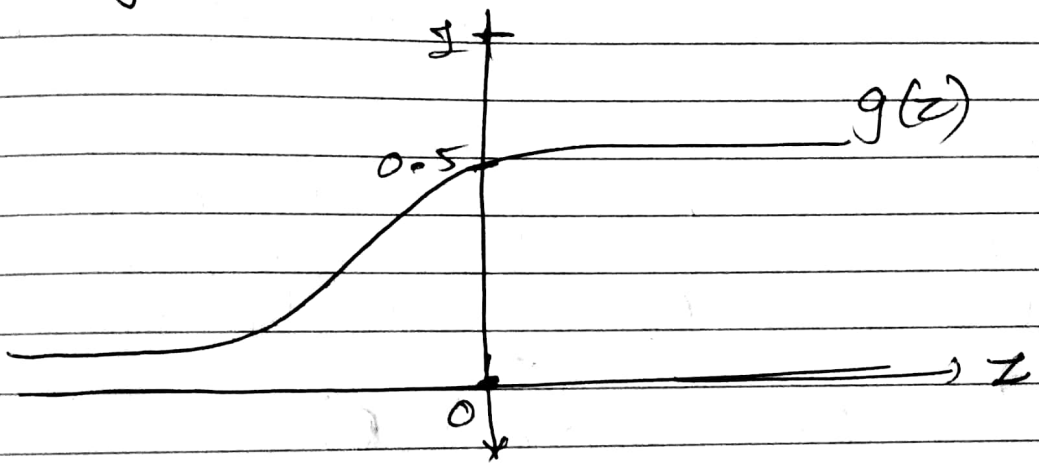
$$h_\theta(x) = g(\theta^T x)$$

Page No.
Date

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

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

→ g is called sigmoid function or logistic function.



→ Now, $P(y=1 \mid x; \theta) = h_\theta(x)$, then

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

→ Combining both of results

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

like binomial distribution

→ So, $L(\theta) = P(y \mid x; \theta) = \prod_i P(y^{(i)} \mid x^{(i)}; \theta)$

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

Page No.
Date

→ Now, to maximize the likelyhood of θ, let's assume

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

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

→ Now, we can apply gradient descent to maximize the value of $l(\theta)$ or likelyhood. so,

$$\theta = \theta + \alpha \nabla_\theta l(\theta)$$

↓

this is '+' here because we are maximizing θ. It was '-' in gradiant descent, bcz we were minimizing error.

→ so, this algo. can be called gradient Ascent.

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

→ So,

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

→ so, the formula is exactly same as batch gradient descent. But the only difference is value of $h_\theta(x)$.

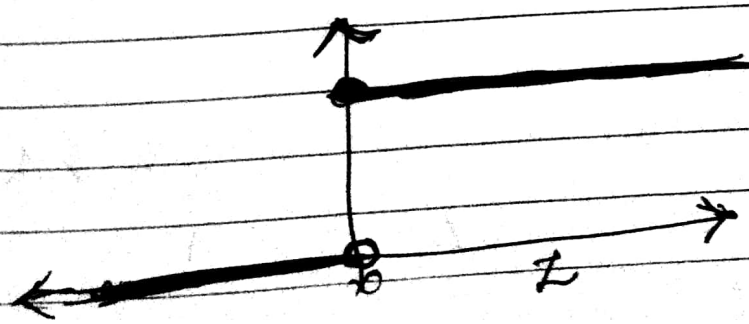
→ here $h_\theta(x)$ is $\frac{1}{1+e^{-\theta^T x}}$ where as in gradient descent,

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

* Perceptron Algo:-

→ In sigmoid ~~func~~ function $g(z)$ can give any value between $[0,1]$. but if we only want to get value as 0 or 1, then it can be defined as

$$g(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



Date

→ so, here $h_\theta(x) = g(\theta^T x)$ (Same as logistic regression)

&

$$\theta_j = \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right) \cdot x_j^{(i)}$$

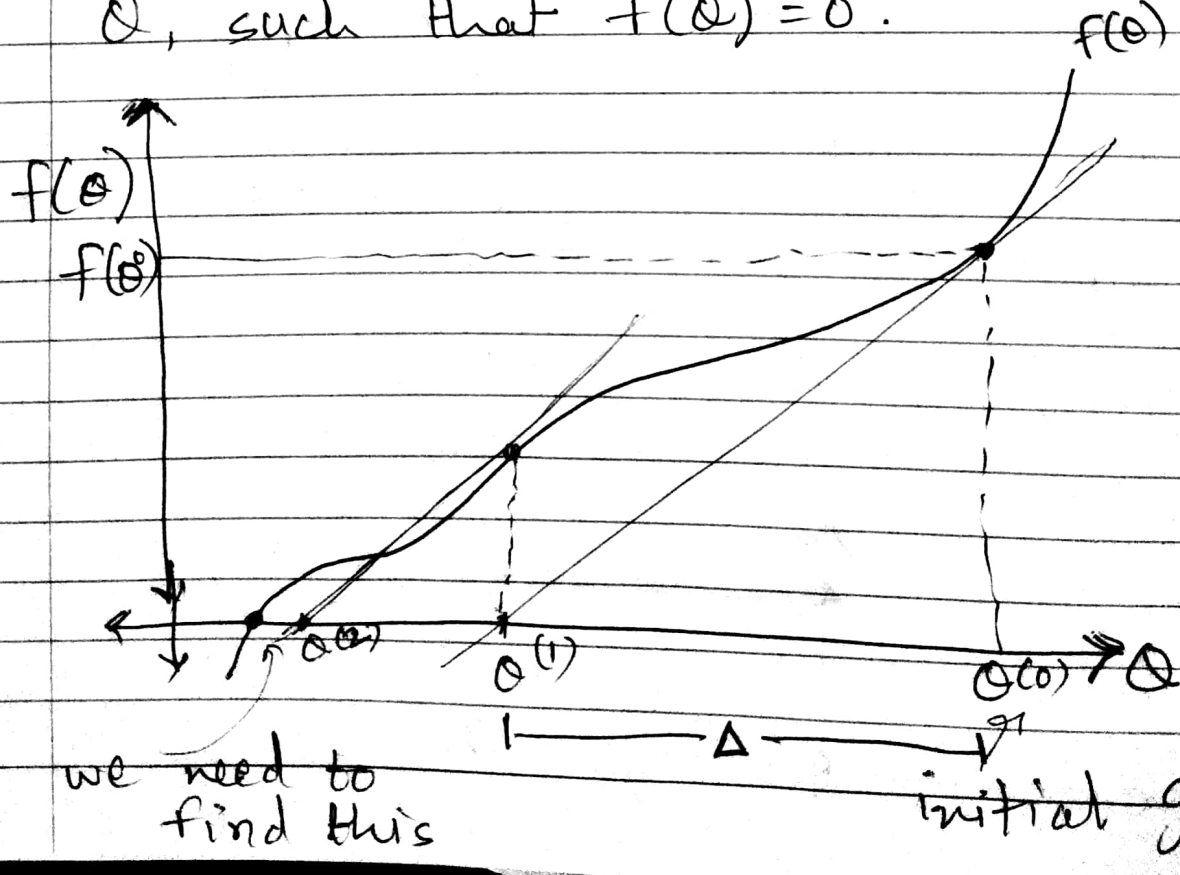
also, same as previous logistic regression.

→ Perceptron algo is more simpler then logistic regression. (previous one).

Lecture-4 25/4/'18

* Newton's Method:-

→ Let's say we've a function $f(\theta)$ and we need to find the value of θ, such that $f(\theta) = 0$.



→ So, in this method, start with random guess of θ, and then find slop at that point, now where the line (tangent at $\theta^{(0)}$) intersects to θ-axis, the value at that point will be our next approx. of θ.

→ $f'(\theta^{(0)})$: slop of tangent at $f(\theta^{(0)})$

$$= \frac{f(\theta^{(0)})}{\Delta}$$

$$\therefore \quad \Delta = \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}$$

→ now, $\theta^{(1)} = \theta_0 - \Delta$ (from figure)

$$\theta^{(1)} = \theta_0 - \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}$$

→ In general,

$$\theta^{(t+1)} = \theta^{(t)} - \frac{f(\theta^{(t)})}{f'(\theta^{(t)})}$$

↳ one iteration of newton's method.

Page No.
Date

→ Now, this algo. can be used to maximize the log likelyh of θ or $[l(\theta)]$.

→ So, we want θ, such that $l'(\theta) = 0$, because $l(\theta)$ is maximum, when $l'(\theta) = 0$, [Global Maxima].

so,

$$\theta^{(t+1)} = \theta^{(t)} - \frac{l'(\theta^{(t)})}{l''(\theta^{(t)})}$$

→ Here θ is a Real number $[\theta \in R]$, but if we've θ as a vector [matrix] then the above generalized form will be

$$\theta^{(t+1)} = \theta^{(t)} - H^{-1} \nabla_\theta l,$$

where H^{-1} is a Hessian Matrix and $\nabla_\theta l$ is gradient, as described previously.

→ In Hessian matrix,

$$H_{ij} = \frac{\partial^2 l}{\partial \theta_i \partial \theta_j}$$

* Convergent Rate is very high

→ Convergent Rate of Newton's method is very high as compared to gradient descent.

→ Convergent rate of Newton's method is Quadratic. so, if initial error is, (error in 1st iteration)

0.01 → 0.0001 (in 2nd iteration) → 0.00000001 (in 3rd iteration).

→ But the disadvantage of Newton's method is that we need to invert the Hessian Matrix in every iteration. Hessian matrix has dimention of $(n+1) \times (n+1)$, where n is no. of features.

Page No.
Date

* Generalized Linear Model :- (GLM)

→ Suppose we've the data that is discrete ({0, 1} valued) and we want to model it with Bernoulli Random variable, parameterised by ϕ,

Bernoulli(ϕ), → it's not a single distribution, but its set of distributions

→ $P(y=1 ; \phi) = \phi$

→ As we vary the value of θ, we'll get different probability distributions.

→ Bernoulli and Gaussian distributions are special cases of exponential family of distribution.

→ $P(y;\eta) = b(y)\exp(\eta^{T} T(y) - a(\eta))$ → (1)

(T → transpose)

where,

η = natural param of distribution

$T(y)$ = sufficient statistic

(Usually $T(y) = y$).

Page No.
Date

→ Now let's show that bernoulli and Gaussian distribution is special case of exponential family of distribution.

→ Let's start with Bernoulli.

→ $Ber(\phi)$, $p(y=1;\phi) = \phi$

→ Now, let's choose a, T and b, such that formula (1) becomes identical to the above one.

→ So, $P(y;\phi) = \phi^y (1-\phi)^{1-y}$

$= \exp(\log \phi^y (1-\phi)^{1-y})$

$= \exp(y \log \phi + (1-y)\log(1-\phi))$

$= \exp(y\log\phi + \log(1-\phi) - y\log(1-\phi))$

$= \exp(y(\log\phi - \log(1-\phi)) + \log(1-\phi))$

$P(y;\phi) = \exp\left(y \log \frac{\phi}{1-\phi} + \log(1-\phi)\right)$

$T(y)$, η, $-a(\eta)$

and $b(y) = 1$.

$\therefore \eta = \log \frac{\phi}{1-\phi}$

$\therefore \phi = \frac{1}{1+e^{-\eta}}$ $\Rightarrow$ Same as logistic function or sigmoid f^n.

$\rightarrow a(\eta) = -\log(1-\phi)$

$a(\eta) = \log(1+e^{\eta})$

$\rightarrow T(y) = y$ and $b(y) = 1$

$\rightarrow$ Now let's do the same for Gaussian.

33:22 $N(\mu, \sigma^2)$ set $\sigma^2 = 1$, because σ^2 value doesn't matters in maximizing of likelyhood of $l(\theta)$

$$N(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(y-\mu)^2\right)$$

$\rightarrow$ taking this in terms of formula (1) we will have

$$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right) \exp\left(\mu y - \frac{1}{2}\mu^2\right)$$

$b(y)$; $\mu = \eta$, $T(y) = y$; $a(\eta) = +\frac{\mu^2}{2}$

Page No.
Date

* Having chosen exponential family distribution, how to derive a generalized linear model? (GLM) ?

→ Assume:- is distributed.

1) $y|x;\theta \sim \text{Exp Family}(\eta)$

2) Given x, our goal is to output expected value of y, given x. ie. $E(T(y)|x)$

In other words, $h(x) = E[T(y)|x]$
(usually $T(y) = y$).

3) $\eta = \theta^T x$ [design choise]
this is when η is real number.
but if η is a vector, then

$[\eta_i = \theta_i^T x$, if $\eta \in R^k]$

→ For Bernoulli distribution,

$y|x;\theta \sim \text{Exp Family}(\eta)$

→ for any fixed value of x and θ, our learning algo, make prediction,

$h_\theta(x) = E[y|x;\theta] = P(y=1|x;\theta)$
$= \phi$

$$= \frac{1}{1+e^{-\eta}} = \frac{1}{1+e^{-\theta^T x}}$$

→ $g(\eta) = E[y;\eta] = \frac{1}{1+e^{-\eta}}$

which is canonical response f^n.
$g^{-1}(\eta)$ is canonical link function.

* Multinomial :-

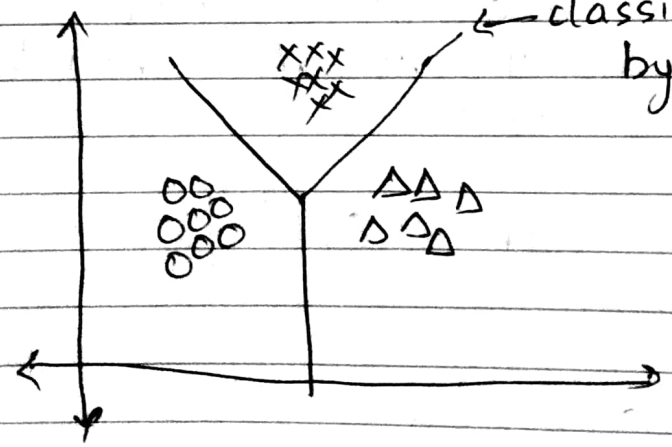
→ It's a distribution over k descrete values

So,

$$y \in \{1, 2, \ldots, k\}$$

→ for example, patient has which of k diseases.

→



→ So, parameters :- $\phi_1, \phi_2, \phi_3 \ldots, \phi_k$

where, $P(y=i) = \phi_i$

$$\phi_1 + \phi_2 + \ldots + \phi_k = 1$$

$$\therefore \phi_k = 1 - (\phi_1 + \phi_2 \ldots + \phi_{k-1})$$

→ for this derivation, let take $\phi_1, \phi_2, \dots \phi_{k-1}$ as parameters and $\phi_k = 1 - (\phi_1 + \phi_2 \dots + \phi_{k-1})$.

→ here $y \in \{1, 2, \dots, k\}$.

$$T(1) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad T(2) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \in R^{k-1}$$

$$T(k-1) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad T(k) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in R^{k-1}$$

→ Note that here $T(y) \neq y$

→ Consider,

$$1\{\text{TRUE}\} = 1, \quad 1\{\text{FALSE}\} = 0$$

So, if $1\{2=3\} = 0$ and $1\{1+1=2\} = 1$

→ $T(y)_i = 1\{y=i\}$ ⇒ this means, that the i^{th} element of vector $T(y)$ will only be equal to 1, iff $y = i$.

→ $P(y) = \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \cdots \phi_k^{1\{y=k\}}$

→ Remember this is not the parameter.

$= \phi_1^{T(y)_1} \cdot \phi_2^{T(y)_2} \cdots \phi_{k-1}^{T(y)_{k-1}} \cdot \phi_k^{1 - \sum_{j=1}^{k-1} T(y)_j}$

$= \cdots$ [after few steps]

→ $= b(y) \exp(\eta^T T(y) - a(\eta))$

where,

$$\eta = \begin{bmatrix} \log(\phi_1/\phi_k) \\ \vdots \\ \log(\phi_{k-1}/\phi_k) \end{bmatrix} \in \mathbb{R}^{k-1}$$

making ϕ subject,

→ $a(\eta) = -\log(\phi_k)$, and $b(y) = 1$.

$$\phi_i = \frac{e^{\eta_i}}{1 + \sum_{j=1}^{k-1} e^{\eta_j}}, \quad (i = 1, \ldots k-1)$$

→ using the assumption,

$$\phi_i = \frac{e^{\theta_i^T x}}{1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}}, \quad [\eta_i = \theta_i^T x]$$

→ $h_\theta(x) = E[T(y) \mid x; \theta]$

$$= E\left[\begin{bmatrix} 1\{y=1\} \\ \vdots \\ 1\{y=k-1\} \end{bmatrix} \middle| x; \theta\right]$$

$$= \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{k-1} \end{bmatrix}$$

→ This algo. is called Softmax Regression. (Generalization of logistic Regression).

→ In a particular machine learning problem, if,

$y \in \{1, \ldots k\}$ [ex. Prediction of digits from images].

our training set will be

$(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$, where $y^{(i)}$ can take any value from $\{1, \ldots, k\}$, then, we'll try to maximize the likelyhood of θ, as following.,

Date

$$L(\theta) = \prod_{i=1}^{m} P(y^{(i)} | x^{(i)}; \theta)$$

$$= \prod_{i=1}^{m} \phi_1^{1\{y^{(i)}=1\}} \cdot \phi_2^{1\{y^{(i)}=2\}} \cdots \phi_k^{1\{y^{(i)}=k\}}$$

where $\phi_i = \frac{e^{\theta_i^T x}}{1 + \sum_{j=1}^{k+1} e^{\theta_j^T \cdot x}}$

→ here $\theta_1, \theta_2 \ldots, \theta_{k+1} \in R^{n+1}$, where n is no. of features.

Lecture-5

✱ Generative learning Algorithms :-

→ In this we train different model for each class and then compare the data with that models to pedict the result.

→ For example, to predict cancer, whether it is malignent or benign, we train separate model for both & then at the time of prediction, we compare our data to both the model.

Page No.
Date

* Logistic regression is discriminative learning algo. so,

→ it learns $p(y|x)$ or learns $h_\theta(x) \in \{0, 1\}$ directly.

→ Generative algo. builds the model on what the features looks like. conditioned on class label. so,

$P(x/y)$ and $P(y)$

$$P(y=1|x) = \frac{P(x/y=1)\,P(y)}{P(x)}$$

$$\therefore P(x) = P(y=0|x)P(x) + P(y=1|x)P(x)$$

Ex Assume that the input features $x \in R^n$ and continuous valued.

→ Gaussian Discriminant Analysis
Here assume that
$P(x/y)$ is Gaussian.

→ Multi variable Gaussian can be given by, (mean matrix, covariance matrix)

$$x \sim \mathcal{N}(\vec{\mu}, \Sigma),$$

$$P(x) = \frac{1}{\sqrt{\Sigma}\,(2\pi)^{n/2}} \exp\left(-\tfrac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

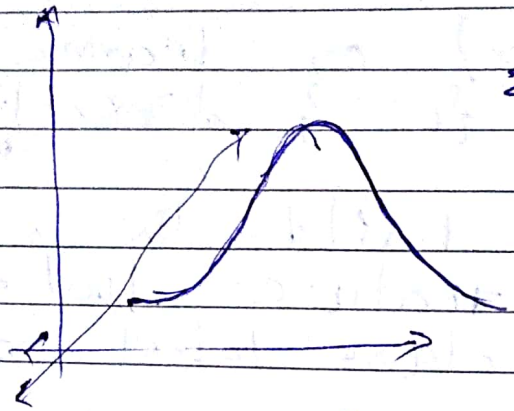
where $\Sigma = E[(x-\mu)(x-\mu)^T]$

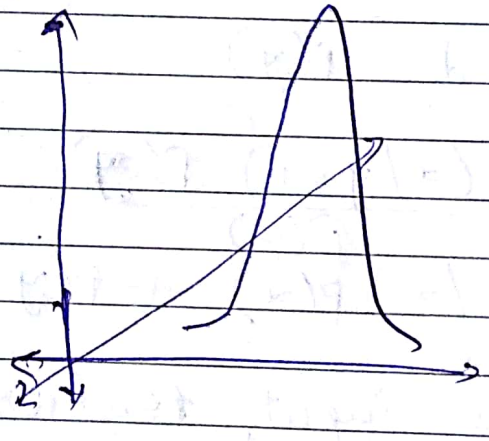
Page No.
Date 12/5/18

→ Examples of Gaussian with mean 0 and different co-variance matrix.

