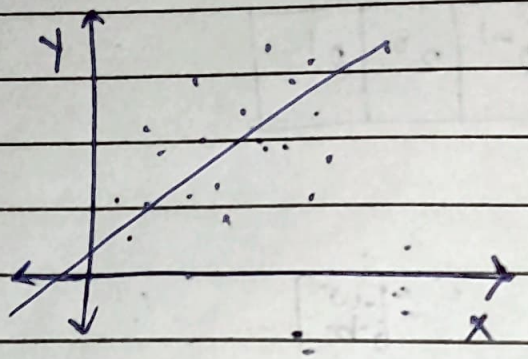
23/2/23

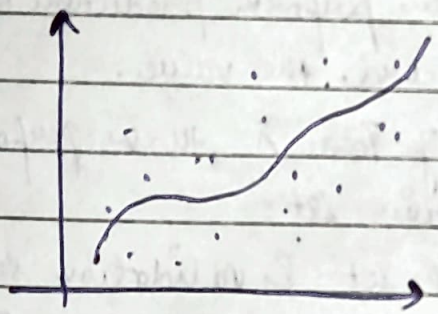
$f(x) = w^T x + b \quad , x \in R^n, w \in R^n, b \in R.$



(linear)

$f(x) = w^T \phi(x) + b, \quad w \in R^m$

$$\phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_m(x) \end{bmatrix} \quad m \to \infty$$

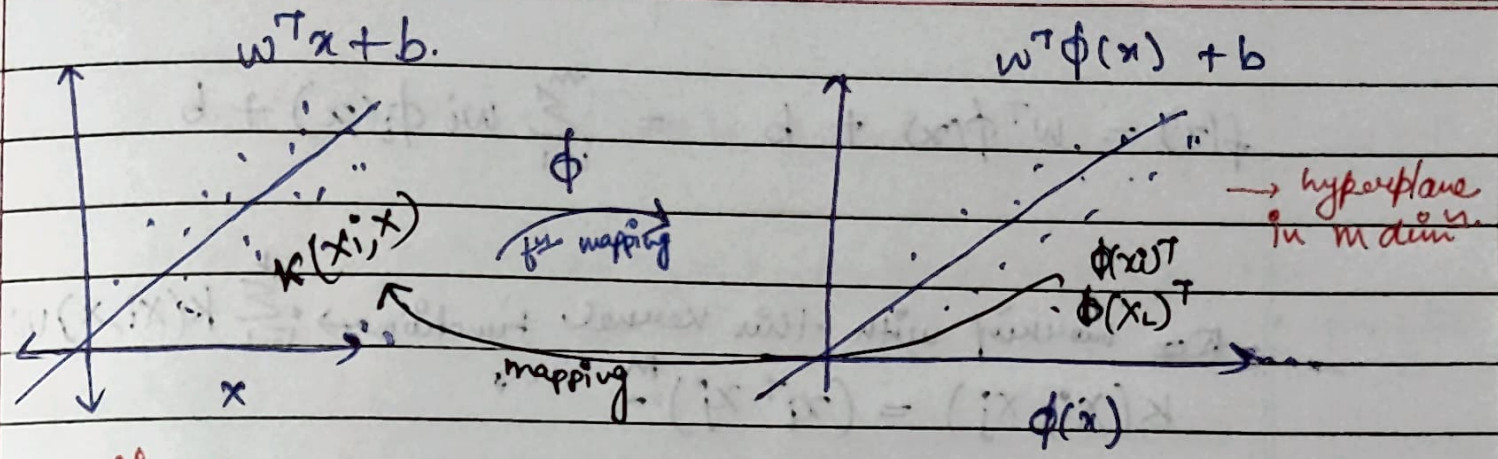


(non-linear)

* mapping x to $\phi(x)$ when we work with non-linear ϕ.
So, at last, we are working with linear function $(w^T \phi(x) + b)$.

$w^T \phi(x) + b$ is a line in m-dimension for $\phi(x) = \phi_1(x), \ldots \phi_m(x)$.

* jitne no. of pts in $\phi(x)$, utne ka dimension hoga.



Linear (input space)

(feature space)

$T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L)\}$

$x_i \in \mathbb{R}^n$

$y_i \in \mathbb{R}$

$W = \sum_{i=1}^{L} v_i \phi(x_i)$

$= v_1 \phi(x_1) + v_2 \phi(x_2) + \cdots + v_L \phi(x_L)$

* In higher dimension space, w can be obtained by linear comb^n of L training pts. in feature space.

$\rightarrow w^T \phi(x) + b$

$\Rightarrow \left(\sum_{i=1}^{L} v_i \phi(x_i)\right)^T \phi(x) + b$

$\Rightarrow \sum_{i=1}^{L} v_i \phi(x_i)^T \phi(x) + b$

* Kernel Generated Function

$\sum v_i K(x_i, x) + b$

↳ This will tell in higher dim^n how $\phi(x)$ looks like.

$$f(x) = w^T \phi(x) + b = \sum_{i=1}^{m} w_i \phi_i(x) + b$$

working with this kernel function $\rightarrow \sum_{i=1}^{L} K(x_i, x) v_i + b$

$$K(x_i, x_j) = (x_i^T x_j)^m$$

$$K(x_i, x_j) = (x_i^T x_j)^m$$

* If we are working with Gaussian basis fn & have taken m no. of basis, then we use this type of kernel fn

$$K(x_i, x_j) = \exp \frac{-\|x_i - x_j\|^2}{\sigma^2} = \phi(x_i)^T \phi(x_j)$$

→ Kernel fn. for Gaussian basis fn.