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ↑ responsible for location of curve.

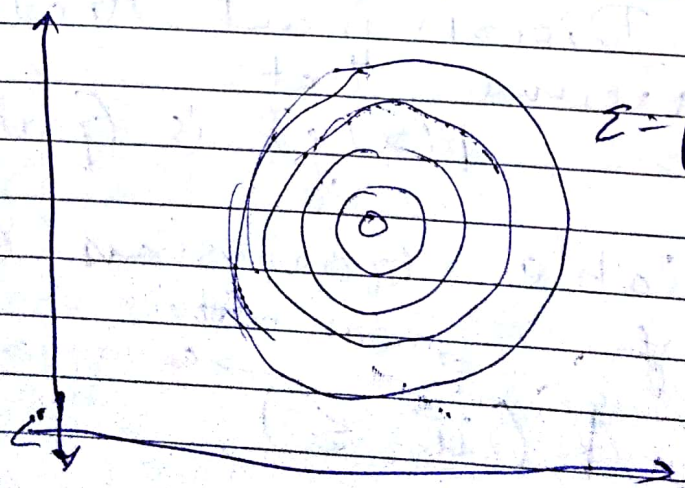
$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ↑ responsible for shape of curve.



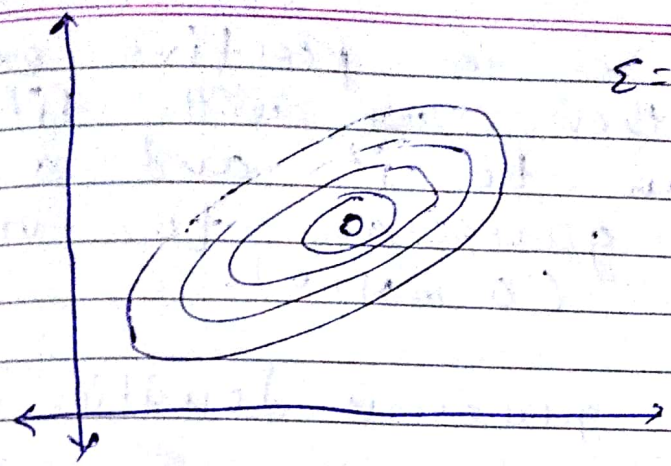


$\begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$

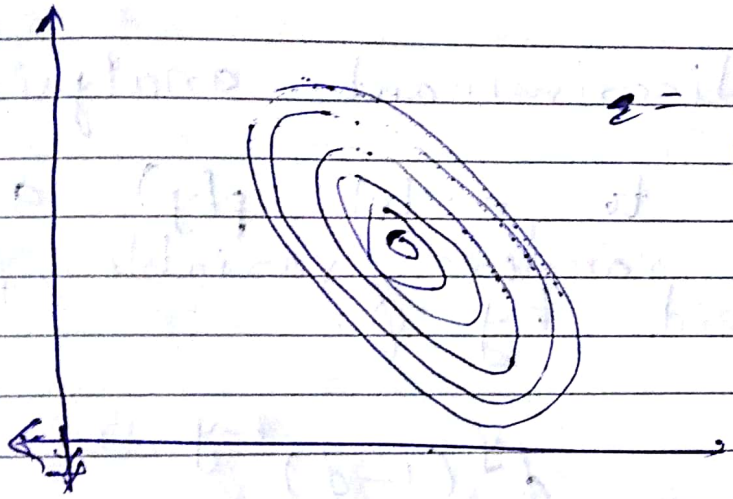
→ In 2-D it can be shown as (Top view)



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

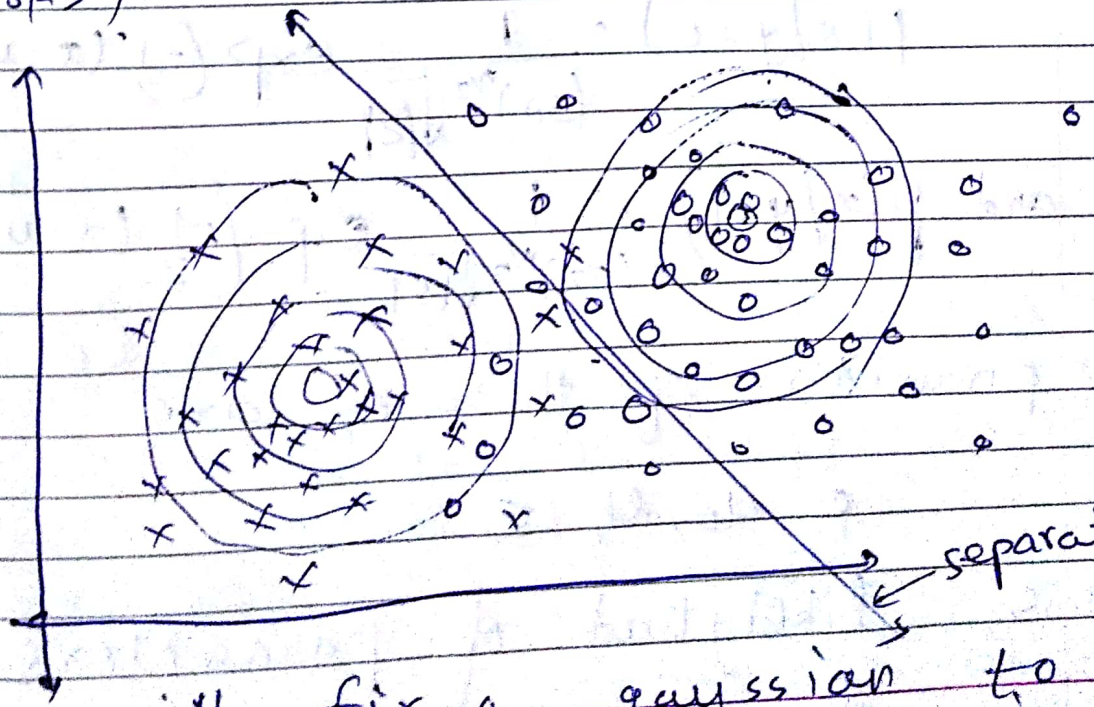


$$\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

* How Gaussian discriminant analysis works?



-) we will fix a gaussian to x and different gaussiant to o.

→ Let's say 'x' is positive examples (y=1), then we will fit a gaussian to it and a different gaussian to -ve examples ('0' or y=0).

→ And this two gaussian densities defines a separator.

* Gaussian discriminant analysis model:

→ It's going to model p(y) as a bernoulli random variable, ~~parameter~~ parameterized by ϕ.

so,

$$p(y) = \phi^{y}(1-\phi)^{1-y}$$

$$p(x/y=0) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} \exp\left(\frac{-1}{2}(x-\mu_0)^{T}\Sigma^{-1}(x-\mu_0)\right)$$

and $$p(x/y=1) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} \exp\left(\frac{-1}{2}(x-\mu_1)^{T}\Sigma^{-1}(x-\mu_1)\right)$$

→ parameters of these eqⁿ are

$$\phi, \mu_0, \mu_1, \Sigma$$

⇒ So, likelyhood of parameters

also called
Joint likelyhood.

$$\ell(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{m} P(x^{(i)}, y^{(i)})$$

$$= \log \prod_{i=1}^{m} P(x^{(i)}/y^{(i)}) \cdot P(y^{(i)})$$

→ Now, as previous, to maximize ℓ, w.r.t ϕ, μ_0, μ, and Σ,

where,

max. likely hood $\phi = \dfrac{\sum_i y^{(i)}}{m} = \dfrac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}{m}$

$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}$$

indicates only the terms where $y^{(i)} = 1$.

no. of examples with label 1.

⇒ so, numarator says sum of $x^{(i)}$

corresponding to the examples where $y^{(i)} = 1$.

→ It simply indicates that find all the examples for $y=1$ (+ve examples) and take the ~~avg~~ average.

→ predict:

$$\arg\max_y P(y|x) = \arg\max_y \frac{P(x|y)P(y)}{P(x)}$$

$$= \arg\max_y P(x|y)P(y)$$

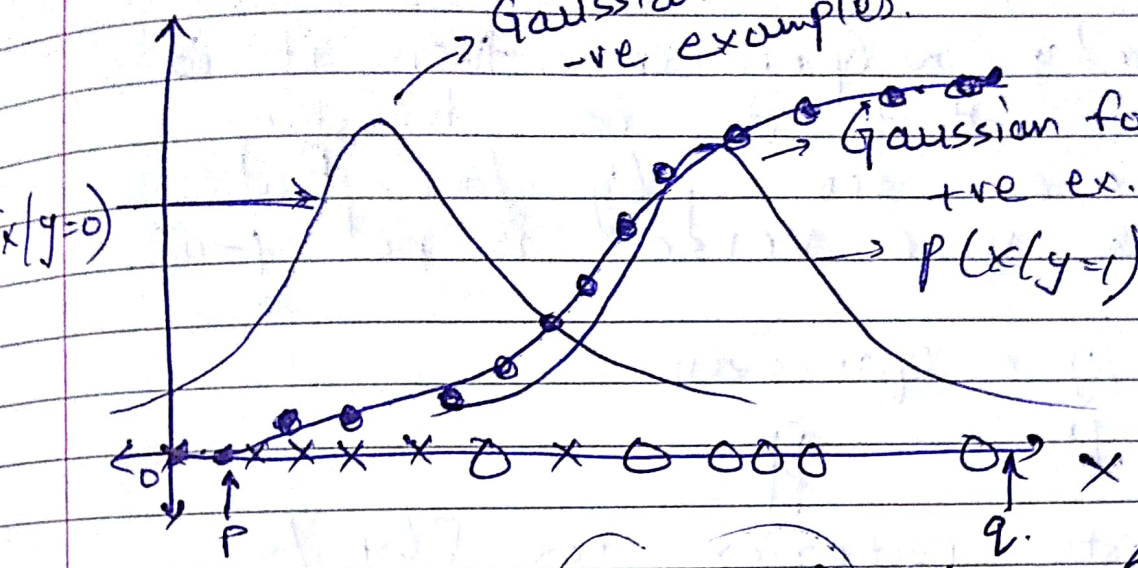
$P(x)$ is ignored because it does not depend on y.

→ If $P(y)$ is uniform, $(P(y=0) = P(y=1))$ then,

$$= \arg\max_y P(x|y)$$

* Note:- $\arg\max_y P(x|y)$ means that value of y, @ for which $P(x|y)$ is maximum.

→ Gaussian discriminant analysis has interesting relation with logistic regression.



→ $P(y=1|x) = \frac{P(x|y=1)\,P(y=1)}{P(x)}$ ← this is simply a bays rule.

get from graph

→ If we calculate $P(y=1|x)$ at point p, then it will be very low. (almost 0)

→ As we keep moving right side on X-axis, the probability increases. If we calculate $P(y=1|x)$ at point q, it will be almost 1. So, If we plot the curve for all such points, it will give the curve of logistic regression.

→ and $p(x) = p(x|y=0)P(y=0) + P(x|y=1)P(y=1)$

Date

Advantages and Disadvantages of Generative learning algo.

→ if $x/y \sim$ Gaussian, then it implies that it is logistic posterior for $p(y=1/x)$. But vice versa is not true

$x/y \sim$ Gaussian

⇏ ⇍

Logistic posterior for $P(y=1/x)$

→ Also, if we assume,

$\begin{cases} x/y=1 \sim \text{Poisson}(\lambda_1) \\ x/y=0 \sim \text{Poisson}(\lambda_0) \end{cases}$

→ ⇒ then $P(y=1|x)$ is logistic.

→ So, x/y is Gaussian is stronger assumption, as, it implies $y|x$ is logistic but vice versa is not true.

Page No.
Date

→ Generative learning Algo. used when data set is small. And also gives good results when such assumenptions do not hold.

→ More generally,

$x|y=1 \sim$ Exp Family (η_1)
$x|y=0 \sim$ Exp Family (η_0)

↓ implies

$P(y=1|x)$ is logistic.

→ Other Generative learning algo. is Naive Bayes Algo.

→ for ex. we want to classify spam emails, then we need to represent an email. mathematically. so, if we create a matrix (vector) of dictionary, then email can be represented by as following.

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \begin{matrix} a \\ aardvark \\ \vdots \\ \\ buy \\ \vdots \\ zulu \end{matrix}$$

problem is in the form of $y \in \{0, 1\}$
↑ not spam ↑ spam

Page No.
Date

→ now, we want to find $P(x/y)$, where $x \in \{0,1\}^n$

$n = 50,000$ or more dimensions.

→ so, if we use softmax or multinomial regression here, then no. of possible matrix are $2^{50,000}$, so, $2^{50,000} - 1$ params.

→ Assume x_i's are conditionally independent given y.

$$P(x_1, x_2, x_3 \cdots x_{50,000} / y) = P(x_1/y) \cdot P(x_2/y, x_1) \cdot P(x_3/y, x_1, x_2) \cdots$$

→ also assume in above eq^n, that

$$P(x_2/y, x_1) = P(x_2/y)$$
$$P(x_3/y, x_1, x_2) = P(x_3/y) \cdots$$

so, $$P(x_1, x_2, \cdots / y) = \prod_{i=1}^{n} P(x_i/y)$$

* What $P(x_2/y, x_1)$ means?

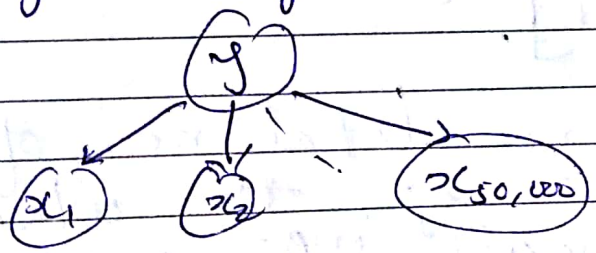
→ It means, given the value of y (0/1) [spam or not spam], and given the word x_1 appears in ~~the~~ email, does not affect the probability

of word x_2 appears in email.

→ But practically this assumption is false, for example if there is word 'CS229' in email, then word 'Andrew Ng' will be there. But dispite of this false assumption, the algo. works correctly in practice.

→ Particularly, this algo. works very well for classifying text archives.

→ Bayesian Network for this model can be given by,



→ Following are the parameters of this models.

$$\phi_{i|y=1} = P(x_i = 1 \mid y = 1)$$

$$\phi_{i|y=0} = P(x_i = 1 \mid y = 0)$$

$$\phi_y = P(y = 1)$$

$\phi_{j|y=1}$ means that the fraction of spam emails ($y=1$) in which word j does appear.

→ Joint likelihood:

Date

$$J(\phi_y, \phi_{i/y=1}, \phi_{i/y=0}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)})$$

→ We can get value of this params based on max. liklihood analysis.

$$\phi_{j/y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}.$$

→ This means:-

denominator :- Total no. of spam emails in training set. [emails where $y=1$].

numerator :- total no. of words x_j, for which the email was spam.

→ $$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}{m}$$

→ To predict for a new email, compute $p(y|x)$ using bayes rule.

→ $$p(y|x) = \frac{p(x|y)\,p(y)}{p(x)}$$

* Laplace smoothing:

→ For Example, if any a in email classification problem using naive bayes algo, if we see a new word 'nips' which was never seen before in any of the training set.

→ Assume that this word is 35000^{th} word in the dictionary.

→ So, our spam classifier algo. will predict following probabilities.

$$\phi_{35000|y=1} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} = 0$$

and

$$\phi_{35000|y=0} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} = 0$$

→ This is bcz it has never seen 'nips' in either spam or non spam. so it thinks seeing this in either type of email has probability 0.

→ And from this, when it computes (using bayes' rule)

$$P(y=1|x) = \frac{P(x|y=1)\,P(y=1)}{P(x)}$$

$$= \frac{P(x|y=1)\,P(y=1)}{P(x|y=1)P(y=1) + P(x|y=0)P(y=0)}$$

$$\prod_{i=1}^{50,000} P(x_i|y=1)$$

one of the term in this product will be zero because of previous calculation

$$= \frac{0}{0+0} = \frac{0}{0} \text{ (undefined)}$$

→ But the problem is that actually

$P(x_{35000}|y=1) = 0$ is bad assumption.

→ We can't set probability of this to zero, as we haven't seen the word 'nips' before.

→ So we need to fix this problem.

* Example of basket ball team:

Date	Vs match	result (0=lose, 1=win)
2/8	Washington state	0
2/11	Washington	0
2/22	USC	0
2/24	UCLA	0
3/8	USC	0
3/15	Louisville	?

→ So, here we can't directly predict that there is no chance to win the last match just because of previous data.

→ Here comes the laplase smoothing.

→ Normally the maxlikelihood of (y=1) can be given by (informally).

$$P(y=1) = \frac{(\#\text{ of } 1)}{(\#\text{ of } 0) + (\#\text{ of } 1)}$$

(total training data)

→ Laplace just add 1 to each term above

So,

$$P(y=1) = \frac{(\#\text{ of } 1) + 1}{(\#\text{ of } 1) + 1 + (\#\text{ of } 0) + 1}$$

→ So, in our ex.

$$P(y=1) = \frac{1}{5+1+1} = 1/7$$

probability of winning next game

instead of 0.

→ More generally,

if $y \in \{1, 2, \ldots k\}$ (instead of 0 & 1 as above)

$$P(y=j) = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = j\} + 1}{m+k}$$

→ So for naive bayes,

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1, y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2}$$

and,

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1, y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2}$$

Lecture-6

* Multinomial Event model for Naive Bayes.

→ In this model, representation of email will be different.

(i) is i^{th} training example.

→ $(x_1^{(i)}, x_2^{(i)}, \ldots x_{n_i}^{(i)})$, where n_i = # of words in an email

→ So, here email# is represented by feature vector.

$x_j^{(i)} \in \{1, 2, \ldots 50000\}$, so j is index of word $x^{(i)}$ in dictionary

→ So, joint distribution is given by

$$P(x, y) = \left(\prod_{i=1}^{n} P(x_i|y)\right) P(y)$$

where n is length of an email

Page No.
Date

→ meaning of this eq^n is that as it is generative learning algo. first $P(y)$ is chosen [Email is spam or not spam], and then for that email, probability of all the words, depending on email is spam or not is calculated.

→ So, params of this,

$$\phi_{k/y=1} = P(x_j = k \mid y = 1)$$

$$\phi_{k/y=0} = P(x_j = k \mid y = 0)$$

$$\phi_y = P(y = 1).$$

→ Max. likelyhood of param. can be given by

$$\phi_{k/y=1} = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k\} \quad (+1)}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} \cdot n_i \quad (+k)}$$

laplace smoothing

→ Numerator means, take all the spam emails, and in each such email, count the word k.

→ Denomitor means that sum over entire training set, when email is spam, add the length of that email to the sum.

→ So, it is sum of length of all spam emails.