7/2/23

Fitting the linear function

$$f(x) = w_1 x_1 + w_2 x_2 + b. \quad \longrightarrow \text{linear}$$

$$f(x) = w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2 + w_4 x_1 + w_5 x_2 + b$$

kernel function

x_1	x_2
-	-
-	-
-	-

ϕ

x_1	x_2	x_3
-	-	-
-	-	-
-	-	-

$$\phi(x) = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \end{bmatrix}$$

$$K(x,y) = \phi(x)^T \phi(y)$$

Date
Page

$w_1 x_1^2 + w_2 x_2^2 + \sqrt{2} w_3 x_1 x_2 + b$. → linear fn. in 3-D.

$w_1 x_1^2 + w_2 x_2^2 + \sqrt{2} w_3 x_1 x_2 + w_4 x_1 + w_5 x_2 + b$
↳ linear fn. in 5D

$\phi(x)^T \phi(y) = ?$

$$\begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \\ x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2} y_1 y_2 \\ y_1 \\ y_2 \end{bmatrix} = (x_1 y_1)^2 + (x_2 y_2)^T + 2 x_1 x_2 y_1 y_2 + x_1 y_1 + x_2 y_2$$
$$= (x^T y)^2 + (x^T y)$$

$\therefore \phi(x)^T \phi(y) = (x^T y)^2 + (x^T y)$

$$K(x,y) = (x^T y)^2 + x^T y$$ → Kernel Generated fn.

$\sum_{i=1}^{n} K(x_i, x) u_i + b$

	x_1	x_2
x_1	5.9	3
x_2	6.9	3.1
x_3	6.6	2.9
x_4	4.6	3.2
x_5	6	2.2

$\sum_{i=1}^{n} K(x_i, X) u_i + b$

$u_i = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ b \end{bmatrix}$

$$K\left(\begin{bmatrix} 5.9 \\ 3 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \left((5.9 x_1 + 3 x_2)^2 + (5.9 x_1 + 3 x_2) u_1 + b\right.$$

u_i → we need to find. $\begin{bmatrix} w_2 \\ w_1 \\ b \end{bmatrix}$

b → bias term

$$K(x_1, x_1) = K\left(\begin{bmatrix} 5.9 \\ 3 \end{bmatrix}, \begin{bmatrix} 5.9 \\ 3 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 5.9 & 3 \end{bmatrix} \begin{bmatrix} 5.9 \\ 3 \end{bmatrix} \Rightarrow \because K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$$

Date
Page

* For every kernel fn., we have a kernel matrix.

$T = \{(x_1, y_1), (x_2, y_2) \ldots, (x_L, y_L)\}$ $\quad x_i \in R^n$, $y_i \in R$.

Kernel Matrix:

$$K = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_L) \\ K(x_2, x_1) & K(x_2, x_2) & \cdots & K(x_2, x_L) \\ \vdots & \vdots & & \vdots \\ K(x_L, x_1) & K(x_L, x_2) & & K(x_L, x_L) \end{bmatrix}$$

- always +ve semi definite
- symmetric matrix

→ Quadratic Kernel

$$K(X, Y) = (c + X^T Y)^2$$

c → constant (user defined)

$$K(X, Y) = (c + X^T Y)^m \rightarrow \text{for polynomial of order } m.$$

$$K(X, Y) = \exp^{-\frac{\|X - Y\|^2}{2\sigma^2}}$$

$\sigma^2 \rightarrow$ user defined

this kernel fn can be used to learn any type of function in any dimension.

→ Jisme map kare h → feature space.

$$\frac{\lambda}{2} u^T u + \frac{1}{L}\sum_{i=1}^{L}\left(y_i - \left(\sum_{j=1}^{L} K(x_j, x_i)\, u_j + b\right)\right)^2$$

$$\min_{(u,b)}$$

$$\begin{bmatrix} u \\ b \end{bmatrix} = (A^T A + \lambda I)^{-1} A^T Y$$

$$A = \begin{bmatrix} K(x_1,x_1) & K(x_1,x_2) & \cdots & K(x_1,x_L) & 1 \\ K(x_2,1) & K(x_2,x_2) & \cdots & K(x_2,x_L) & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ K(x_L,x_1) & K(x_L,x_2) & \cdots & K(x_L,x_L) & 1 \end{bmatrix}$$

18/23

Given the training set,

$$T = \{(x_i, y_i) : x_i \in \mathbb{R}^n, y_i \in \mathbb{R}\}$$

$f(x) = w^T x + b$

$w \in \mathbb{R}^n$

$b \in \mathbb{R}$.