→ Note:- If x can get 'l' different values, i.e. $x \in \{1, 2, \dots l\}$, then,

max. likelihood of $x=k$ $(k<l)$ is given by

$$P(x=k) = \frac{\#\text{ of observations of } x=k}{\text{total observations.}}$$

and with laplace smoothing

$$P(x=k) = \frac{\#\text{ of obs. when } x=k+1}{\text{total obs.} + l}$$

→ How Maximum likelihood formula are derived?

$$l(\phi_{k|y=1}, \phi_{k|y=0}, \phi_y)$$

$$= \log \prod_{i=1}^{m} P(x^{(i)}, y^{(i)}; \phi_{k|y=1}, \phi_{k|y=0}, \phi_y)$$

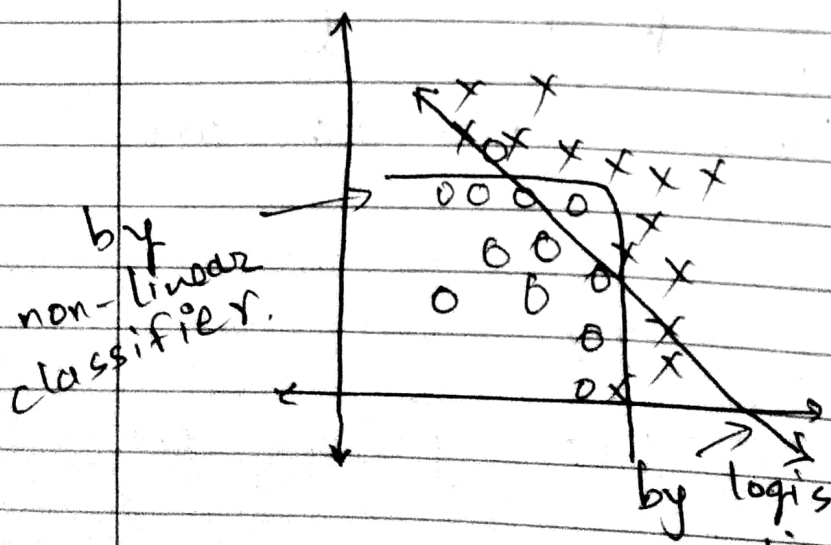
$$= \log \prod_{i=1}^{m} \prod_{j=1}^{n_i} P(x_j^{(i)} / y^{(i)}; \phi_{k|y=1}, \phi_{k|y=0}) \cdot P(y^{(i)}; \phi_y)$$

→ And by maximizing the above eq^n we can get likelihood estimation of the parameters of the models. [Maximizing means $\frac{\partial}{\partial(\text{param})} = 0$ or any other algo like newton raphson, gradient descent etc)

* Non-linear classifiers:-

→ The first classifier we've learned was logistic regression which is linear classifier

→ so, for ex. data is



→ Generative learning algorithms like Naive Bayes and Gaussian discreminant Analysis are also linear classifiers. Because under those models, we use linear classifiers as discussed in previous section of relation between GDA and Logistic Regression model.

So, as discussed before, if

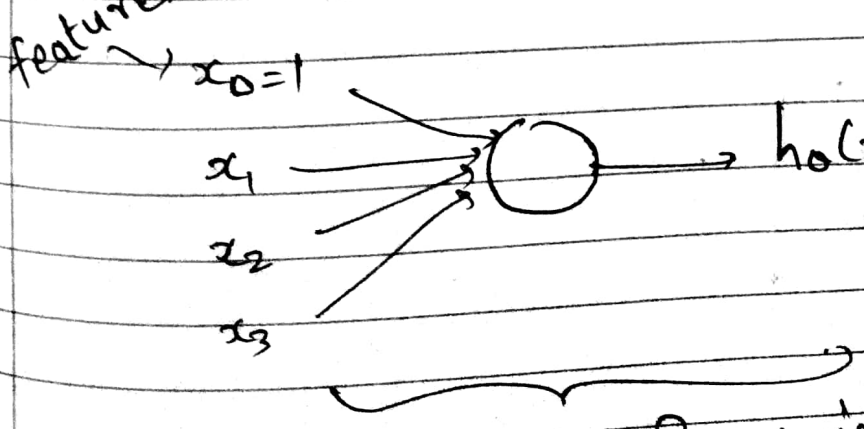
$$x|y=1 \sim \text{ExpFamily}(\eta_1)$$
$$\text{and } x|y=0 \sim \text{ExpFamily}(\eta_0)$$

↓ implies that

$p(y=1|x)$ is logistic

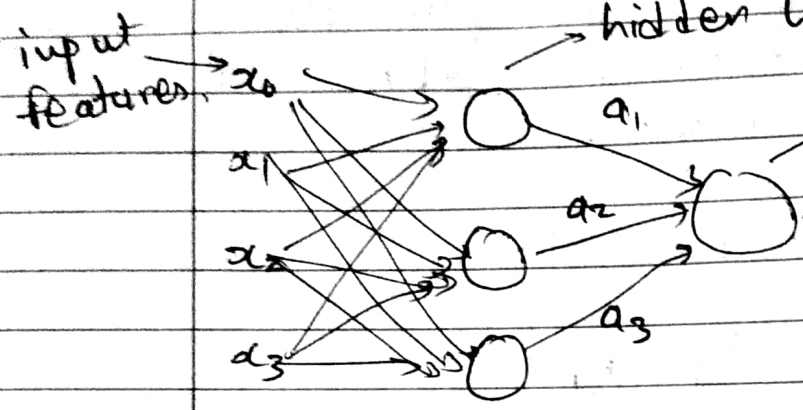
↳ $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

→ Consider a notation for further non-linear classification algorithms.



→ One way to come up with non-linear boundary, is Neural Network.

* **Neural Network:**



$a_1 = g(x^T \theta^{(1)})$ where g is sigmoid function and

$a_2 = g(x^T \theta^{(2)})$

$a_3 = g(x^T \theta^{(3)})$

$\theta^{(1)}, \theta^{(2)}, \theta^{(3)}$ are different parameters associated with each logistic unit above.

$h_\theta(x) = g(\vec{a}^T \theta^{(4)})$, where

$\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$.

$h_\theta(x)$ is function of $\theta_1, \theta_2, \theta_3$ & θ_4.

→ now, as before cost function is,

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

→ to learn the parameters $(\theta_1, \theta_2 \dots \theta_4)$ we can use gradient descent.

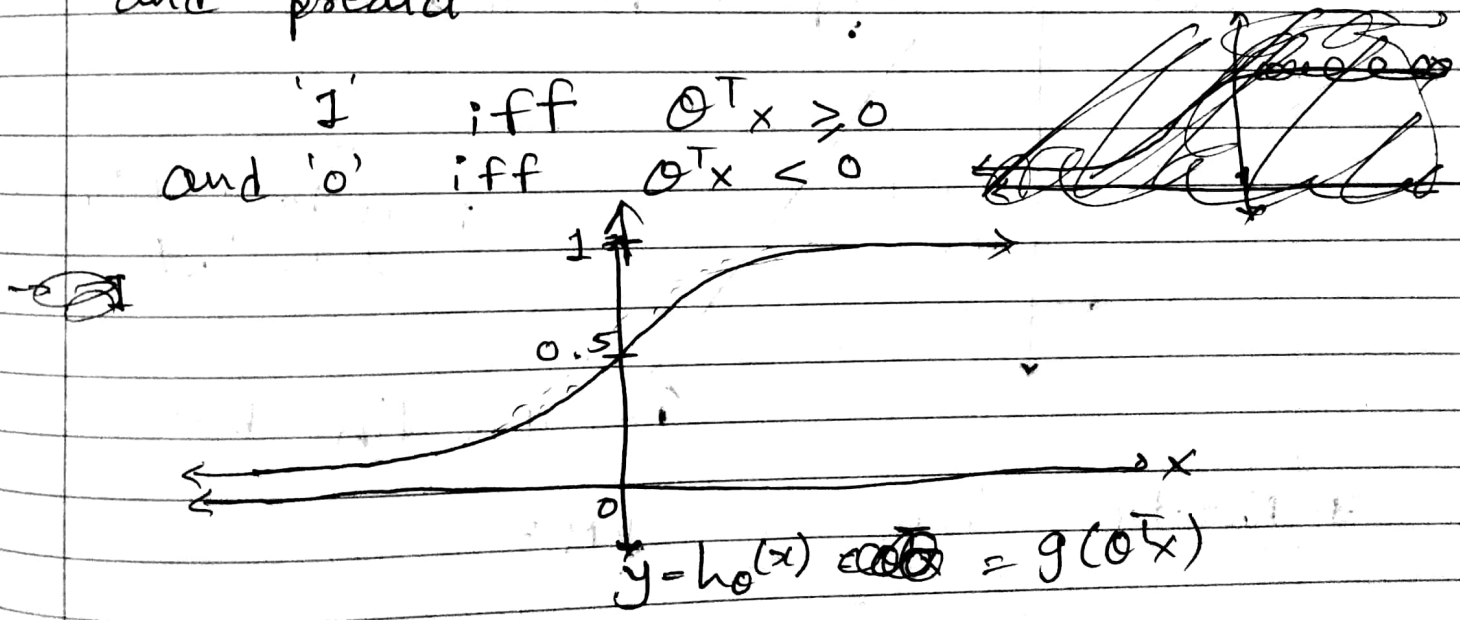
→ Now Gradient Descent on Neural Network is called Back Propogation.

* Support Vector Machines :- (SVM). [supervised learning algo, classification algo].

→ In Logistic regression, we compute $\theta^T x = y$ and predict

'1' iff $\theta^T x \geq 0$

and '0' iff $\theta^T x < 0$



→ if $\theta^T x \gg 0$, very 'confident' prediction that $y = 1$.

if $\theta^T x \ll 0$, very 'confident' prediction that $y = 0$

→ It would be nice, if

Date

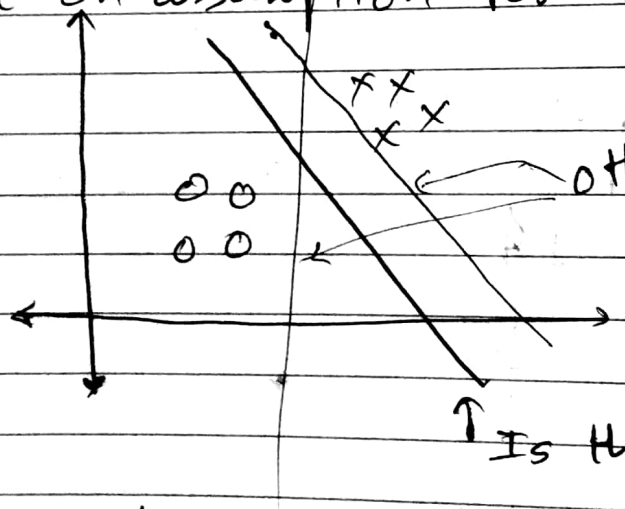
for $\forall i$, such that $y^{(i)} = 1$, we've $\theta^T x^{(i)} >> 0$

and $\forall i$, such that $y^{(i)} = 0$, we've $\theta^T x^{(i)} << 0$,

→ So, our predictions be very confident.

→ The above intution is called functional margin.

→ The 2nd intution is called Geometric Margin. If data set is linearly separable (Just an assumption for intution).



* Notation change of SVM :-

→ $y \in \{-1, +1\}$ instead of $y \in \{0, 1\}$

→ Have h (hypothesis) output values in $\{-1, +1\}$

→ $g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$

Page No.
Date

→ $h_\theta(x) = g(\theta^T x)$ $x \in R^{n+1}$ } DROP IT
$x_0 = 1$

Instead, of this,

$$h_{w,b}(x) = g(w^T x + b)$$
$b = \theta_0$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix}$$

where $w \in R^n$ and $x \in R^n$ and $b \in R$

→ Def^n:- Functional margin of a hyperplane (w, b) w.r.t training examples $(x^{(i)}, y^{(i)})$ is defined as,

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b).$$

→ Informally hyperplane means decision boundary.

If $y^{(i)} = 1$, we want $w^T x + b >> 0$
if $y^{(i)} = -1$, we want $w^T x + b << 0$

so if $y^{(i)}(w^T x^{(i)} + b) > 0$, then we've classified $(x^{(i)}, y^{(i)})$ correctly.

→ functional margin w.r.t the entire training set can be given by

$$\hat{\gamma} = \min_i \hat{\gamma}^{(i)}$$

→ But for our case let's define that we want worst case functional margin. [Large f^n margin]

→ To make large functional margin, we can just scale the parameters

$w \to 2w$

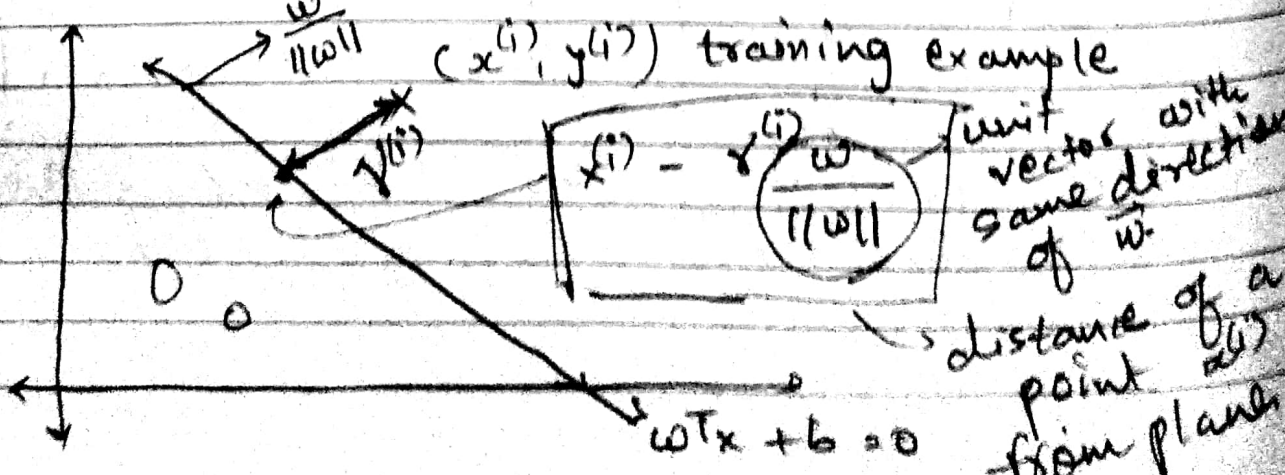
$b \to 2b$, then as per

$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$, our f^n margin will be doubled.

→ We can add normalization cond^n that

$$||w|| = 1$$

* About Geometric Margin:-



→ now the point $x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|}$ is on the hyperplane, it satisfies the eqⁿ of the plane

so,

$$w^T \left(x^{(i)} - \gamma^{(i)} \frac{w}{\|w\|} \right) + b = 0$$

→ solve above eqⁿ for $\gamma^{(i)}$,

$$w^T x^{(i)} - \gamma^{(i)} \frac{w w^T}{\|w\|} + b = 0$$

$w w^T = \|w\|^2$, so

$$\therefore w^T x^{(i)} + b = \gamma^{(i)} \|w\|$$

$$\therefore \gamma^{(i)} = \left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|}$$

→ This signifies that if we've a training set $(x^{(i)}, y^{(i)})$, then the distance b/w $x^{(i)}$ and separating hyperplane is defined by $\gamma^{(i)}$.

→ More generally, Geometric margin:-

$$\gamma^{(i)} = y^{(i)} \left[\frac{w^T}{\|w\|} x^{(i)} + \frac{b}{\|w\|} \right]$$

→ We want geometric margin to be large.

→ If $\|w\| = 1$, the

$$\hat{\gamma}^{(i)} = \gamma^{(i)}$$

↑ functional Margin ↖ Geometric Margin.

→ So, more Generally,

$$\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\|w\|}$$

→ this is correct understanding of both margins.

→ Geomatric margin w.r.t entire training set is

$$\gamma = \min_i \gamma^{(i)}$$

Max. margin classifier :- It is the learning algo, which chooses the parameters w and b so to maximize Geomatric margin.

Best explanation of this SVN is on www.svn-tutorial.com.

Page No.
Date

$$\max_{\gamma, w, b} \gamma$$

such that

$$y^{(i)}(w^T x^{(i)} + b) \geq \gamma \quad \text{for } i \in \{1, 2, \ldots, m\}, \quad \|w\| = 1$$

this condition ensures there is no training data point b/w two initially selected hyperplane

Geomatric margin does not depend on $\|w\|$.

Lecture - 7

→ Now the constraint $\|w\| = 1$ is non-convex optimization constraint. we need to get rid of this constraint.

→ So let's change the optimization problem.

since we know that

$$\gamma = \frac{\hat{\gamma}}{\|w\|},$$ we can write above problem as,

called objective

$$\max_{\gamma, w, b} \frac{\hat{\gamma}}{\|w\|},$$

subject to $y^{(i)}(w^T x^{(i)} + b) \geq \hat{\gamma}, \quad i \in \{1, 2, \ldots m\}$

this is condition of constraint.

→ Now for simplicity, let's assume that

$\hat{\gamma} = 1$,

so,

$$\min_i y^{(i)}(w^T x^{(i)} + b) = 1$$

→ Now rewriting the problem with $\hat{\gamma} = 1$

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{(same as)} \quad \max \frac{1}{\|w\|}$$

subject to $y^{(i)}(w^T x^{(i)} + b) \geq 1$

→ Now this is convex optimization problem with constraints. So we can use Lagrange multipliers to optimize it.

* Lagrange Multipliers:-

$$\min_w f(w)$$

subject to $h_i(w) = 0$, $i = 1, 2 \ldots l$

Lagrangian:-

$$\mathcal{L}(w, \beta) = f(w) + \sum_i \beta_i h_i(w)$$

where β_i = Lagrangian Multipliers.

→ To solve it, set

$$\frac{\partial L}{\partial w} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \beta} = 0$$

→ for some w^* to be a solution, it is necessary that

$\exists \beta^*$ such that

$$\frac{\partial L(w^*, \beta^*)}{\partial w} = 0 \quad \text{and}$$

$$\frac{\partial L(w^*, \beta^*)}{\partial \beta} = 0$$

→ In General,

$$\min_w f(w) \longrightarrow \text{primal problem}$$

subject to $g_i(w) \leq 0$, for $i = 1, 2 \ldots k$

and $h_i(w) = 0$ for $i = 1, 2, \ldots l$

Lagrangian:-

$$L(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w) \rightarrow (1)$$

Best Explained in Lagrange duality lecture

→ Define,

$$\theta_p(\omega) = \max_{\substack{\alpha, \beta \\ \alpha_i \geq 0}} L(\omega, \alpha, \beta)$$

Consider :

$$p^* = \min_{\omega} \max_{\substack{\alpha, \beta \\ \alpha_i \geq 0}} \mathcal{L}(\omega, \alpha, \beta)$$

$$= \min_{\omega} \theta_p(\omega)$$

→ Primal Problem

→ $\theta_p(\omega)$:- If $g_i(\omega) > 0$, then $\theta_p(\omega) = \infty$

how $\theta_p(\omega) = \infty$?

→ From previous page Eqⁿ 1,

$$\sum_{i=1}^{k} \alpha_i g_i(\omega),$$

we can scale α_i to arbitrarily large to make $\theta_p(\omega)$ ∞

→ And if $h_i(\omega) \neq 0$, then $\theta_p(\omega) = \infty$ (Same reason)

→ otherwise, $\theta_p(w) = f(w)$ [By setting Lagrangian terms to zero].

→ Thus,

$$\theta_p(w) = \begin{cases} f(w) & \text{if constraints are satisfies } (g_i, h_i) \\ = \infty & \text{otherwise.} \end{cases}$$

So, $\min_w \theta_p(w)$ = Original problem

* **Dual Problem:-**

$$\theta_D(\alpha, \beta) = \min_w \mathcal{L}(w, \alpha, \beta)$$

$$d^* = \max_{\substack{\alpha \geq 0 \\ \beta}} \min_w \mathcal{L}(w, \alpha, \beta) = \max_{\substack{\alpha \geq 0 \\ \beta}} \theta_D(\alpha, \beta)$$

see the diff. with previous problem.