$= \min_{(w,b)} J(w,b)$

We are solving

$$\min_{w,b} \frac{\lambda}{2} w^T w + \frac{1}{2L}\sum_{i=1}^{L}\left(y_i - (w^T x_i + b)\right)^2$$

- $\frac{d}{dx}(x^T A x) = 2Ax$

$\frac{d}{dw}\left(\frac{\lambda w^T w}{2}\right) = \frac{\lambda \cdot 2w}{2} = \lambda w$

Gradient Descent Method

$$\frac{\partial}{\partial w} = \nabla_w J(w,b) = \lambda w - \frac{2}{L}\sum_{i=1}^{L} x_i\left(y_i - (w^T x_i + b)\right)$$

$$\frac{\partial}{\partial b} = \nabla_b J(w,b) = -\frac{2}{L}\sum_{i=1}^{L}\left(y_i - (w^T x_i + b)\right)$$

$$\begin{bmatrix} w^0 \\ b^0 \end{bmatrix} \in \mathbb{R}^{n+1} \longrightarrow \text{Initialise}$$

Repeat

$$\begin{bmatrix} w^{k+1} \\ b^{k+1} \end{bmatrix} = \begin{bmatrix} w^k \\ b^k \end{bmatrix} - \eta \underbrace{\begin{bmatrix} \nabla_w J(w^k, b^k) \\ \nabla_b J(w^k, b^k) \end{bmatrix}}_{\text{Gradient}}$$

Until $\left\| \begin{bmatrix} \nabla_w J(w^k, b^k) \\ \nabla_b J(w^k, b^k) \end{bmatrix} \right\|_2 \leq \varepsilon$.

* **For any basis function :-**

$$\min_{w,b} \frac{\lambda}{2} w^T w + \sum_{i=1}^{L} \left(y_i - (w^T \phi(x_i) + b) \right)^2 = \min_{(w,b)} J(w,b)$$

$$\nabla_w J(w,b) = \lambda w - \frac{2}{L} \sum_{i=1}^{L} \left(y_i - (w^T \phi(x_i) + b) \right) \phi(x_i)$$

$$\nabla_b J(w,b) = -\frac{2}{L} \sum_{i=1}^{L} \left(y_i - (w^T \phi(x_i) + b) \right)$$

* **For Kernel Generated function :-**

$$\min_{u,b} \frac{\lambda}{2} u^T u + \frac{1}{L} \sum_{i=1}^{L} \left(y_i - \left(\sum_{j=1}^{L} K(x_j, x_i) u_j + b \right) \right)^2$$

$$f(x) = \sum_{j=1}^{L} k(x_j, x) u_j + b$$

$$\nabla_u J(u,b) = \lambda u - \frac{2}{L} \sum_{i=1}^{L} \left(y_i - \left(\sum_{j=1}^{L} K(x_j, x_i) u_j + b \right) \right) K(x_j, x_i)$$

$$\nabla_b J(u,b) = -\frac{2}{L} \sum_{i=1}^{L} \left(y_i - \left(\sum_{j=1}^{L} k(x_j, x_i) u_j + b \right) \right)$$

$W = \begin{bmatrix} w \\ b \end{bmatrix}^0 =$ (101×1)

$K = [\ ]_{100 \times 100}$.

Date _____
Page _____

	x_1	x_2	Y
x_1	0	0	0
x_2	0	1	0
x_3	1	1	0
x_4	1	0	1

$K(u,v) = (u^T v)^2$

↑ kernel fn

~~$K(x_1, x_1) = (x_1^T x_1)^2$~~

write, Gradient Descent Algo.

→ Kernel matrix, $K = \begin{bmatrix} K(x_1,x_1) & K(x_1,x_2) & \cdots & K(x_1,x_L) \\ K(x_2,x_1) & K(x_2,x_2) & & K(x_2,x_L) \\ \vdots & \vdots & & \\ K(x_L,x_1) & K(x_L,x_2) & \cdots & K(x_L,x_L) \end{bmatrix}$

$K = \begin{bmatrix} K(0,0)\,0 & K(0,0)\,0 & K(0,1)\,0 & K(0,1)\,0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{4\times4}$

$x_i = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$ $b = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$

3 $y_1 - \left(\sum_{j=1}^{4} K(x_j^0, x_1)\, v_j^0 + b\right) \sum_{j=1}^{4} K(x_j^0, x_1)$

5 $y_2 - \left(\sum_{j=1}^{4} K(x_j^0, x_2)\, v_j^0 + b\right)$

$0 - \big((K(x_1,x_1)v_1 + K(x_2,x_1)v_2 + K(x_3,x_1)v_3 + K(x_4,x_1)v_4) + b\big)\big(K(x_1,x_1) \cdot K(x_2,x_1) \cdot K(x_3,x_1) \cdot K(x_4,x_1)\big)$

$= 0 - [0+0+0+0+b](0)$

$= 0$

$K(x_1,x_1)^2 \quad (x_2^T x_2)^2$

$\begin{bmatrix} 0 \\ 1 \end{bmatrix}_{2\times1} \begin{bmatrix} 0 & 0 \end{bmatrix}_{1\times2} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}_{2\times1} \begin{bmatrix} 0 & 1 \end{bmatrix}_{1\times2}$

$x_1^T = [0\ \ 0]$

$x_2^T = [0\ \ 1]$

$x_3^T = [1\ \ 1]$

$x_4^T = [1\ \ 0]$

$(x_2^T x_1)^2 = [0\ 1] \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0^2 = 0$

$(x_2^T x_2)^2 = [0\ 1] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1^2 = 1.$

$1\times2 \quad 2\times1$

13/3/23

Least Square Kernel Regression

$$\min_{(u,b)} \frac{\lambda}{2} u^T u + \sum_{i=1}^{L} \left(y_i - \left(\sum_{j=1}^{L} K(x_j, x_i) u_j + b \right) \right)^2$$

$$\begin{bmatrix} u \\ b \end{bmatrix} = (A^T A)^{-1} A^T Y$$

$$A = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_L) & 1 \\ \vdots & & & & \\ K(x_L, x_1) & K(x_L, x_2) & \cdots & K(x_L, x_L) & 1 \end{bmatrix}$$

First let us consider

$$J(u, b, x_k, y_k) = \left(y_k - \left(\sum_{j=1}^{L} K(x_j, x_k) u_j + b \right) \right)^2$$

$$\frac{\partial J_u(u, b, x_k, y_k)}{\partial u} = -2 \left(y_k - \left(\sum_{j=1}^{L} K(x_j, x_k) u_j + b \right) \sum_{j=1}^{L} K(x_j, x_k) \right)$$

Differentiating this term wrt u.

$$\sum_{j=1}^{L} K(x_j, x_k) u_j = K(x_1, x_k) u_1 + K(x_2, x_k) u_2 + \cdots + K(x_L, x_k) u_L + b$$

$$= \begin{bmatrix} K(x_1, x_k) \\ K(x_2, x_k) \\ \vdots \\ K(x_L, x) \end{bmatrix}$$

$$\therefore \frac{\partial J(u, b, x_k, y_k)}{\partial u} = -2 \left(y_k - \left(\sum_{j=1}^{L} K(x_j, x_k) u_j + b \right) \right) \begin{bmatrix} K(x_1, x_k) \\ K(x_2, x_k) \\ \vdots \\ K(x_L, x) \end{bmatrix}$$

now wrt b,

$$\frac{\partial J(u,b,x_k,y_k)}{\partial b} = -2\left(y_k - \left(\sum_{j=1}^{l} K(x_j,x_k)u_j + b\right)\right)$$

Gradient Descent least Square Kernel Regression

Algorithm : Gradient descent method :-

Initialise $u^0 = u^{start} \in R^l$ and $b \in R$

Repeat

$$u^{(j+1)} = u^{(j)} - \eta_k\left(\lambda u + \sum_{i=1}^{l} \frac{\partial J(u,b,x_k,y_k}{\partial u}\right)$$

η_k = step length

$$b^{(j+1)} = b^{(j)} - \eta_k\left(\sum_{i=1}^{l} \frac{\partial J(u,b,x_k,y_k)}{\partial b}\right)$$

Until $\left\| \begin{bmatrix} \lambda u + \sum_{i=1}^{l} \frac{\partial J(u,b,x_k,y_k)}{\partial u} \\ \sum_{i=1}^{l} \frac{\partial J(u,b,x_k,y_k)}{\partial b} \end{bmatrix} \right\| \leq \varepsilon$

Stochastic Gradient Descent Least Square Regression

$$\min_{(w,b)} \frac{\lambda}{2} w^T w + \sum_{i=1}^{l} (y_i - (w^T x_i + b))^2$$

$$\frac{\partial}{\partial w} = \lambda w - 2 \sum_{i=1}^{l} (y_i - (w^T x_i + b)) x_i$$

$$J(w, b, x_k, y_k) = (y_k - (w^T x_k + b))^2$$

$$\nabla_w J(w, b, x_k, y_k) = -2(y_k - (w^T x_k + b)) x_k$$

$$\frac{\partial J(w, b, x_k, y_k)}{\partial b} = -2(y_k - (w^T x_k + b))$$

Algorithm: Stochastic Gradient Descent Method

* At every iteration, value of η_k should be changed.

Initialise $w^0 = w^{start} \in R^l$ and $b^0 \in R$

Repeat

Randomly select subset B from Training set T.

$$w^{(j+1)} = w^{(j)} - \eta_k \left(\lambda w + \sum_{(x_k, y_k) \in B} \frac{\partial J(w^{(j)}, b^{(j)}, x_k, y_k)}{\partial w} \right)$$

$$b^{(j+1)} = b^{(j)} - \eta_k \left(\sum_{(x_k, y_k) \in B} \frac{\partial J(w^{(j)}, b^{(j)}; x_k, y_k)}{\partial b} \right)$$

Until $\left\| \begin{bmatrix} w^{(j+1)} \\ b^{(j+1)} \end{bmatrix} - \begin{bmatrix} w^{(j)} \\ b^{(j)} \end{bmatrix} \right\| \leq \varepsilon$

One of the method to reduce the value of eta.