Dual problem.

$d^* \leq p^*$ is informally,

$\max(\min(\ldots)) \leq \min(\max(\ldots))$

Date

- but sometime,

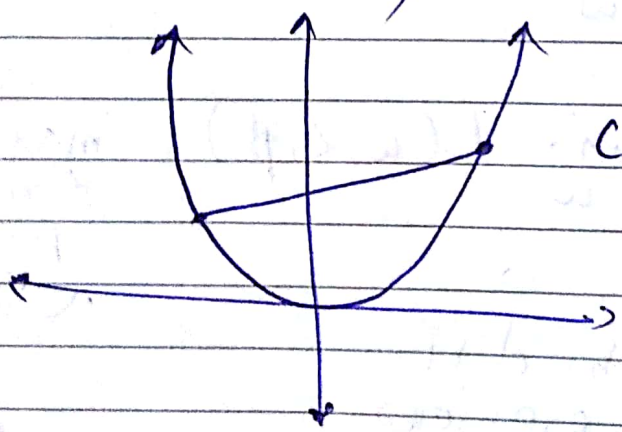
$$d^* = p^*$$

→ It turns out sometimes it's easy to solve dual problem instead of primal problem

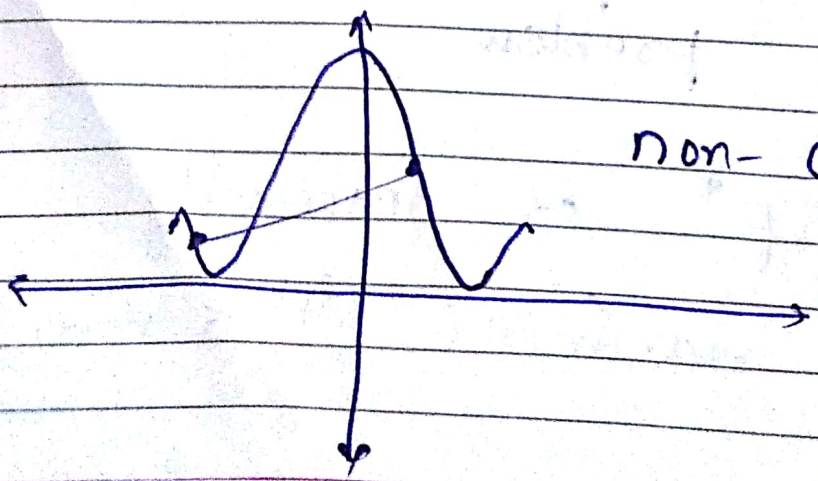
→ So here in SVM, we'll derive dual problem from primal problem and then solve dual problem.

→ Conditions for which dual and primal problems give the same solution.

Let f be convex (Hessian $H \geq 0$ or +ve semidefine)



convex (local minima is equal to global minima)



non-convex.

→ suppose,

h_i is affine, $[h_i(\omega) = \alpha_i^T \omega + b_i]$

and

g_i is strictly feasible, $[\exists \omega_i \text{ s.t } \forall i \; g_i(\omega) < 0]$.

→ Then $\exists \omega^*, \alpha^*, \beta^*$ such that ω^* solves the primal problem and α^*, β^* solves dual problem, then.

$$p^* = d^* = \mathcal{L}(\omega^*, \alpha^*, \beta^*)$$

farther,

$$\frac{\partial \mathcal{L}}{\partial \omega}(\omega^*, \alpha^*, \beta^*) = 0$$

$$\frac{\partial \mathcal{L}}{\partial \beta}(\omega^*, \alpha^*, \beta^*) = 0$$

and $\boxed{\alpha_i^* g_i(\omega) = 0}$ [KKT complementarity condition].

→ $g_i(\omega^*) \leq 0$ and $\alpha_i^* \geq 0$

from this we can say that if product is zero, then one of the α_i^* or $g_i(\omega)$ is zero.

Page No.
Date

* Karush-Kuhn-Tucker complementarity cond' (KKT)

→ It implies that,

If $\alpha_i^* > 0$ then $g_i(w^*) = 0$

[bcz $\alpha_i^* g_i(w^*) = 0$]

→ Usually, $\alpha_i^* \neq 0 \iff g_i(w^*) = 0$ [Atleast one of them is zero].

→ When $g_i(w^*) = 0$, then, $g_i(w)$ is an "active constraint".

* Note:-

	In Dual Problem	In Actual SVM
Lagrange Multipliers	α_i, β_i	α_i
Parameters	w	w, b

→ Actual SVM minimization problem is,

$$\min_{w} \frac{1}{2}\|w\|^2 \quad \left[\text{½ and square is for math. simplicity}\right]$$

s.t.

$$y^{(i)}(w^T x^{(i)} + b) \geq 1 \quad , \quad i = 1, 2, \ldots, m$$

→ so; constraint can be rewritten as,

$$g_i(w,b) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0$$

→ Implication of this constraint $[g_i(w,b)]$ in terms of KKT conditions.

→ we've

$$\alpha_i > 0 \Rightarrow g_i(w,b) = 0 \quad [\text{"Active const."}]$$

$$\Leftrightarrow (x^{(i)}, y^{(i)}) \text{ has functional margin} = 1.$$

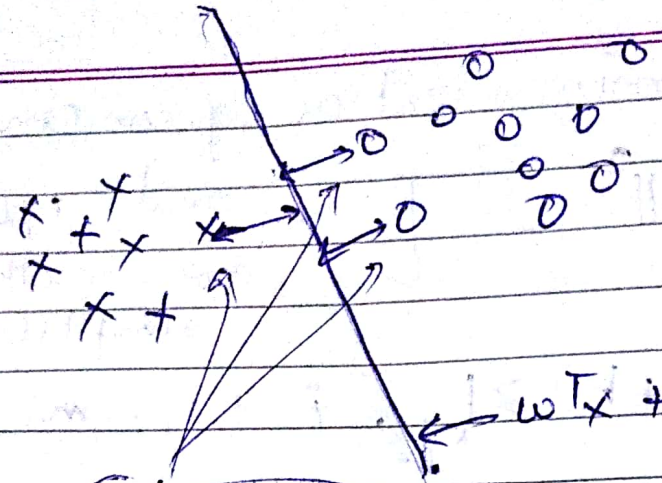
because,

$$g_i(w,b) = 0 \;\; \therefore = -y^{(i)}(w^T x^{(i)} + b) + 1$$

$$\therefore y^{(i)}(w^T x^{(i)} + b) = 1$$

$\therefore$ functional Margin $= 1$.

→ Geometrically what this means can be given by :-



functional margin = 1 and other examples have functional margin > 1.

> Usually, $\alpha_i \neq 0$ $[\alpha_i > 0]$

→ Here the 3 points with functional margin = 1 are called support vectors. [few support vectors]
So, usually $\alpha_i = 0$ for most of the support vectors.

→ $$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(y^{(i)}(w^T x^{(i)} + b) - 1\right)$$ → (1)

Dual problem:-

$$\Theta_D(\alpha) = \min_{w,b} \mathcal{L}(w, b, \alpha)$$

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \overset{\text{set}}{=} 0.$$

$$\boxed{w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}}$$

$$\rightarrow \frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{m} \alpha_i y_i \overset{\text{set}}{=} 0$$

→ substitute these values back in (1)

$$\mathcal{L} = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i \left(y^{(i)} \left(w^T x^{(i)} + b \right) - 1 \right)$$

$$\hookrightarrow \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T \left(\sum_{j=1}^{m} \alpha_j y^{(j)} (x)^{(j)} \right)$$

$w^T w = \|w\|^2 = w \cdot w$ [Dot product]

So,

$$\mathcal{L} = \frac{1}{2} w \cdot w - \sum_{i=1}^{m} \alpha_i \left(y^{(i)} \left(w \cdot x + b \right) - 1 \right)$$

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

$$\sum_{i=1}^{m} \alpha_i \left[y_i \left(\left(\sum_{j=1}^{m} \alpha_j y^{(j)} x^{(j)} \right) \cdot x_i + b \right) - 1 \right]$$

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

$$\sum_{i=1}^{m} \alpha_i y_i \left(\left(\sum_{j=1}^{m} \alpha_j y_j \alpha_j \right) \cdot x_i + b \right)$$

$$= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i x_j - \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i x_j$$

$$- b\sum_{i=1}^{m}\alpha_i y_i + \sum_{i=1}^{m}\alpha_i$$

$$= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j x_i x_j y_i y_j - b\sum_{i=1}^{m}\alpha_i y_i$$

Now, $\sum_{i=1}^{m}\alpha_i y_i = 0$,

call it

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

→ So, Dual problem is:

$$\max_{\alpha} w(\alpha)$$

s.t

$\alpha_i \geq 0$

and $\sum_{i=1}^{m} y_i \alpha_i = 0$

→ Interpretation of $\sum_{i=1}^{m} y_i \alpha_i = 0$,

If $\sum_i y_i \alpha_i \neq 0$, then $\theta_D(\alpha) = -\infty$, since we are maximizing $\theta_D(\alpha)$, we better choose $\sum_i y_i \alpha_i = 0$ to prevent penalty.

→ So, if $\sum_i y_i \alpha_i = 0$, then $\theta_D(\alpha) = w(\alpha)$

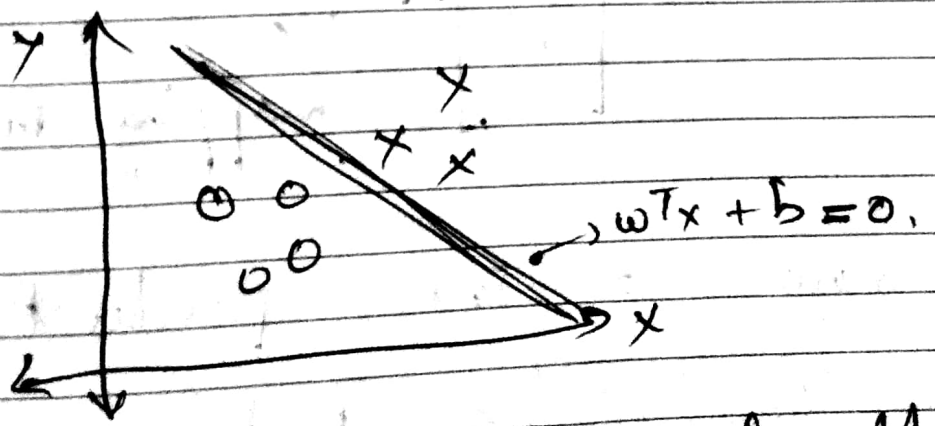
→ Solve $w(\alpha)$ for α, then

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}.$$

→ then plug α and w to primal problem and solve for b,

$$b = \frac{\max_{y_i=-1} (w^T x^{(i)}) + \min_{y_i=1} (w^T x^{(i)})}{2}.$$

→ Intution is, place the hyperplane in the exact middle of +ve & -ve example. [We've the orientation of plane given by w, b just gives it's place].



→ This is called Optimal Margin Classifier or SVM.

Lecture-8

Page No.
Date 29 | 5 | 18

→ As mentioned in dual problem,

$$\max_{\alpha} W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

→ we only care about inner product or dot product of feature vectors $x^{(i)}$ and $x^{(j)}$, the actual values are not required.

* Idea behind Kernels:-

→ Let's say we've an attribute $x \in \mathbb{R}$ (for ex. living area of house).

→ Now take these feature and map it to reacher set of feature vectors.

$$x \xrightarrow{\phi} \begin{bmatrix} x \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} = \phi(x)$$

mapping function.

→ So, now our inner product $\langle x^{(i)}, x^{(j)} \rangle$ will be replaced by

$$\langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$$

→ which is inner product in higher dimensions.

→ some times $\phi(x)$ is very high dimensional (∞ dimensional), so we can't compute this inner product efficiently.

→ Here Kernel comes into the picture

$$K(x^{(i)}, x^{(j)}) = \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$$

→ Kernels computes this inner product inexpensively, without actually transforming the original vector to higher dimensions. [without mapping]

* How this is done using Kernels?

→ Let's say we've two vectors $x, z \in R^n$ and ~~ken~~ Kernel defined as,

$$K(x,z) = (x^T z)^2 = \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_j z_j\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} (x_i x_j)(z_i z_j)$$

$$= (\phi(x))^T (\phi(z))$$

→ where,

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix}$$ for $n=3$.

$\phi(z)$ = same as $\phi(x)$.

→ So, this $(x^T z)^2$ kernel calculates the inner product without even transforming or mapping the original vector to ϕ. So it is highly efficient idea.

→ So, we need $O(n^2)$ to compute just $\phi(x)$.

→ And to compute $K(x, z)$, it needs $O(n)$ time.

→ one general example of kernels is,

$K(x, z) = (x^T z + c)^d$, which is $\binom{n+d}{d}$ features of all monomials upto degree d.

Page No.
Date

* How to come up with a good kernel in actual machine learning problems?

→ if we've two feature vector and mappings,

$x \longrightarrow \phi(x)$, $z \rightarrow \phi(z)$, and

$\langle \phi(x), \phi(z) \rangle$ is inner product in higher dimensions, then one intitution is,

→ if x and z are similar kind of features (vectors), then they are pointing in similar direction, so inner product or dot product will be higher. because of

$x \cdot z = x z \cos\theta$ and

$\cos\theta \rightarrow 1$, when $\theta \rightarrow 0$.

→ And similarly, if data or vectors are very dissimilar, then dot product will be very low.

So, for particular problem, choose,

$k(x,z)$ { large if x, z are similar
Small, if x, z are dissimilar.

Gaussian kernel.

Date

* $K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$, so,

$\exists \phi$ such that $K(x,z) = \langle \phi(x), \phi(z) \rangle$?
or in other words, is this kernel valid?

→ Suppose K is kernel., then let $\{x^{(1)}, \ldots x^{(m)}\}$ be given and,
Let $K \in \mathbb{R}^{m \times m}$ [K is matrix],
such that

$K_{ij} = K(x^{(i)}, x^{(j)})$, then for any vector $z \in \mathbb{R}^m$,

consider,

$$z^T K z = \sum_i \sum_j z_i K_{ij} z_j$$

$$= \sum_i \sum_j z_i \phi(x^{(i)})^T \phi(x^{(j)}) z_j$$

$$= \sum_i \sum_j z_i \sum_k (\phi(x^{(i)}))_k (\phi(x^{(j)}))_k z$$

$$= \sum_k \sum_i \sum_j z_i (\phi(x^{(i)}))_k (\phi(x^{(j)}))_k z_j$$

paytms - 011 3996699

Page No.
Date

$$= \sum_k \left(\sum_i z_i \phi(x^{(i)})_k \right)^2 \geq 0$$

So, $K \geq 0$, so matrix K is positive semi definite matrix.

→ So, from this, we can say that if K is valid kernel, then matrix K is +ve semi-definite.

→ the converse also holds true: that is if matrix K is +ve semi-definite then the kernel corresponding to it is valid kernel.

→ So, Theorem (Mercer) :- Let $K(x, z)$ be given, then K is valid kernel, or mercer kernel (i.e. $\exists \phi$ s.t $K(x, z) = \phi(x)^T \phi(z)$),

iff, for any set of m examples $\{x^{(1)}, \ldots, x^{(m)}\}$ $(m < \infty)$, the kernel matrix $K \in \mathbb{R}^{m \times m}$ is symmetric +ve semidefinite.

2j → Converse is also true as mentioned previously.

→ Ex. of invalid kernel,

$K(x,z) = -1 \neq \phi(x)^T \phi(x)$.

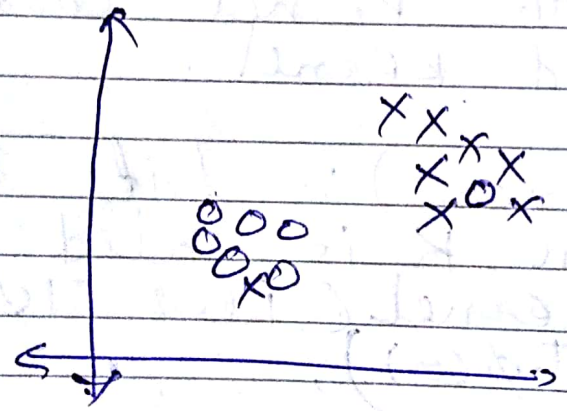
because inner product always ≥ 0.

→ Intution for this kernel is given best in SVM book.

* L1 Norm Soft Margin SVM :-

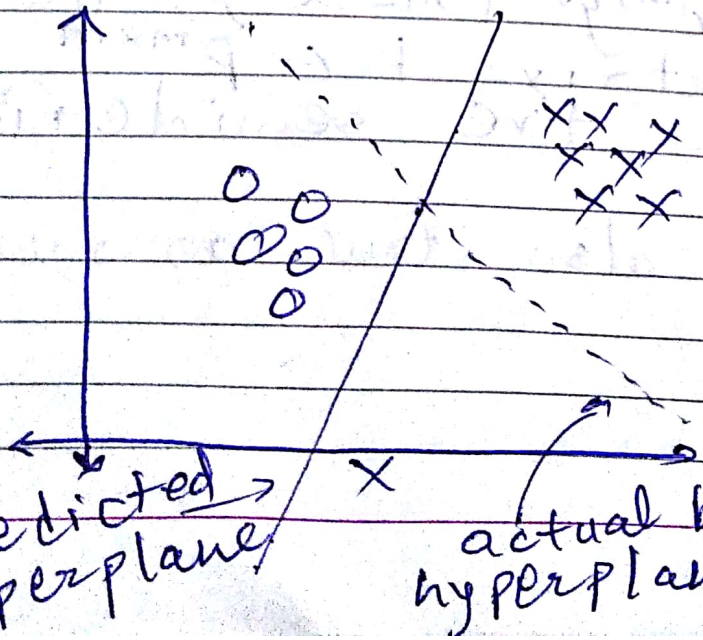
→ There are few cases, where hard margin SVM does not work well.

1)



⇒ Here hard margin SVM won't figure out the hyper plane, even if only one ex. is not at right place.

2)



→ here, due to single ex. whole classification will give wrong predictions

ξ (zeta).

Page No.
Date

→ The updated formulations for soft-margin SVM is,

$$\min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$

such that

→ functional margin $\begin{cases} y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \quad \text{for any } i = 1, 2, \ldots m \end{cases}$

→ If $y^{(i)}(w^T x^{(i)} + b) > 0$ that means SVM has classified the example correctly. and it's ≤ 0, we miss classified it. so, here we can set some of $\xi_i \geq 1$, we are allowing SVM to miss classify for some of the examples.

→ Now, it is not encouraged, as we are adding penalty to our final goal by

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$$

from this, the more $\xi_i \geq 1$, less minimized our goal will be.

→ Here 'C' controls the effect of penalty.

→ Now, corresponding to the set of new constraints, our new generalized Lagrangian will be,

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left(y^{(i)}(w^T x^{(i)} + b) - 1 + \xi_i\right) - \sum_{i=1}^{m} r_i \xi_i$$

[from $\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k}\alpha_i g_i(w) + \sum_{j=1}^{l}\beta_i h_i(w)$]

→ So, here α and r are lagrange multipliers, and

$$g_i(w) = y^{(i)}(w^T x^{(i)} + b) - 1 + \xi_i$$
$$h_i(w) = \sum_{i=1}^{m}\xi_i$$

} constraints

and $f(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$

→ And from this we can derive dual of this optimization problem same as previously, we get following. fn.

$$\max_{\alpha} \; W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i}^{m} \sum_{j}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

subject to

$$\sum_{i=1}^{m} y^{(i)} \alpha_i = 0 \quad \text{and}$$

$$0 \leq \alpha_i \leq C \quad \text{for} \quad i = 1, 2, \ldots, m.$$

Note that only this changes compared to previous dual problem.