$\eta = 0.2$

for $i = 1 : 1000$

$\eta_i = \frac{\eta}{(1 + 0.0008)}$

end.

other methods to reduce the value of η (eta)

- Line search method
- Momentum concept

Stochastic Gradient Descent, Kernel Method Regression (least square)

Algorithm : Stochastic gradient descent method

Initialize $x^0 = u^{start} \in R^k$ and $b \in R$

Repeat

Randomly select subset B from Training set T.

$$u^{(j+1)} = u^{(j)} - \eta_k \left(\lambda u + \sum_{(x_k, y_k) \in B} \frac{\partial J(u, b, x_k, y_k)}{\partial u} \right)$$

$$b^{(j+1)} = b^{(j)} - \eta_k \left(\sum_{(x_k, y_k) \in B} \frac{\partial J(u, b, x_k, y_k)}{\partial b} \right)$$

Until $\left\| \begin{bmatrix} u^{(j+1)} \\ b^{(j+1)} \end{bmatrix} - \begin{bmatrix} u^{(j)} \\ b^{(j)} \end{bmatrix} \right\| \leq \varepsilon$

(x_1, y_1) (x_2, y_2) . . . (x_k, y_k)

$f(x_1)$ $f(x_2)$ $f(x_k)$

- $RMSE = \sqrt{\frac{1}{K}\sum_{i=1}^{k}(y_i - f(x_i))^2}$

- $MAE = \frac{1}{K}\sum_{i=1}^{K}|y_i - f(x_i)|$

$NMSE = \frac{SSE}{SST} = \frac{\sum_{i=1}^{K}(y_i - f(x_i))^2}{\sum_{i=1}^{K}(y_i - \bar{y})^2}$ → Normalised Mean Sq. Error

SST → Sum of Sq. Training Deviation

Variance of predicted fn →

$$\frac{1}{K^2}\left(\sum_{i=1}^{K} f(x_i) - f(\bar{x})\right)^2$$

- $R^2 = \frac{\sum_{i=1}^{K}(y_i - \bar{y})^2}{\sum_{i=1}^{K}(f(x_i) - f(\bar{x}))^2}$

- $Sparsity = \frac{\#(w_i = 0)}{\#(w_i)}$ → $\frac{\text{no. of } (w_i = 0)}{\text{no. of } w_i}$

Sparsity Calculates how many $w_i = 0$, ie it Calculates simplicity of the model and how many total w_i

let,

$w_1 = 2.8$

$w_2 = 0$

$w_3 = 0$

$b = 0.4$

sparsity = 2/3 (here) $= \frac{\#(w_i = 0)}{\#w_i}$ $2\ w_i = 0$ & total $w_i = 3$

* Sparsity should be high is preferable.

& if $w_1 = 2.8$

$w_2 = 1.2$

$w_3 = 0.8$

$b = 0.4$,

then sparsity $= 0$, because no $w_i = 0$.

15/3/23

* MLE

Let $\{x_1, x_2, \ldots, x_n\} \sim D(\theta)$ (any distrbn.)

$\{39.5, 68, 78, 105, 101, 82, \ldots\} \sim N(\mu, \sigma)$

we select parameter θ, for which has max^m prob. given sample D.

ie, $\max_\theta P(\theta/D)$

$= \max_\theta \underbrace{P(D/\theta)}_{\text{Likelihood}} \cdot \underbrace{P(\theta)}_{\text{expert value (prior info)}}$ → sometimes not given, so, we maximise only $P(D/\theta)$

$= \max_\theta \cdot P(D/\theta)$

pts. in D are i.i.d.

$\max_\theta P(D/\theta) = \max_\theta P(x_1, x_2, \ldots, x_n/\theta)$

$\max_\theta P(D/\theta) = \max_\theta \prod_{i=1}^{n} P(x_i/\theta) \cdot = \max_\theta \sum_{i=1}^{n} \log P(x_i/\theta)$

* Let $\{x_1, x_2, \ldots\} \sim N(\mu, \bar{\sigma})$

$\bar{\sigma}$ is any const., then estimate μ.

$\max_{\theta \mu} \sum_{i=1}^{n} \log P(x_i/(\mu, \bar{\sigma}))$ → in case of normal distribution,

$J(\mu) = \max_\mu \sum_{i=1}^{n} \log \frac{1}{\bar{\sigma}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i - \mu}{\bar{\sigma}}\right)^2}$

$J(\mu) = \max_\mu \sum_{i=1}^{n} \log\left(\frac{1}{\bar{\sigma}\sqrt{2\pi}}\right) - \frac{1}{2} \cdot \sum_{i=1}^{n} \left(\frac{x_i - \mu}{\sigma}\right)^2$

derivative $= 0$, ie $\frac{\partial J(\mu)}{\partial \mu} = 0$

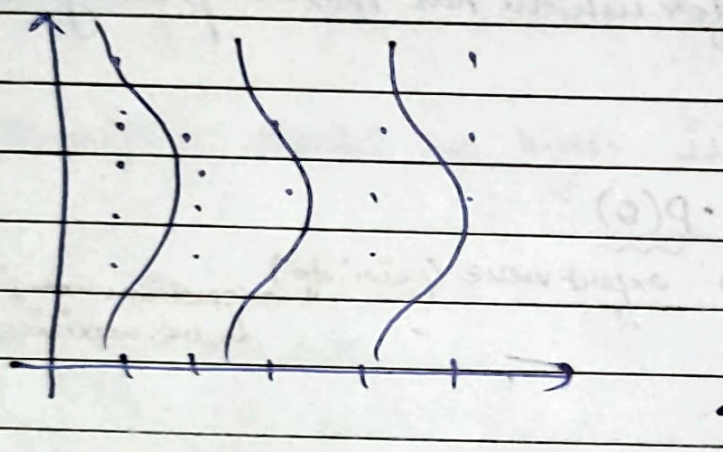
$\frac{\partial J(\mu)}{\partial \mu} \quad \frac{2}{2} \cdot \sum_{i=1}^{n} (x_i - \mu) = 0$

$\mu = \frac{1}{n}\sum x_i$ $\quad \boxed{\mu = \bar{x}}$

* If we take μ as const. & want to estimate σ & maximize σ, then following same step we get as,

then $\boxed{\hat{\sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$

$\Rightarrow$



$\boxed{Y_i = f(x_i) + \varepsilon_i}$

$E(\varepsilon_i) = 0$

$E(Y_i/X) \rightarrow w^T x_i + b$

$w^T \phi(x_i) + b$ (for basis fn)

$Y_i/X \sim N(w^T \phi(x_i) + b, \sigma)$

$\varepsilon_i = y_i - (w^T x_i + b) \sim N(0, \sigma)$

In regression problem, we have,

$T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L)\}$

$x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$.

$Y_i/X_i \sim N(w^T \phi(x_i), \sigma)$

$\max P((y_1, y_2, \ldots, y_L)/(x_1, x_2, \ldots, x_L))$

$\max P(y_1/x_1, y_2/x_2, \ldots, y_L/x_L)$

$$\max \prod_{i=1}^{L} P(y_i/x_i)$$

$$= \max \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i - w^T\phi(x_i))^2}{2\sigma^2}} \quad (\because y_i/x \sim N(w^T\phi(x_i), \sigma)$$

$$\simeq \max \sum_{i=1}^{L} \log\left(\frac{1}{\sigma\sqrt{2\pi}} \cdot e^{\frac{-(y_i - w^T\phi(x_i))^2}{2\sigma^2}}\right)$$

$$= \max \sum_{i=1}^{L} \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \sum_{i=1}^{L} \frac{(y_i - w^T\phi(x_i))^2}{2\sigma^2}$$

$$\max -\sum_{i=1}^{L} (y_i - (w^T\phi(x_i)))^2$$

$$\min_{w} \sum_{i=1}^{L} (y_i - w^T\phi(x_i))^2$$

→ least squared loss fn.

Advantages :

This is eq. fn. of w & b, smooth fn., solving system of equation is easy.

For solving $Y/X \sim N(\,)$, then, only this loss fn. will work.

Disadvantages

It is sensitive to outliers.

↓

To solve this, we need to solve or go for **weighted least square regression model**.

20/3/23

Bias - Variance

$$T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L)\}$$

$$Y \rightarrow f(x) \rightarrow X.$$

$$\min_f \frac{1}{L} \sum_{i=1}^{L} (y_i - f(x_i))^2 \longrightarrow \text{Empirical risk}$$

we should minimise below one instead of above because, only L data points are considered above.

$$\iint (y_i - f(x_i))\, P(X,Y)\, d(X,Y)$$

$$\left[E_T \underbrace{\left(\underbrace{f_T(x_i)}_{\text{Training set}} - \underbrace{f_0(x_i)}_{\text{actual estimate}} \right)}_{\text{Bias}} \right]^2$$

$$Var(f_T(x))$$

→ When we increase the degree of polynomial / we use more complex function.

- Bias decreasing
- Variance increasing

→ $y_i = f_0(x_i) + \varepsilon_i$

Assumptions :

(i) $E(\varepsilon_i) = 0$

(ii) X_i's are iid.

minimise $E_T(Y_i - f_T(x_i))^2$

can be written as,

$$\min E_T(Y_i - f_0(x_i) + f_0(x_i) - f_T(x_i))^2$$

$$\Rightarrow E_T(Y_i - f_0(x_i))^2 + E_T(f_0(x_i) - f_T(x_i))^2 + 2E_T(Y_i - f_0(x_i))(f_0(x_i) - f_T(x_i))$$

Expanding this term below

$$= E_T[(Y_i - f_0(x_i))(f_0(x_i) - f_T(x_i))]$$

$$= E_T(Y_i f_0(x_i)) - E_T(f_0(x_i) f_0(x_i)) - E_T(Y_i f_T(x_i)) + E_T(f_0(x_i) f_T(x_i))$$