→ convergence conditions,

$\alpha_i = 0 \Rightarrow y^{(i)}(w^T x^{(i)} + b) \geq 1$

$\alpha_i = C \Rightarrow y^{(i)}(w^T x^{(i)} + b) \leq 1$

$0 \leq \alpha_i < C \Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1$

→ To actually solve the above optimization problem, we need efficient algo. (SMO).

Page No.
Date

* Digression :- Coordinate ascent :-

$\max W(\alpha_1, \alpha_2, \ldots \alpha_m)$ [No constraints on α_i's]

→ In coordinate ascent, maximization is done by maximizing W along one direction at a time or using one α_i at a time. I.e. solving univariate opt. problem in loop.

→ Algo. can be given by :-

Repeat {

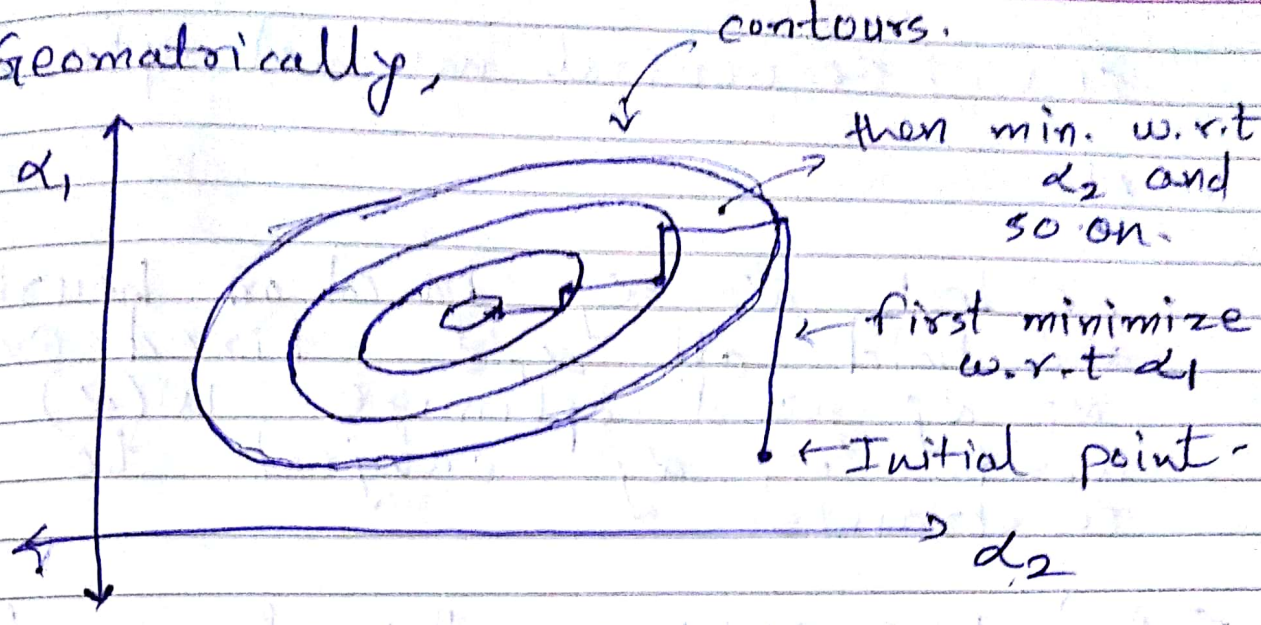
for (i=1 to m) → order of choosing α can be random.

$\alpha_i = \underset{\hat{\alpha}_i}{\arg\max}\ W(\alpha_1, \ldots \hat{\alpha}_i, \alpha_{i+1} \ldots \alpha_m)$

}

Informally this means hold everything except α_i fixed.

→ Geomatrically,



→ Compared to Newton's method, this take lot more steps, but the main advantage is sometime the object $W(\alpha_i)$ is very inexpensive w.r.t any one of the parameters (α_i).

→ Now, for applying this algo. to our SVM softmargin problem, the problem is ~~this~~ with basic form of this algo., having no. constraints over α_i. But we've constraints over α_i in softmargin SVM., which is

$$\sum_{i=1}^{m} y^{(i)} \alpha_i = 0.$$

→ So, changing 1 α_i at a time won't work. Now we'll change 2 α_i at a time, which algo. is called

ζ (zee)

SMO (sequencial minimal optimization)

→ Algo:-

Select α_i, α_j based on heuristic and hold all α_i's fixed except α_i, α_j and optimize $W(\alpha)$ w.r.t. α_i, α_j subject to constraints.

→ And keep running the algo. until the convergence criteria is meet. [convergence cond" as mentioned previously].

* How α_i and α_j are updated to optimize $W(\alpha)$?

→ In every step, we respect the constraint

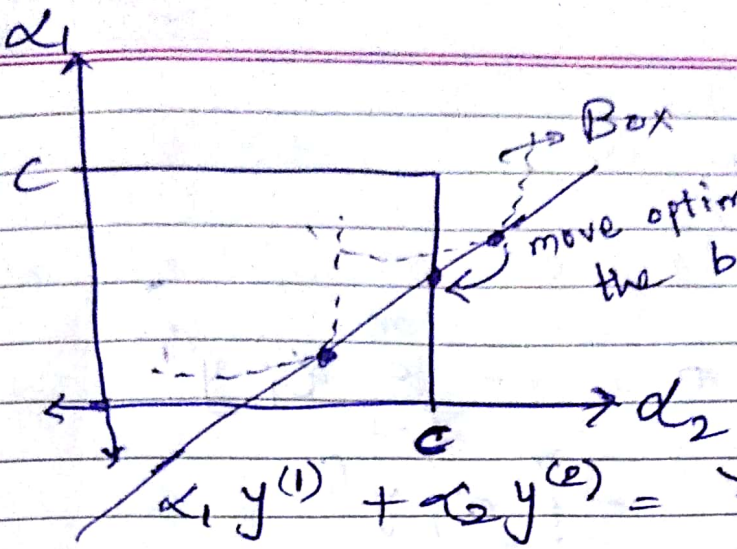
$$\sum_i \alpha_i y^{(i)} = 0.$$

→ Let consider $i=1$ and $j=2$ for sake of simplicity, so,

$$\alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^{m} \alpha_i y^{(i)} = \zeta$$

→ also other constraint is

$$0 \leq \alpha_i \leq C. \quad \text{[Box constraint]}$$



$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = \xi$$

$$\text{so, } \alpha_1 = \frac{\xi - \alpha_2 y^{(2)}}{y^{(1)}}$$

$$w(\alpha_1, \alpha_2 \cdots \alpha_m)$$

$$= w\left(\frac{\xi - \alpha_2 y^{(2)}}{y^{(1)}}, \alpha_2, \alpha_3 \cdots \alpha_m\right)$$

→ Now we've $\alpha_3, \alpha_4 \cdots \alpha_m$ are fixed, so this fⁿ is only 1-D quadric function of α_2.

$$= a\alpha_2^2 + b\alpha_2 + c$$

→ And by optimizing above fⁿ, will give optimized value of $w(\alpha_1, \alpha_2 \cdots \alpha_m)$ w.r.t α_2.

→ As underlined on previous page why changing one α_i at a time won't work for constrained α_i's?

→ The answer is,

$$\sum_i \alpha_i y^{(i)} = 0$$

$$\therefore \alpha_1 y^{(1)} = -\sum_{i=2}^{m} \alpha_i y^{(i)}$$

$$\therefore \alpha_1 = \frac{-\sum_{i=2}^{m} \alpha_i y^{(i)}}{y^{(1)}}$$

→ So, α_1 is written as function of $\alpha_2, \alpha_3 \cdots \alpha_m$ and if we keep $\alpha_2, \alpha_3 \cdots \alpha_m$ fixed, then we can't change α_1.

→ But in case of changing 2 α_i (α_1 and α_2 for example), then if we change α_1, α_2 will adjust accordingly without affecting other $\alpha_3, \alpha_4 \cdots \alpha_m$.

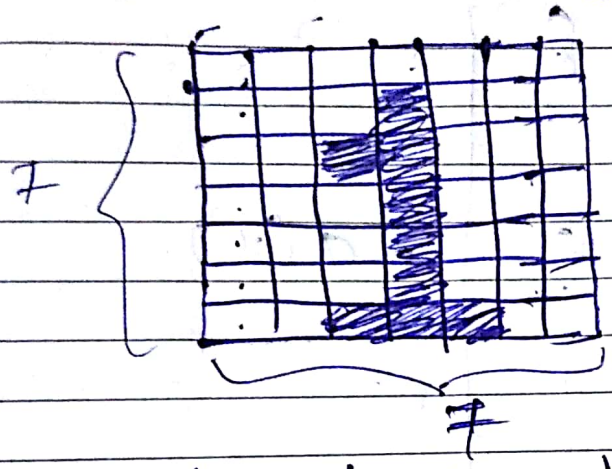
→ And this is the reason why we have to change 2 α_i at a time in constrained problem.

→ We are changing minimum possible α_i's at a time, and that's why it is called sequencial minimal optimization.

Page No.
Date

Applications of SVMs:-

→ In handwritten digit recognizion, it gives same performance as best trained Neural network.



7x7 image.

→ This results best when choosing kernel as following,

$$K(x,y) = (x^T y)^d \quad \text{or}$$

gaussian kernel $K(x,z) = \exp\left(\frac{-\|x-z\|^2}{2\sigma^2}\right)$

→ Another application is in classification of protines.

→ Protein sequence is sequence of Amino Acids, represented by A, B, C, ---, Z.

→ So, for a random length protein sequence,

BAJTSIAIBAJTAU...

→ our feature vector $\phi(x)$ will be constructed by writing all comb^n of length 4 amino Acids.

~~AAAA~~	0
~~AAAB~~	0
⋮	⋮
BAJT	2
⋮	⋮
TSTA	1
⋮	⋮
ZZZZ	0

$= \phi(x)$

So, $\phi(x) \in R^{(20^4)}$

$\in R^{160000}$

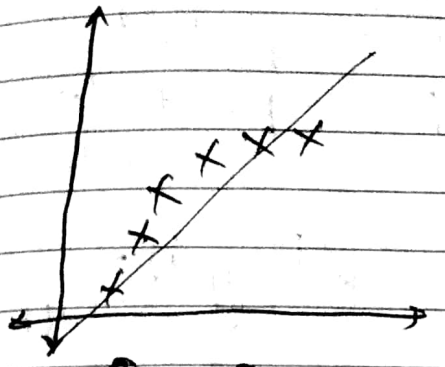
→ Very high dimensional feature vector. but can be calculated $\phi(x)^T \phi(x)$ by efficient dynamic programming algo.

Lecture - 9

Page No.
Date 3/5/18

* Learning Theory :-

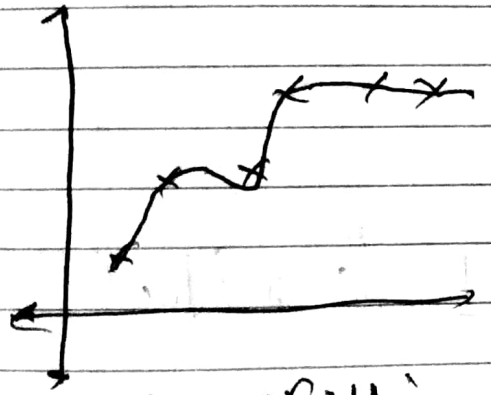
-> Bias - variency trade off.



→ This fails to fit the quadratic nature of the data so it is unfit or bias.

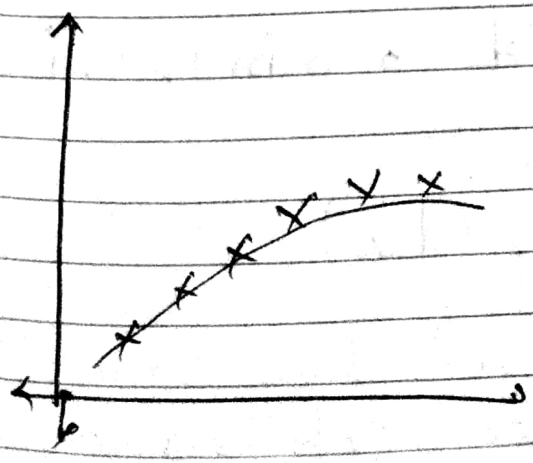
$\theta_0 + \theta_1 x$

"unfit" or "bias"



→ $\theta_0 + \theta_1 x + \theta_2 x \dots$

overfitting or high variance



best fit.

Page No.
Date 1 6 18

* Linear classifier :-

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

$g(z) = 1\{z \geq 0\}$ $(y \in \{0,1\})$

→ nothing but perceptron learning algo.

$S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ $(x^{(i)}, y^{(i)}) \sim_{IID} D$

training set

some distribution like guassian

→ Training Error of h_θ :-

$$\hat{\varepsilon}(h_\theta) = \hat{\varepsilon}_S(h_\theta) = \frac{1}{m} \sum_{i=1}^{m} 1\{h_\theta(x^{(i)}) \neq y^{(i)}\}$$

→ Training Error is also called risk.

* ERM (Empirical Risk minimization) :-

$$\hat{\theta} = \arg\min_{\theta} \hat{\varepsilon}_S(h_\theta)$$

→ This problem is solved by logistic regression, SVM etc. Learning algo.

→ This is non-convex optimization problem. But SVM, Logistic regression solves this ~~log~~ as convex optimization problem.

* Hypothesis class $H = \{h_\theta : \theta \in R^{n+1}\}$.
  ↳ class of all hypothesis, from which all learning algo. choose from.

each $h_\theta : X \to \{0,1\}$
- X → input domain
- $\{0,1\}$ → range
- → class of all linear classifiers
- → In general it can be any set like class of neural networks

→ So, ERM can be written as,

$$\hat{h} = \arg\min_{h \in H} \hat{\varepsilon}_S(h)$$

→ In training a model, our goal is not to minimize error while predicting on training data set. But our goal is to make correct predictions (min. error) on test data set (the data which is unseen to the model while training).

→ So, Generalization Error :-

$$\varepsilon(h) = P_{(x,y) \sim D}\left(h(x) \neq y\right)$$

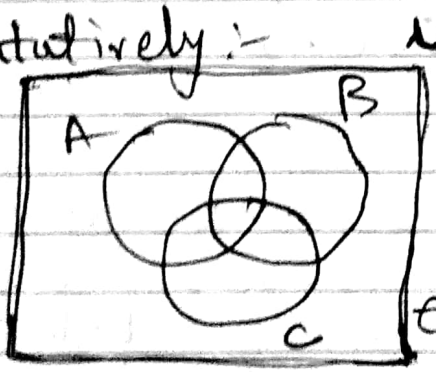
new example → mistable the new example.

→ some imp. lemma :-

1) Union Band :- Let $A_1, A_2, A_3 \ldots A_k$ be k events (not necessarily independent), then

$$P(A_1 \cup A_2 \cup A_3 \ldots \cup A_k) \leq P(A_1) + P(A_2) + \ldots + P(A_k)$$

→ Intuitively :-



→ Area of the whole shape is less then equal to area of the individual circles A, B, C.

→ hat on any symbol is estimation of that. for ex. $\hat{\epsilon}$, $\hat{\phi}$ etc.

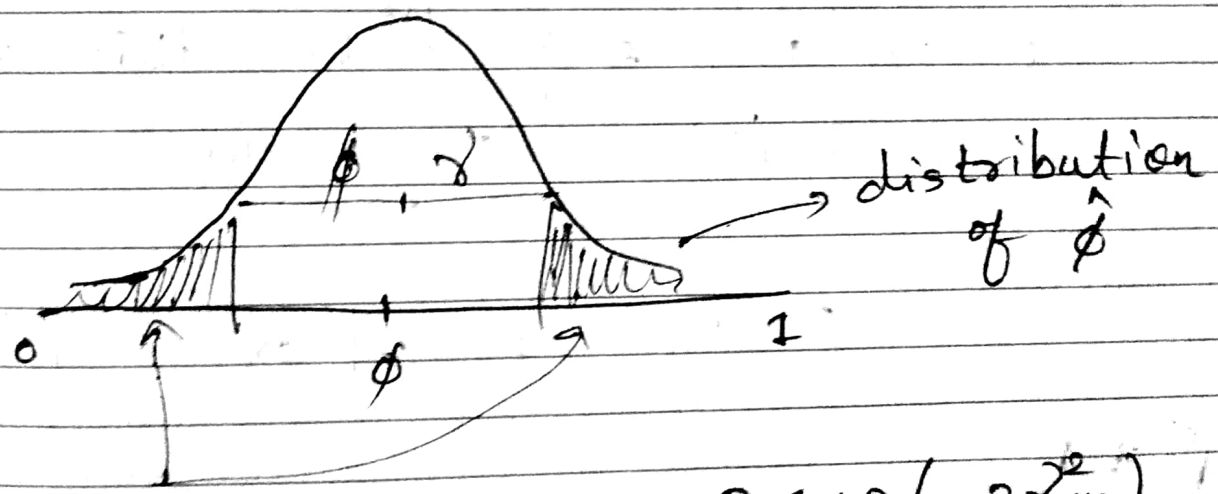
Page No.
Date

2) Hoeffding Inequality :- Let $z_1, z_2 \cdots, z_m$ be m IID Bernoully (ϕ) random variables ($P(z_i = 1) = \phi$),

Let $\hat{\phi} = \frac{1}{m} \sum_{i=1}^{m} z_i$ and Let for any $\gamma > 0$ be fixed, then

$$P(|\hat{\phi} - \phi| > \gamma) \leq 2\exp(-2\gamma^2 m)$$

→ Intutively :-



this mass $\leq 2\exp(-2\gamma^2 m)$

→ $2\exp(-2\gamma^2 m)$ decreases exponentially with m. bez γ is fixed.

Page No.
Date 2 6 |4

→ ERM for finite hypothesys class H:-

$H = \{h_1, h_2, \ldots, h_k\}$: k hypothesys.

each h_i is a function

$$\hat{h} = \arg\min_{h_i \in H} \hat{\varepsilon}_s(h_i)$$

↳ This means ERM picks the h_i, with lowest training error.

→ Need to prove that minimizing training error also minimizes the generalization error.

Strategy :-1) $\hat{\varepsilon} \approx \varepsilon$ ^ first show this result.

$\hat{\varepsilon}$: training error

ε : generalizing error.

2) show bound on $\varepsilon(\hat{h})$ (also generalization error)

1st step

→ So, According to above steps, let's start proof.

→ fix any $h_j \in H$.

Define $z_i = 1\{h_j(x^{(i)}) \neq y^{(i)}\} \in \{0,1\}$

Training error is also called in-sample error, and Generalization is also called out-sample error.

Page
Date

$P(z_i = 1) = \varepsilon(h_j)$ → generalization error of h_j.

probability of misclassification.

→ so, z_i is bernoully random variable with mean $\varepsilon(h_j)$.

→ why mean is $\varepsilon(h_j)$?

bcz according to bernoulli distribution,

$P(x=1) = p$ and
$P(x=0) = 1-p$.

so, mean = expected value

so, $E[x] = P(x=1)\cdot 1 + P(x=0)\cdot 0$
$= p \cdot 1 + q \cdot 0$

$E[x] = p = \text{mean}$.

→ training error $\hat{\varepsilon}(h_j) = \frac{1}{m} \sum_{i=1}^{m} z_i$

$= \frac{1}{m} \sum_{i=1}^{m} 1\{h_j(x^{(i)}) \neq y^{(i)}\}$.

→ mean of each $z_i = \varepsilon(h_j)$.
= generalization error.