~~$E_T(Y_i f$~~ Writing $Y_i = f_0(x_i) + \varepsilon_i$, we get,

$$= E_T[(f_0(x_i) + \varepsilon_i) f_0(x_i)] - E_T[f_0(x_i) f_0(x_i)] - E_T[(f_0(x_i) + \varepsilon_i) f_T(x_i)] + E_T[f_0(x_i) f_T(x_i)]$$

$$= E_T(f_0(x_i) f_0(x_i)) + E_T(\varepsilon_i f_0(x_i)) - E_T(f_0(x_i) f_0(x_i)) - E_T(f_0(x_i) f_T(x_i)) - E_T(\varepsilon_i f_T(x_i)) + E_T(f_0(x_i) f_T(x_i))$$

$$= E_T(\varepsilon_i f_0(x_i)) - E_T(\varepsilon_i f_T(x_i))$$

$\because E(\varepsilon_i) = 0.$

$\therefore = 0$

Hence, $\min E_T(Y_i - f_T(x_i))^2 =$

Hence, $E_T(Y_i - f_0(x_i))^2 + E_T(f_0(x_i) - f_T(x_i))^2 + 0.$

$$= E(\varepsilon_i^2) + E_T(f_0(x_i) - f_T(x_i))^2$$

let $z = f_0(x_i) - f_T(x_i)$

$E(z^2) = Var(z) + [E(z)]^2 \quad (\because V(z) = E(z^2) - (E(z))^2)$

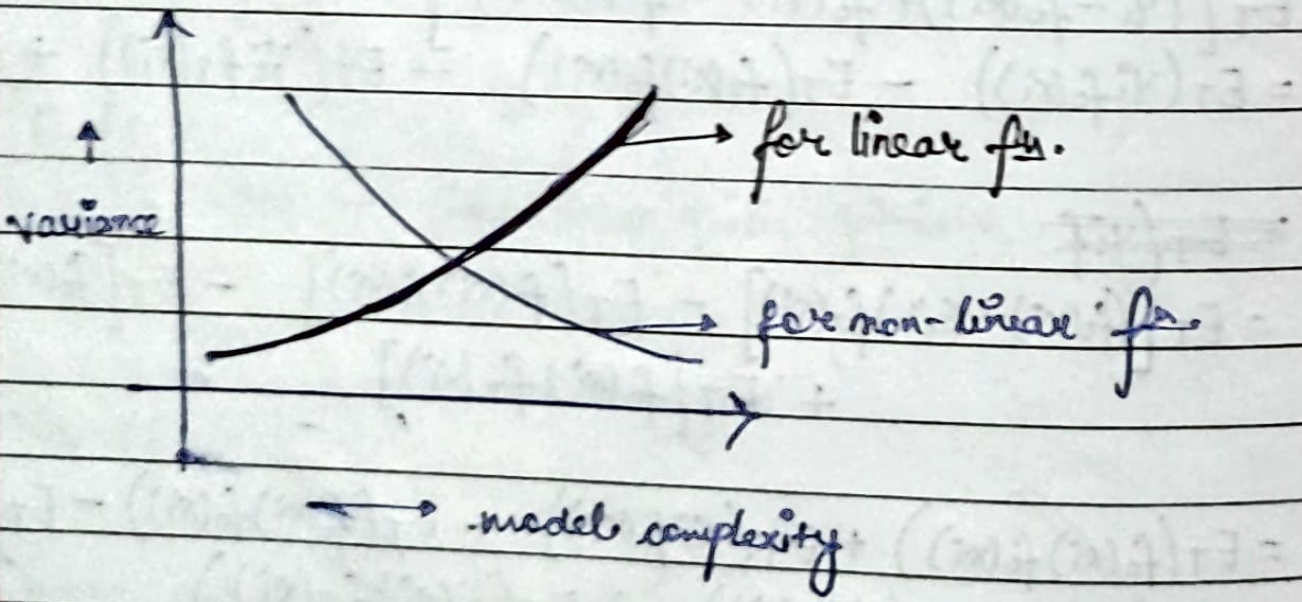
$= E(\varepsilon^2) + Var_T[f_0(x_i) - f_T(x_i)] + E_T(f_0(x_i) - f_T(x_i))^2$

$= E(\varepsilon^2) + Var_T(f_T(x_i)) + E_T[f_0(x_i) - f_T(x_i)]^2$

$(\because Var(kz) = k^2 Var(z))$

$E_T(y_i - f_T(x_i))^2 = [E_T(f_0(x_i) - f_T(x_i))]^2 + Var(f_T(x_i)) + E(\varepsilon^2)$

$\Rightarrow E_T(y_i - f_T(x_i))^2 = \underbrace{[E_T(f_0(x_i) - f_T(x_i))]^2}_{\text{Bias}} + \underbrace{V(f_T(x_i))}_{\text{Variance}} + \underbrace{E(\varepsilon^2)}_{\text{irreducible error}}$



- for linear → as model complexity ↑, Variance ↑
- for non-linear → as complexity ↑, Variance ↓

$\frac{(m+n)!}{m!\,n!}$ m = features, n = degree, n = no. of ph.