Page No.
Date

→ By hoeffding inequality:-

$$P(|\varepsilon(h_j) - \hat{\varepsilon}(h_j)| > \gamma) \leq 2\exp(-2\gamma^2 m)$$

→ Intutively this means that when we've large training examples, then the m will large and the R.H.S will be small.

→ so, the absolute difference between training error and generalization error is very small.

so, $P(|\varepsilon(h_j) - \hat{\varepsilon}(h_j)| > \gamma) \approx 0$

→ We've proved this for a fix hypothesys h_j, now we want to prove it for entire H or for h_i, $i = 1, 2 \ldots k$.

→ Let define A_j = event such that

$$|\varepsilon(h_j) - \hat{\varepsilon}(h_j)| > \gamma$$

so, $P(A_j) \leq 2\exp(-2\gamma^2 m)$

→ $P(\exists h_j \in H, |\varepsilon(h_j) - \hat{\varepsilon}(h_j)| > \gamma)$

$= P(A_1 \cup A_2 \cup \ldots \cup A_k) \leq \sum_{i=1}^{k} P(A_i)$

[By Union Bound]

$$\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m)$$

and $\sum_{i=1}^{k} 2\exp(-2\gamma^2 m)$

$$= 2\exp(-2\gamma^2 m) \sum_{i=1}^{k} 1.$$

$$= 2k\exp(-2\gamma^2 m)$$

→ Taking negation $(1-P)$ at both sides,

$$1 - P(\exists h_j \in H \,/\, |\varepsilon(h_j) - \hat{\varepsilon}(h_j)| > \gamma)$$

$$\geq 1 - 2k\exp(-2\gamma^2 m)$$

$$\therefore P(\text{not } \exists h_j \in H \,/\, |\varepsilon(h_j) - \hat{\varepsilon}(h_j)| > \gamma)$$

$$\geq 1 - 2e^{-2\gamma^2 m}$$

$$\therefore P(\forall h_j \in H \,/\, |\varepsilon(h_j) - \hat{\varepsilon}(h_j)| \leq \gamma)$$

$$\geq 1 - 2ke^{-2\gamma^2 m}$$

→ So, with the probability $1 - 2ke^{-2\gamma^2 m}$, $\hat{\varepsilon}(h)$ will be within γ of $\varepsilon(h)$ for all $h \in H$.

→ This is called Uniform convergence.
→ γ is constant and above result is true for any value of γ.

m = training set size

Page No.
Date

→ here we proved that given γ and m, what is the probability of uniform convergence.

→ Another equivalent form is given γ and probability δ (delta) of making a large error, what should be the value of m?

$$\delta = 2k e^{-2\gamma^2 m}$$

$$\therefore \frac{\delta}{2k} = e^{-2\gamma^2 m}$$

$$\therefore \frac{2k}{\delta} = e^{2\gamma^2 m}$$

$$\therefore 2\gamma^2 m = \ln \frac{2k}{\delta}$$

$$\therefore m = \frac{1}{2\gamma^2} \ln \frac{2k}{\delta}$$

So long as $m \geq \frac{1}{2\gamma^2} \ln \frac{2k}{\delta}$, then with probability $1-\delta$, we've $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for all $h \in H$.

→ This is called "Sample Complexity" bound.

→ In practice

$$\forall k, \quad \log k \leq 30$$

* Another form is "Error bound", here we solve for γ.

$$\therefore \gamma = \sqrt{\frac{1}{2m} \ln \frac{2k}{\delta}}$$

→ So with the probability $1-\delta$, we've that $\forall h \in H$,

$$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \sqrt{\frac{1}{2m} \ln \frac{2k}{\delta}} = \gamma$$

2nd step

Let's assume $\forall h \in H, \; |\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ → (1)

→ Can we prove something about $\varepsilon(\hat{h})$? where $\hat{h} = \underset{h \in H}{\arg\min}\, \hat{\varepsilon}(h)$ → (2)

→ Also let's define $h^* = \underset{h \in H}{\arg\min}\, \varepsilon(h)$ → (3)

so h^* is best possible hypothesys in case of min. generalization error

$$\varepsilon(\hat{h}) \leq \hat{\varepsilon}(\hat{h}) + \gamma \quad \text{by } (1)$$
$$\leq \hat{\varepsilon}(h^*) + \gamma \quad \text{by } (2)$$
$$\leq (\varepsilon(h^*) + \gamma) + \gamma$$
$$\leq \varepsilon(h^*) + 2\gamma \quad \text{by } (1)$$

→ here $\hat{h}$ is the hypothesys chosen by ERM. [i.e. ∞ the hypothesys, which gives min. error].

→ $\hat{\varepsilon}(h)$ = training error of h

→ $\varepsilon(h)$ = generalization error of h.

* Theorem:- Let given finite set of k hypothesys $|H| = k$, and let m, δ be fixed, then with probability of $1-\delta$,

$$\varepsilon(\hat{h}) \leq \underbrace{\left(\min_{h \in H} \varepsilon(h)\right)}_{\varepsilon(h^*)} + 2\sqrt{\frac{1}{2m}\log\frac{2k}{\delta}} \quad \hookrightarrow (*)$$

Set $\gamma = \sqrt{\frac{1}{2m}\log\frac{2k}{\delta}}$, we know

$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for $\forall h \in H$ holds with probability $1-\delta$ which implies (*)

→ We've proved this for linear hypothesys class H. but if we switch from linear to quadratic hypothesys (to more complex hypothesys class), generalization error $\epsilon(h^*)$ will always less then or equal to linear class.

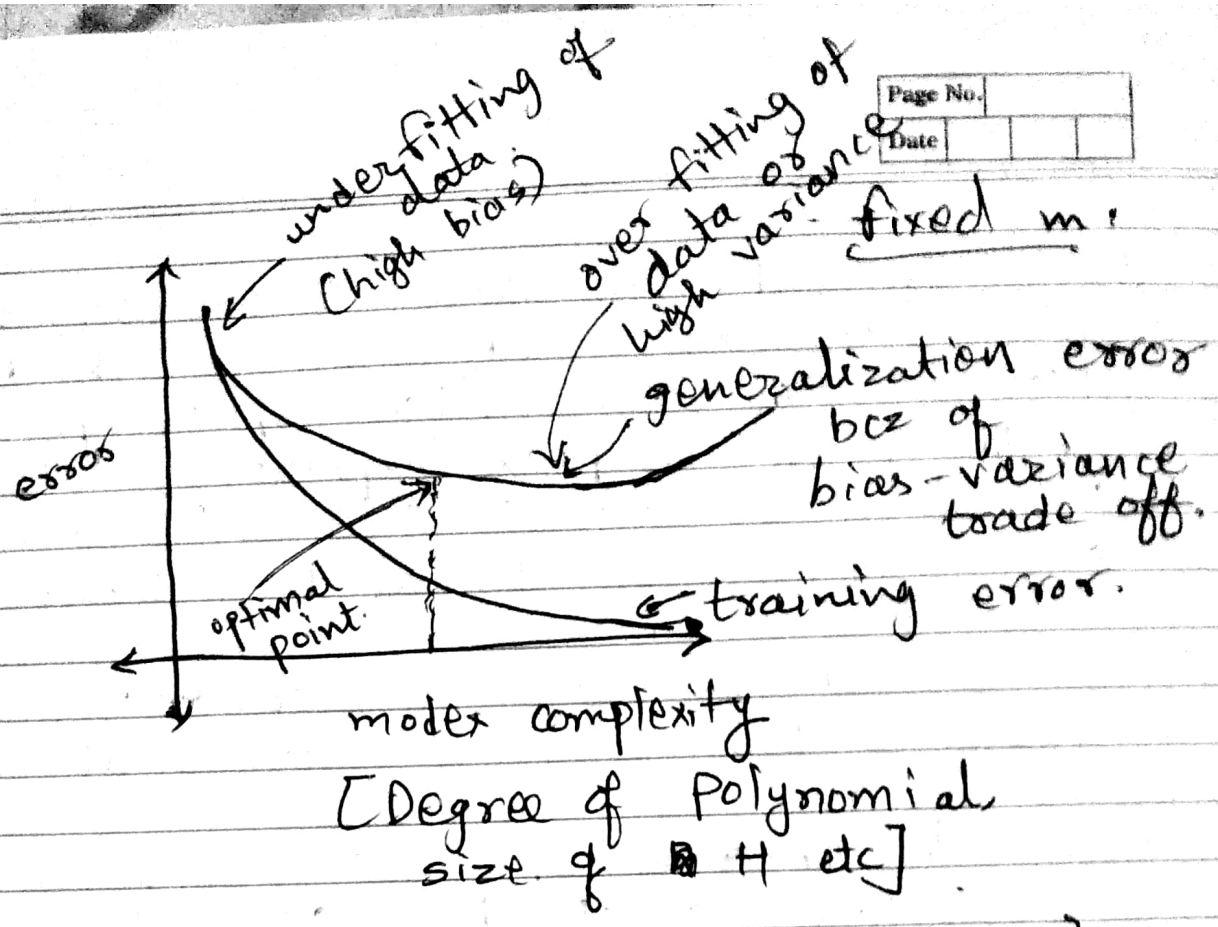
→ So, switching to more complex class decreases the error in generalization

→ But k increases. so generalization error inc. with k, as shown in eq^n (*)

→ This is called bias - variance trade off. It is In the hope of ~~choosing~~ min. generalizing error we tend to choose more complex hypothesis class, but doing so increases value of k, which intern increases the gen. error.

$$\epsilon(\hat{h}) \leq \underbrace{\left(\min_{h \in H} \epsilon(h)\right)}_{\text{"bias"}} + \underbrace{2\sqrt{\frac{1}{2m}\log\frac{2k}{\delta}}}_{\text{"variance"}}$$

→ So more complex class → bias ↓
variance ↑



[Degree of polynomial, size of H etc]

** Corollary :- Let $|H| = k$, Let any δ, γ be fixed, Then for

$$\varepsilon(\hat{h}) \leq \min_{h \in H} \varepsilon(h) + 2\gamma$$

with probability $1-\delta$, if suffices that

$$m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta} = O\left(\frac{1}{\gamma^2} \log \frac{k}{\delta}\right)$$

Lecture-10

Page No.
Date 3 6 18

→ No we need to generalize the previous corollary for ∞ hypothesis class.

→ Let's say H is parameterised by d real numbers.

[So, if we are applying logistic regression with n features, then d = n+1]

→ Computer uses 0 and 1 bits to represent real numbers. And each real no. is represented by 64 bit representation.

→ So H is parameterised by 64d bits.

$|H| = 2^{64d}$ [no. of ways to flip 64d bits]

→ Suffices that

$$m \geq O\left(\frac{1}{\gamma^2} \log \frac{2^{64d}}{\delta}\right)$$

$$\geq O\left(\frac{d}{\gamma^2} \log \frac{1}{\delta}\right)$$

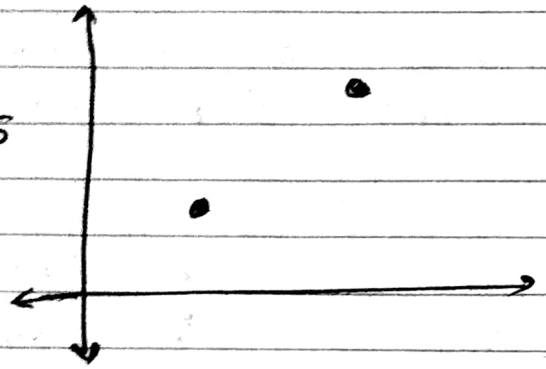
⇒ Definition:- Given a set $S = \{x^{(1)}, x^{(2)}, \ldots, x^{(d)}\}$, we say H shatters S, if H can realize any labeling on it.

→ "H shatters S" means H has some hypothesis 'h', which labels that 'd' examples

correctly.

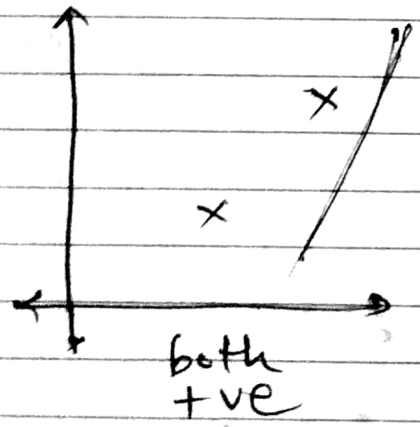
→ For Example :-

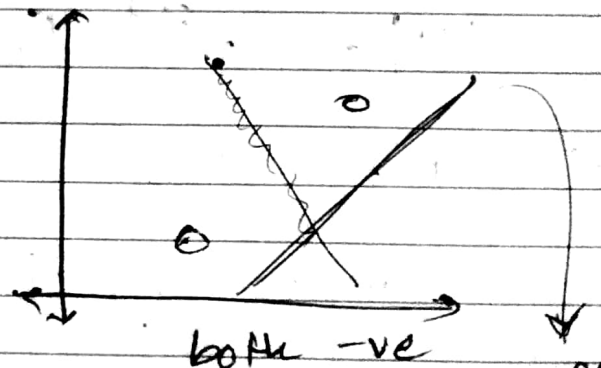
H = { linear classifiers in 2D }

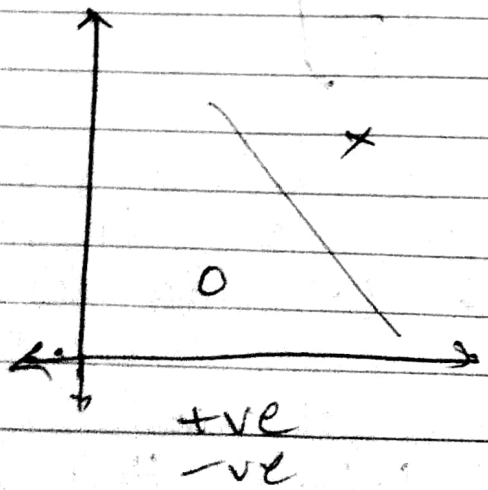


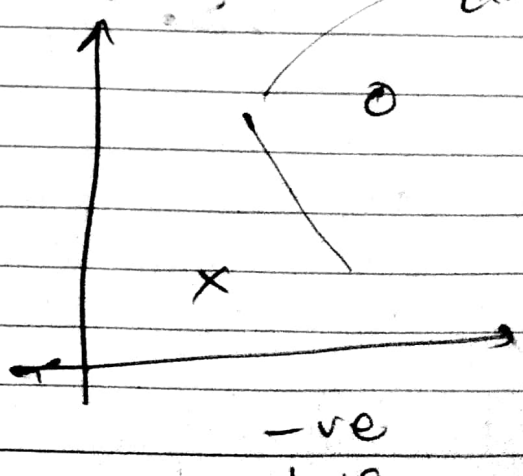
→ set S contains 2 points as shown.

→ Then there are 4 possible ways to label them.



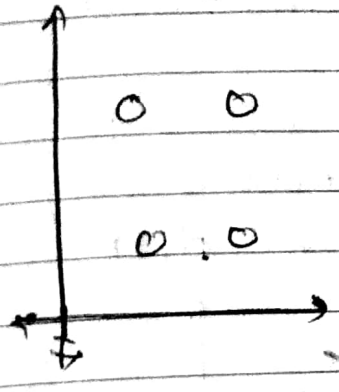






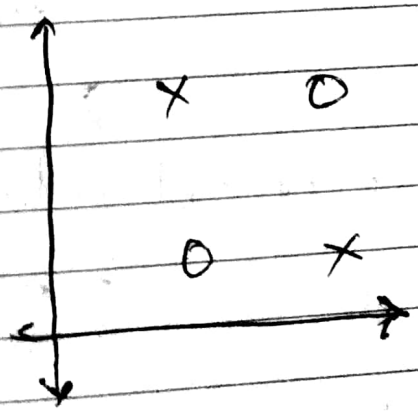
→ So, this show H class shatters the set S of 2 points. (d=2).

→ So for another example, if we have following S,



d=4,

→ then H of linear hypothesis does not shatter S, because there is no linear boundary for some set ∈ S.



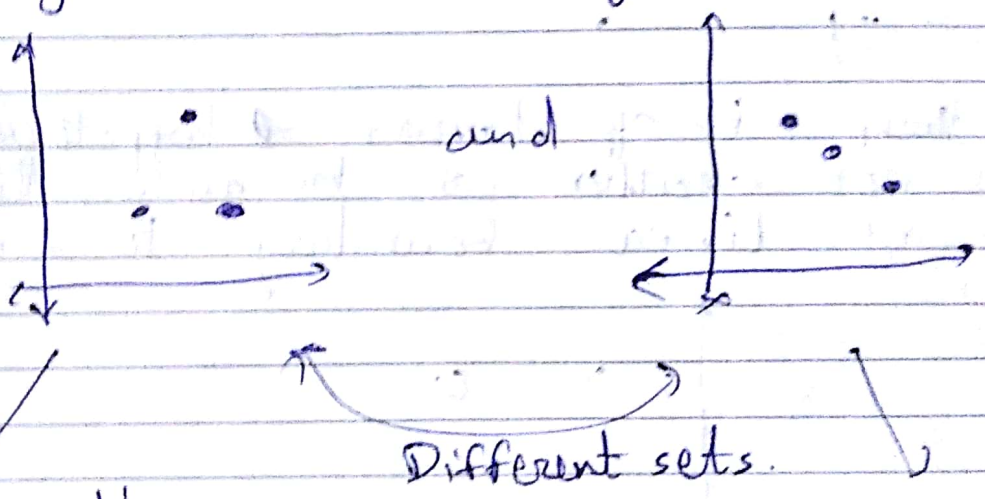
* Definition:- The Vapnik and Chervonenkis dimension (VC dimension) of H (VC(H)) is the size of largest set S, which is shattered by H.

→ for ex.
$H = \{$ Linear classifiers in 2D $\}$, then

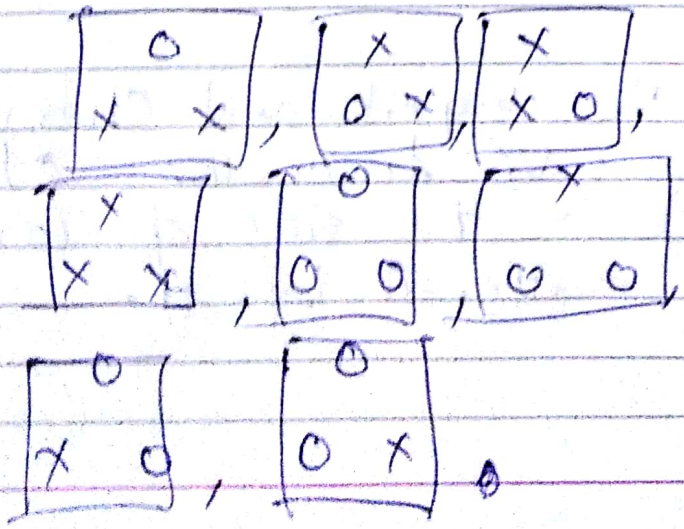
$VC(H) = 3$. → because, we just see that there is no H which shatters S with 4 points in 2D.

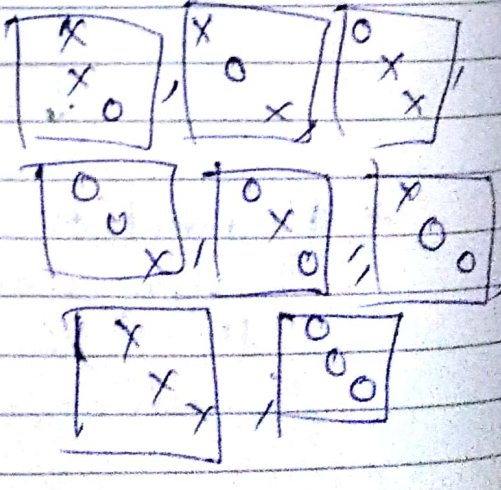
6/6/18
Note

⇒ Shattering means completly classifying any one set. (Set here means any one arrangements). So ex. of different sets is



corresponding possible classifications.





Scanned by CamScanner

→ More generally in n-dimension,

$VC(\{\text{linear classifier in } n\text{-d}\}) = n+1$

→ the best result in learning theory :-

* **Theorem** :- Let H be given and let $VC(H) = d$, then with probability $1-\delta$, we've that $\forall h \in H$,

$$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \cdot 0\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right)$$

↳ condition of uniform convergence

→ Thus, with prob. $1-\delta$, we also have

$$\hat{\varepsilon}(\hat{h}) \leq \varepsilon(h^*) + O\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right)$$

we've already proved this for 2-D this is for n-dimensions.

Page No.
Date

* Corollary:- To guarantee $\epsilon(\hat{h}) \leq \epsilon(h^*) + 2\gamma$, with probability $1-\delta$, it suffices that

$$m = O_{\gamma,\delta}(d)$$

→ Assumed constants while writing Big-O notation.

intuition behind this is that, to achieve uniform convergence, we require the number of training examples, is linearly proportional to the VC dimension of the hypothesis class H.

→ And it turns out for most reasonable hypothesis classes, the VC dimension is very similar to the no. of parameters.

→ for ex. logistic regression in n-dimensions has n+1 parameters, and also VC dimension is n+1 [As mentioned previously for linear classifiers].

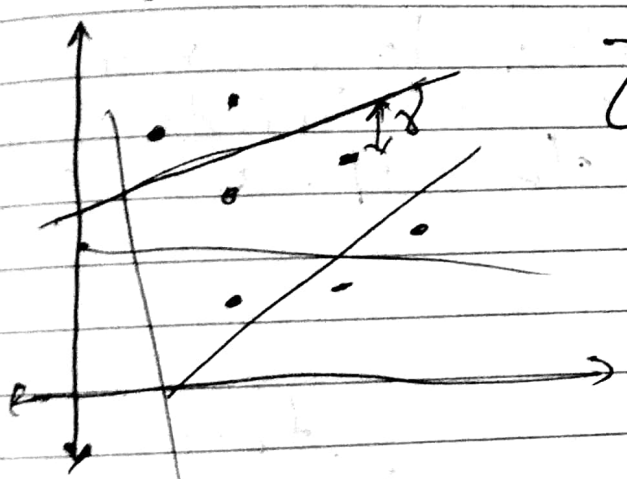
→ So, no. of training ex. required is roughly linear to the no. of parameters of model.

[Not true for few strange examples]

* How SVM don't overfit ?

→ In SVMs, using kernels, we map the features to infinite dimension feature space, so it seems like VC(H) = ∞.

→ But it terns out that the class of linear separators has with large margin has low VC dimension.

→ 

→ Here all the classifiers (hyperplanes) and every data points has at least γ margin. [less then γ margin is not allowed],

→ If data points lie in withing some sphere R,

$$\|x^{(i)}\| \leq R$$

→ and considering only the linear separators, which separate the data with margin of at least γ, then

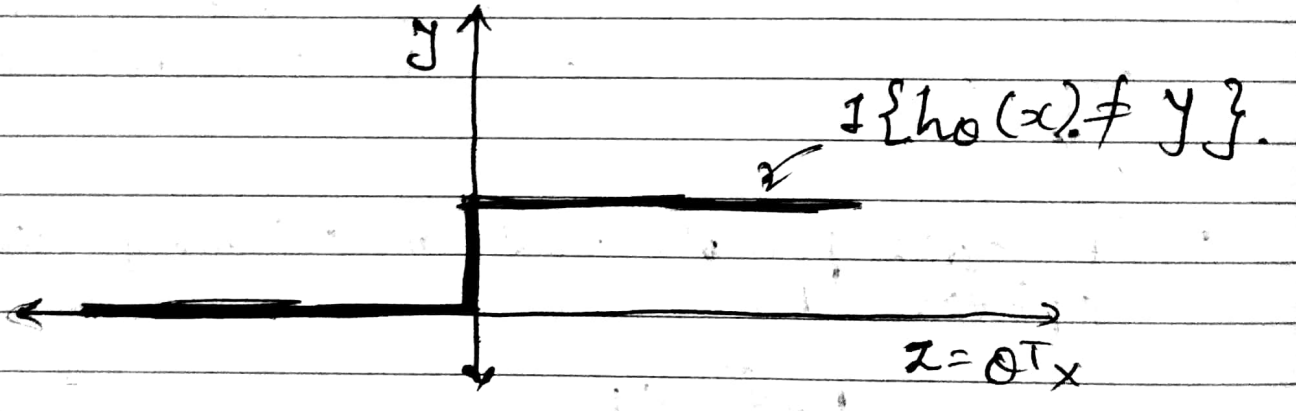
$$VC(H) \leq \left\lceil \frac{R^2}{4\gamma^2} \right\rceil + 1$$

→ This is a bound of VC dimensions, which has no dependence on dimension of H.

→ So, even if the data points are in ∞ dimensions, if we have some class H, for which margin is high, then VC dimension can be low.

→ So, in toying to find large margin, SVM automatically finds lower VC dimensions class H and so it does not overfit.

→

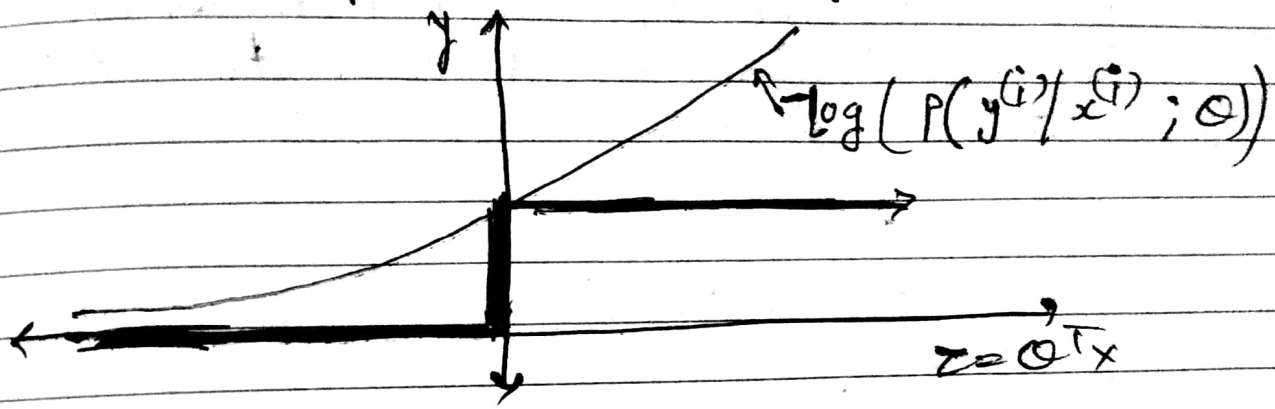


$$h_\theta(x) = g(\theta^T x)$$

→ So, here we want to choose params. θ to minimize the above step function [we want $h_\theta(x) \approx y$, which is on -ve X-axis, and that's why we want to minimize it].

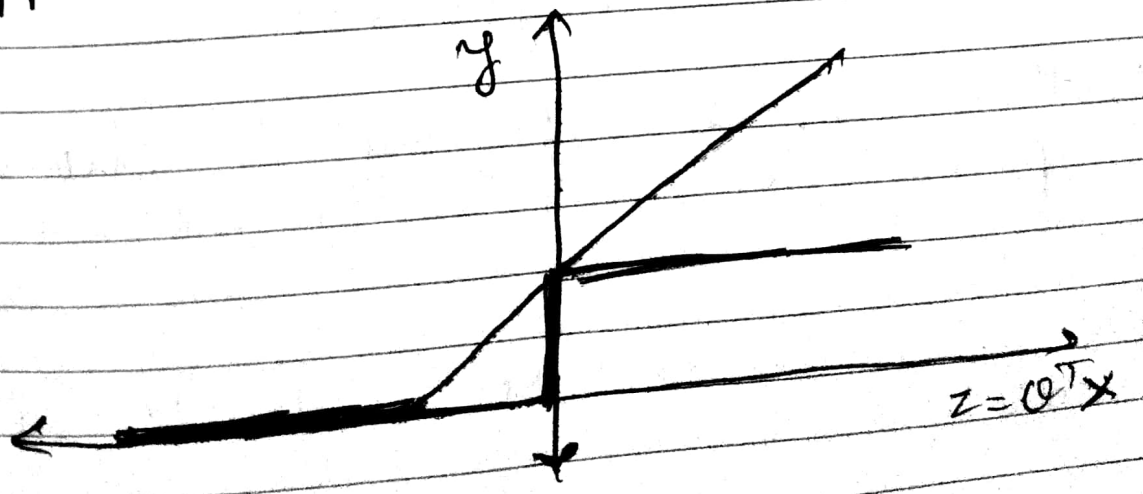
→ But this step function is non-convex function, so non-convex optimization.

→ Both Logistic Regression and SVMs can be viewed as using a rough approximation of this problem.



→ Logistic Regression trying to maximize likelyhood [or log likelyhood], and so minimize $-\log$ likelihood.

→ Also SVM tries to minimize this step function but with different approx. then logistic regression



Page No.
Date 6 6 18

* Model Selection :-

→ It provides class of methods, to automatically make the trade off between bias and variance.

→ Let's say we are deciding what degree of polynomial to choose to best fit any model.

→ $\theta_0 + \theta_1 x$.

→ $\theta_0 + \theta_1 x_1 + \theta_2 x^2$

⋮

→ $\theta_0 + \theta_1 x^1 + \cdots + \theta_{10} x^{10}$.

→ Another ex. of model selection is to choose

τ = bandwidth parameter in locally weighted linear regression.

→ Another ex, is trying to choose param c, in soft margin SVM.

→ Let's say we've some finite set of models,

$u = \{M_1, M_2, M_3 \ldots\}$

may be $\theta_0 + \theta_1 x$ (linear) classifier

$\theta_0 + \theta_1 x + \theta_2 x^2$ ... (quadratic) classifier

~~One terrible idea is to train the model using each~~

→ One terrible idea is to train all the models above and see which model gives min. error.

→ One method to select the perfect model is Hold-out cross validation.

→ In this we take a training set (all data) and randomly split training set (S) into two subsets

S_{train} (70%) and

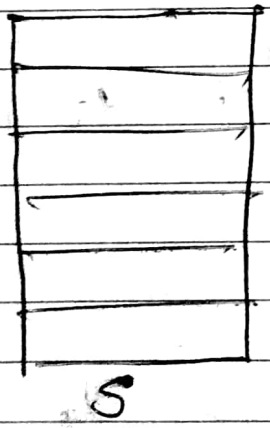
$S_{cross-validation}$ (30%)

→ Now, train each model on only S_{train} and test it on S_{cv}. and pick model with lowest error of S_{cv}.

→ This method works well, but in some applications, there are very small amount of training sets or data available, so holding the 30% of total data just for model selection is more costly in such areas.

* K-fold cross Validation:-

→ In this method, take the whole data set s, divide it in k pieces and then train all the models on k-1 pieces and then test over remaining 1 piece. And the take average of error over the k results.



K = 6

→ So for each model, do the above process and take average. Select the model with lowest average.

→ k=10 is common choise.

→ Advantage is we only hold 1/k th of data instead of 30% of data for testing.

→ But disadvantage is it requires more computation power. Bcz each model needed to be trained k times.

→ If we take k=m, where
m = no. of examples (total data set), then it is called leave one out cross validation.

→ So it works like take out the 1st example and train on rest of the examples and then test on 1st one and so on.... [very expensive in terms of ~~Compution~~ computation).

→ So only used when we've too low training examples. like only 15 training examples for ex.

9/6/18

* Feature selection:-

→ In many ML problem, features are very high or very high dimensional feature space.

→ For ex. in text classification (spam/not-spam), no. of features can easily be as large as size of dictionary [50,000 or more].

→ So, with such high features, there may be a risk of overfitting.

→ If we reduce the no. of features, may be we can reduce the variants of learning algo. and in turn risk of overfitting can be reduced.

→ So, in text classification, there are many such features, which are not relevant to the problem, for ex. if there are words like "and", "is", "not" etc don't tell anything about spam or not spam. But there is also a small set of words which are relevant.

→ So, in feature selection, we only select a subset of features, which is relevant to specific problem.

→ With n features, there are 2^n possible subsets. [include the feature or not].

→ In feature selection, we use different heuristics to search in the possible feature space of 2^n features.

* One simple algo. for this is forward search or selection.

→ start with $F = \phi$

Repeat {

1) For $i = 1 \ldots n$
   try adding feature i to F, and evaluate using cross-validation.
2) set $F = F \cup \{$best feature found in step (1)$\}$

}

→ Output the best hypotheses found.

* "Wrapper" feature selection

→ This is also called wrapper feature selection. Because it is wrapper over learning algo. bcz it calls learning algo. from it.

* Backward search :-

→ start with $F = \{1, 2, \ldots n\}$ (all the features) and delete one feature at a time.

→ Both this algo. does not guarantee of finding best feature. It is NP-hard problem. But both algo. works fine in practice.

→ Forward selection is computationally very intense.

* "Filter" Method:-

→ For each feature i, compute some measure of how informative x_i is about y.

E.g. $corr(x_i, y)$

$$MI(x_i, y) = \sum_{x \in \{0,1\}} \sum_{y \in \{0,1\}} P(x_i, y) \log \frac{P(x_i, y)}{P(x_i) P(y)}$$

MI → Mutual Information

Estimate from training Data

$$= KL(P(x, y) \,||\, P(x) P(y))$$

KL → Kullback-Leibler divergence

KL divergence :- shows how diff. two probability distribution are.

Page No.
Date

→ So, if KL divergence is zero,
then $P(x, y) = P(x)\,P(y)$
that means x and y are independent and they are less mutual informative.

→ x and y are ~~not~~ non-independent then it's assumed that x can tell something about y or vice-versa.
So, they are highly mutual informative.

→ Having chosen above ~~measure~~, measure, pick top k features, with largest $Corr(x_i, y)$ or $MI(x_i, y)$.

→ And for choosing k, we can use Cross-Validation.

Page No.
Date 12 | 6 | 12

Lecture - 11

* Bayesian statistics and Regularization :-

→ In previous lecture learning theory, we saw that those theories tries to simplify the model by selecting simple models.

→ For ex. in feature selection, it was about to reduce the no. of features and hence simplify the model.

→ Regularization is another way to prevent overfitting and it also keep model complexity same as previously.

→ In linear Regression,

maximum likelihood

$$\max_{\theta} \prod_{i} p(y^{(i)} / x^{(i)}; \theta)$$

→ This is called frequentist procedure.

→ The alternative of this method of estimating is Bayesian statistics.

→ $P(\theta)$ - prior

Eg. prior on θ can be gaussian distribution as

$$\theta \sim \mathcal{N}(0, \tau^2 I)$$

→ let data set is,

$$S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$$

→ then in Bayesian statistics, we'll calculate posterior probability of θ given S: $P(\theta/S)$

→ $P(\theta/S) \propto \left(\prod_{i=1}^{m} P(y^{(i)}/x^{(i)}; \theta)\right) P(\theta)$

↑posterior

→ To make a new prediction on new input x,

$$P(y/x, S) = \int_{\theta} P(y/x, \theta) P(\theta/S) d\theta$$

$$E[y/x, S] = \int_{y} y P(y/x, S) dy$$

↓ here θ is considered as random variable, [see it's now separated by , comma instead of ;].

MAP = Maximum A posteriori estimation.

Page No.	
Date	

→ The above integral is commonly hard to calculate when θ is very high dimensional.

→ so, instead of computing above, we will do following.

$$\theta_{MAP} = \arg\max_{\theta} P(\theta/S)$$

$$= \arg\max_{\theta} \left(\prod_{i=1}^{m} P(y^{(i)}/x^{(i)};\theta) \right) P(\theta).$$

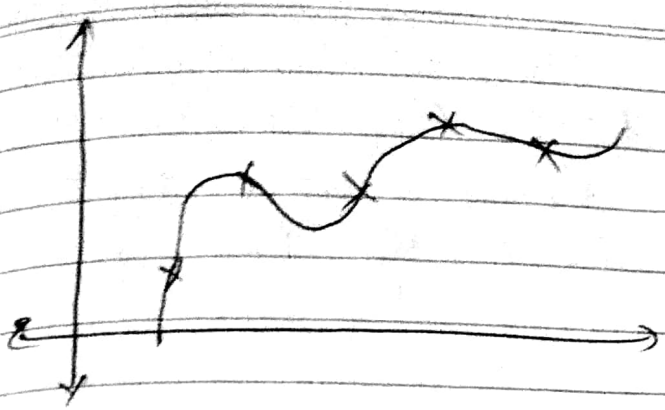
→ And to make prediction,

predict $h_{\hat{\theta}_{MAP}}(x) = \hat{\theta}_{MAP}^{T} x$

→ One intuition is that if our θ is guassian distributed,

$\theta \sim \mathcal{N}(0, \tau^2 I)$, then it's probability mass is concentrated near zero, so it is same as setting feature values to zero and eliminate them, as done by feature selection previously.

→ So, in max. likelihood procedure, if we fit higher order polynomials to small data sets, it will overfit.



→ But in Bayesian regularization, the above curve becomes smoother as we decrease τ. So even we select higher order polynomial [complex model] it still doesn't overfit.

→ Another intuition is that for ex. in linear regression, max. likelihood minimize the following cost function.

$$\min_{\theta} \sum_i \| y^{(i)} - \theta^T x^{(i)} \|^2$$

→ But this bayesian minimizes

$$\min_{\theta} \sum_i \| y^{(i)} - \theta^T x^{(i)} \|^2 + \lambda \|\theta\|^2$$

→ So, we add panelty for being θ large, so it tries to predict θ as small as possible and also minimize the cost.

→ This type of regularization is used generally in text classification, where we have very high feature vector and low training examples.

* ~~Online Learning:-~~

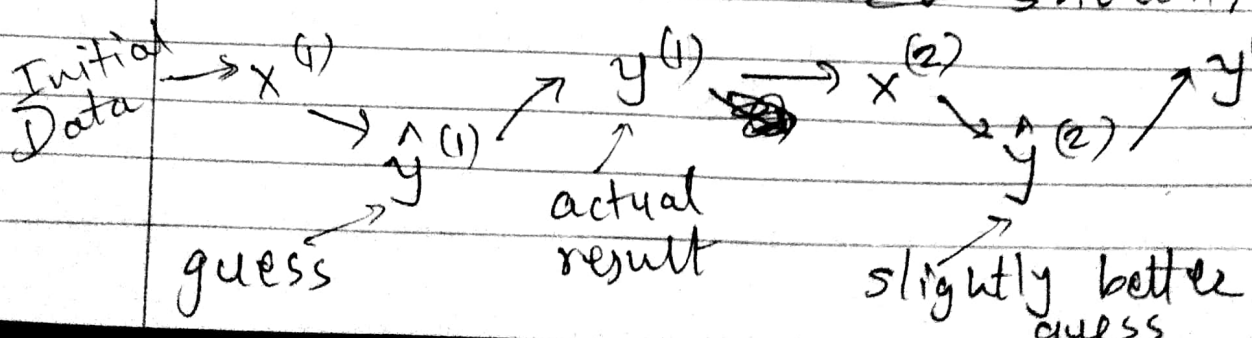
* Online Learning:-

→ So far we've seen batch learning algorithms, where we've given a training set to train the model and another set of data known as test set for testing or evaluating the model.

→ But in online learning, we make predictions, even while we are learning (training the model).

→ Let's take example of classification problem.

→ Initially, without seeing any data we'll first guess the result [just blind guess], then the original result will be shown,



→ Here we care about

Total online error:-

$$\sum_{i=1}^{m} 1\{\hat{y}^{(i)} \neq y^{(i)}\}$$, which is total no. of wrong guess made.

→ One good example, where online learning algo. gives good result is perceptron learning algo.

→ Initialize $\theta = 0$

After i^{th} example, update

$$\theta = \theta + \alpha (y^{(i)} - h_\theta(x^{(i)})) x^{(i)}$$

Page No.
Date 16 6 18

Lecture-12

* Unsupervised learning:-

→ Given a dataset without any labels and from this, it's job of learning algo. to find out patterns or structure in the given data set.

- for example clustering algorithm. which is used in clustering group of customers for business, in biology for clustering genes.

→ "news.google.com" uses such clustering algo. to catagories news in different catagories.

→ Image segmentation is also an ex. of clustering algo.

* K-means clustering Algo:-

Input will be unlabeled data set

$$\{x^{(1)}, x^{(2)}, x^{(3)} \cdots x^{(m)}\}. \in R^n$$

Note that there is no class labels y in the datasets because this is input to unsupervised learning.

→ 1st step:- Initialize cluster centroids.

$\mu_1, \mu_2 \ldots \mu_k \in \mathbb{R}^n$.

2) Then repeat until convergence:-

i) set $c^{(i)} = \arg\min_j \|x^{(i)} - \mu_j\|$

ii) $\mu_j = \dfrac{\sum_{i=1}^{m} 1\{c^{(i)} = j\}\, x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$

→ Refer the animation shown in video for perfect understanding.

* Does this algo. converge?

→ Answer is yes.

→ If we define Distortion function

$$J(c, \mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$x^{(i)}$: data point i

$\mu_{c^{(i)}}$: centroid corresponding to data point i

→ Then K-means is coordinate ascent on $J(c, \mu)$.

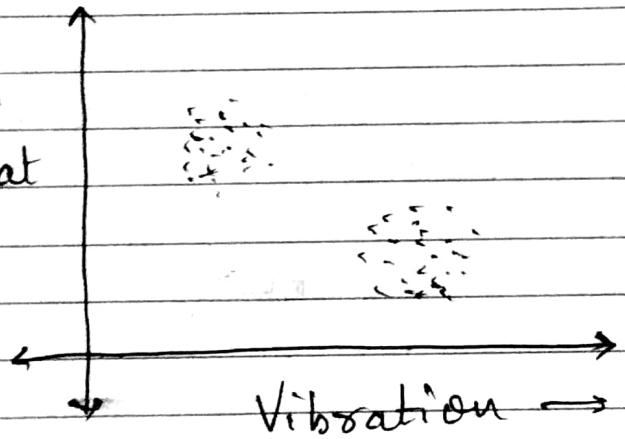
Page No.
Date

* How to choose no. of clusters?

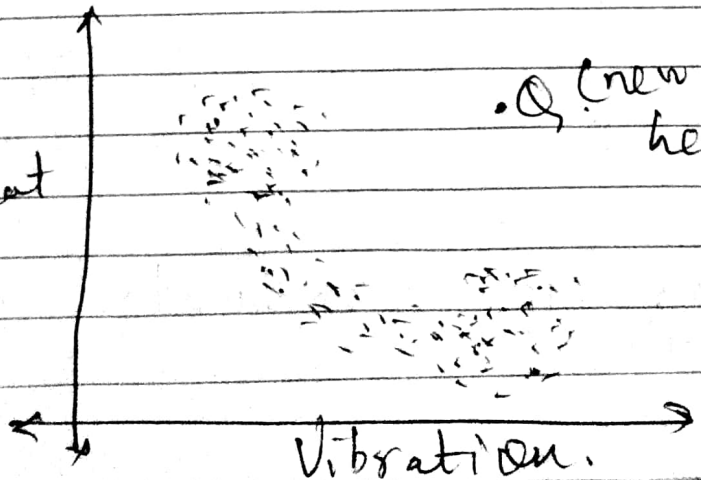
→ It's try-error approach Pick any suitable number & and then try the algorithm and pick the no. of clusters for which algo. gives best result.

* Density Estimation:-

→ Let's say, in an air-craft engine producing company we've following data for different engines.



→ Then we can apply clustering algo. to find 2 clusters in above data set. But if we've following data,



→ And from this we would like to find joint distribution of amount of heat produced corresponding to amount of vibration.

→ Because for any new engine, we would like to decide that it needs further inspection or not.

→ So, can we build a f^n P(x) from the above data from which we can say for every new engine in assembly line, whether it needs inspection or not.

→ This problem is called anomaly detection.

→ And for such problem, take dataset

$\{x^{(1)}, x^{(2)} \dots x^{(m)}\}$

and build the model

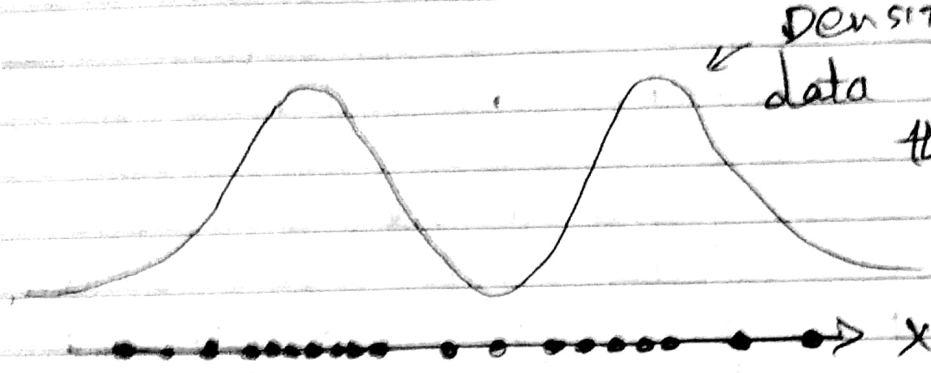
$P(x)$

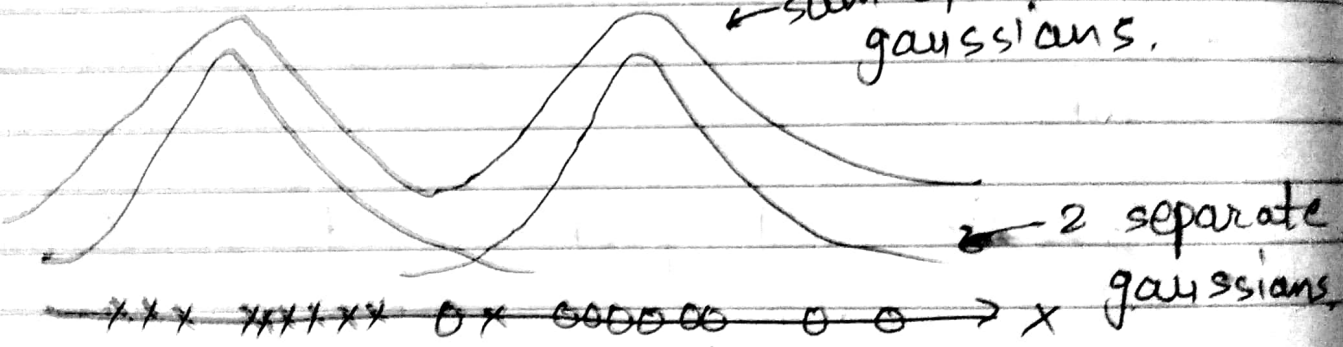
→ If $P(x)$ is very low, then we can say that it is an anomalous example.

→ $P(x)$ here is any distribution function, but it is not like any of the standard distribution functions like gaussian etc.

→ So, we want to come up with a model, which can be used to measure such densities...

Ex. In 1 D, let's our dataset looks like following.





→ But the problem with our data set is we don't know the actual labels of data.

→ So, now we want to come up with an algo. to fix the mixture of such gaussians even when we don't have labeled data.

→ There's a latent (hidden/unobserved) random variable z. And $x^{(i)}, z^{(i)}$ have a joint distribution:-

$$P(x^{(i)}, z^{(i)}) = P(x^{(i)} | z^{(i)}) \, P(z^{(i)})$$

$z \sim \text{Multinomial}(\phi)$ [Bernolli for z Gaussians].

$(\phi_j \geq 0, \sum_j \phi_j = 1)$

&, $P(x^{(i)} | z^{(i)} = j) \sim \mathcal{N}(\mu_j, \Sigma_j)$

→ Same as Gaussian ~~Area~~ Discriminant analysis, the only diff. is that the labels variable y is now replaced with latent z

→ Now, to link this to GDA, let's assume we know z_i's.

→ Then we could write max. $\overset{\log}{\wedge}$ likelihood as,

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}, z^{(i)}; \phi, \mu, \Sigma)$$

$$\phi_j = \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\}.$$

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}, \text{ etc.}$$

Page No.
Date

* EM (Expectation Maximization) Algo.

Repeat {

E-step: Guess values of z_i's

Let $w_j^{(i)} = P(z^{(i)} = j \mid x^{(i)}, \phi, \mu, \Sigma)$

This is probability that data point $x^{(i)}$ come from Gaussian number j. (z_j).

$$= \frac{P(x^{(i)} \mid z^{(i)} = j) P(z^{(i)} = j)}{\sum_{l=1}^{k} P(x^{(i)} \mid z^{(i)} = l) P(z^{(i)} = l)}$$

from bayes theorem.

$$* \quad w_j = \frac{\frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp\left((x^{(i)} - \mu_j)^T \Sigma^{-1} (x^{(i)} - \mu_j)\right) \cdot \phi_j}{\sum_{l=1}^{k} \frac{1}{(2\pi)^{d/2} |\Sigma_l|^{1/2}} \exp\left((x^{(i)} - \mu_l)^T \Sigma^{-1} (x^{(i)} - \mu_l)\right) \cdot \phi_l}$$

M step:- (Maximization step) :-

$$\phi_j = \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}$$

$$\mu_j = \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}$$

$$\Sigma_j = \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

}

→ Note that the formulas of ϕ_j and μ_j are similar to GDA. Also in GDA we were using same covariance matrix for all gaussians, but here it is different for each gaussian.

→ This was special case of more general EM algorithm. For deriving General form of EM, following knowledge is required.

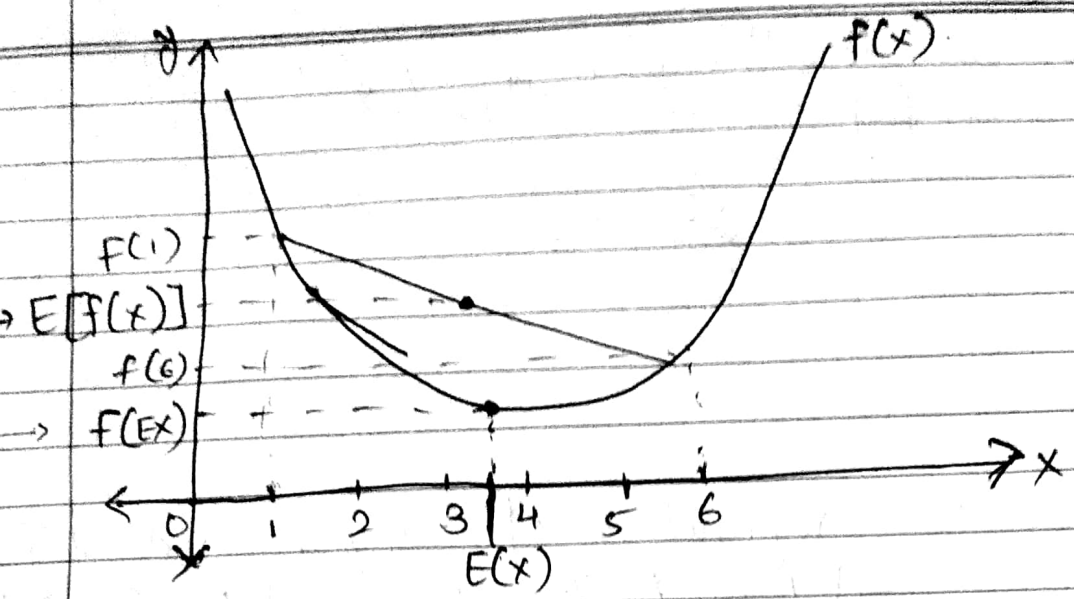
* Jensen's inequality :-

→ Let f be a convex function, (e.g. $f''(x) \geq 0$) and let X be a random variable. Then,

$$f(E[X]) \leq E[f(x)]$$

↑ expectation of X.

↑ expectation of $f(x)$.



Expected value of X = avg of all x

$$= \frac{1+2+\cdots+6}{6}$$

$$= 3.5$$

→ and same can be done for $E[f(x)]$ and the result can be proved.

→ further, if $f''(x) > 0$ (Strictly convex), then,

$$E[f(x)] = f(E[x]) \iff X = E(x) \text{ with prob. 1.}$$

(this means when X is constant)

→ If $f'' \leq 0$ [non convex function], then,

$$f(EX) \geq E[f(x)]$$

Generalized EM :-

→ We've some model for $P(x, z; \theta)$, we observe only x.

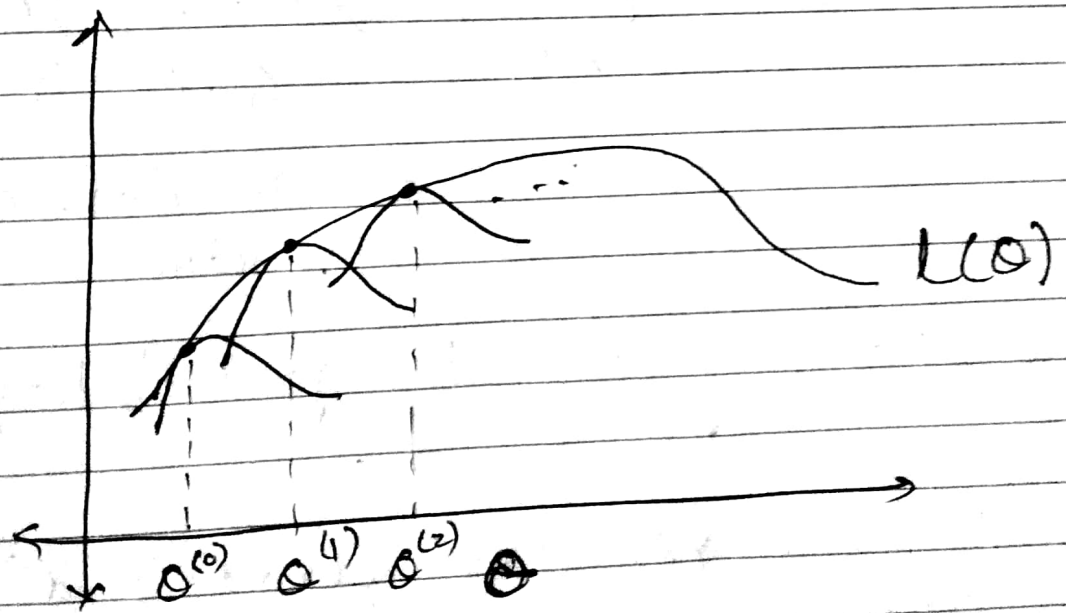
→ Our goal is to maximize

$$l(\theta) = \sum_{i=1}^{m} \log P(x^{(i)}; \theta)$$

log likelihood of param. of model.

$$= \sum_{i=1}^{m} \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta)$$

→ What this algo. does is, initially starts with $\theta^{(0)}$ and constructs a lower bound to $l(\theta)$ and then maximizes it.



→ we want to maximize $l(\theta)$. so,

$$\max_{\theta} l(\theta) = \max_{\theta} \sum_i \log P(x^{(i)}; \theta)$$

$$= \sum_i \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta)$$

$$= \sum_i \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

Q_i is probability distribution function, such that,

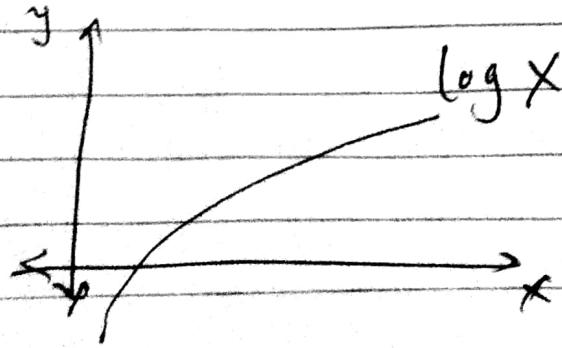
$$Q_i(z^{(i)}) \geq 0, \text{ and } \sum_{z^{(i)}} Q_i(z^{(i)}) = 1$$

$$= \sum_i \log E_{z^{(i)} \sim Q_i} \left[\frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} \right]$$

[Refer to def. of Expection]

[$Q_i(z^{(i)})$ is also the probability of $z^{(i)}$].

→ $\log x$ is concave [non-convex] fn.



→ so, from Jensen's inequality,

$$\log(E[x]) \geq E[\log x]$$

→ so,

$$\sum_i \log E_{z^{(i)} \sim Q_i} \left[\frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} \right] \geq$$

$$\sum_i E\left[\log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} \right]$$

$$= \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

→ $z \sim P$ and a function $g(z)$, the

$$E[g(z)] = \sum_i P(z) g(z)$$

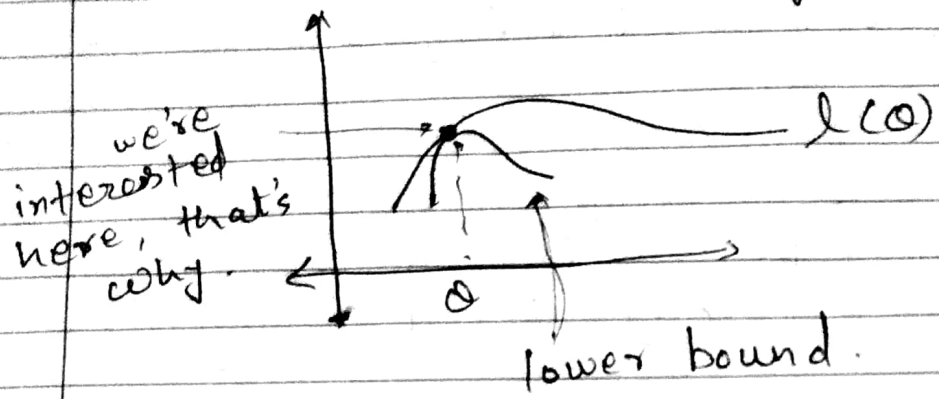
→ we have derived the lower bound of $l(\theta)$, so,

$$l(\theta) \geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

→ Now, to make above inequality to be equality, we need to choose such distribution Q_i.

Page No.
Date

Why we need equality?
→ Refer to the following intuition.



→ So, from, Jensen's inequality, for equality cond^n,

$$\frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} = \text{constant.}$$

(some value for all values of $z^{(i)}$)

so, $Q_i(z^{(i)}) \propto P(x^{(i)}, z^{(i)}; \theta)$

and Q_i is probability distribution,

so,

$$\sum_{z^{(i)}} Q(z^{(i)}) = 1$$

→ $$Q_i(z^{(i)}) = \frac{P(x^{(i)}, z^{(i)}; \theta)}{\sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta)}$$

Page No.
Date

$$= \frac{P(x^{(i)}, z^{(i)}; \theta)}{P(x^{(i)}; \theta)}$$

$$Q_i(z^{(i)}) = P(z^{(i)} | x^{(i)}; \theta).$$

→ So, EM algo given by :-

Repeat {

E-step :-

choose / set $Q_i(z^{(i)}) = P(z^{(i)} | x^{(i)}; \theta)$

M-step :-

$$\theta = \arg\max_{\theta} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}.$$

}