Date _____ Page _____

02/3/23

LASSO → Least Absolute Shrinkage & Selection Operator.

Lasso Regression Model -

$$f(x) = w^T x + b \qquad w \in \mathbb{R}^n,\ b \in \mathbb{R}$$

$$:= \bar{w}^T \bar{x} \qquad \text{where, } \bar{w} \in \mathbb{R}^{n+1} \text{ & } \bar{x} = [x, 1]$$

$$f(x) = w^T \phi(x) + b$$

$$:= \bar{w}^T \bar{\phi}(x)$$

$$\phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_{m-1}(x) \end{bmatrix}, \quad w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{m-1} \end{bmatrix}$$

$$\& \quad \bar{\phi}(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_{m-1}(x) \end{bmatrix} \qquad \bar{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ b \end{bmatrix}$$

$$f(x) = w^T \phi(x), \quad w \in \mathbb{R}^m, \quad \phi(x) \in \mathbb{R}^m$$

- 'm' should be chosen as how many features we have.
- Problem with Least Sq. Regression Model: -
  - → Sensitive with outliers
  - → overfitting, as no. of pts. is very less and features is very high.

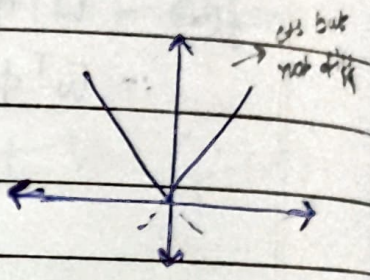
To overcome these problem, we use Lasso Regression Model.

$$\|w\|_1 = \sum_{i=1}^{n} |w_i|$$

$$f(x) = w^T x$$

LASSO Regression Model :

$$J(w) = \min_w \underbrace{\frac{\lambda}{2}\|w\|_1}_{\text{cts but not differentiable}} + \underbrace{\frac{1}{2}\sum_{i=1}^{L}\left(y_i - (w^T \phi(x_i))\right)^2}_{\text{differentiable}}$$



* $\|w\|_1 \longrightarrow$ cts but not differentiable.

$\because \|w\|_1$ is not differentiable, so, we work with sub-gradient.

→ **Sub-Gradient :- (S)**

A sub-gradient of a convex fn. f at $x \in R^m$ is any g, such that

$$f(y) \geq f(x) + g^T(y-x) \quad \forall\ y \in R^m$$

any fn. which satisfies above property, is called the subgradient of that fn.

\# $g(\|x\|_1)$ = sub-gradient of $\|x\|_1$

$$= \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \\ \text{any element in } [-1,1], & x = 0 \end{cases} \qquad x \in R$$

Subgradient of $\|w\|_1$, where $w \in \mathbb{R}^m$.

$$\|w_1\| = \sum_{i=1}^{m} |w_i|$$

$$g(i) = \begin{cases} \text{sign}(w_i) \\ \text{any element in } [-1, 1] \end{cases}, \quad \text{if } w[i] = 0$$

$g(i)$ ← partial subgradient wrt i component

dimension of $g(i)$ is m ie, $g(i) \in \mathbb{R}^m$ ($\because$ here $i = 1 : m$)

$$S(J(w)) = \frac{\lambda g}{2} - \frac{2}{L} \sum_{i=1}^{L} (y_i - (w^T \phi(x_i)) \phi(x_i)$$

$S(J(w))$ ← subgradient; $J(w)$ ← Lasso Reg. model

Algorithm: Subgradient of Lasso Regression model

Input Training set T, tol

Initialize $w^0 \in \mathbb{R}^m$

Repeat

$$w^{k+1} = w^k - \eta_k S(J(w^k))$$

until

$$\| S(J(w^k)) \| \leq tol$$

↳ tolerance.

Stochastic Subgradient Method :-

Input :- Training set T, max_iter

Initialize $w^0 \in R^m$

for $i = 1$ to max_iter

Draw a random subset B of size γ from Training set T

$$w^{k+1} = w^k - \eta_k \, \partial(J_\beta(w^k))$$

end

where,

$$J_\beta(w^k) = \frac{\lambda}{2}\|w\|_1 + \sum_{(x_i, y_i \in \beta)} (y_i - w^T \phi(x_i))^2$$

* Regularised least sq. Regression model is also called Ridge Regression Model.

$$\min_{w,b} \frac{\lambda}{2} w^T w + \frac{1}{L} \sum_{i=1}^{L} (y_i - (w^T x_i + b))^2 \longrightarrow \text{Ridge}$$

$$\min_{w,b} \frac{\lambda}{2}\|w\|_1 + \frac{1}{L} \sum_{i=1}^{L} (y_i - (w^T x_i + b))^2 \longrightarrow \text{Lasso}$$

$$\min_{w,b} \frac{\lambda_1}{2}(w^T w) + \frac{\lambda_2}{2}\|w\|_1 + \frac{1}{L} \sum_{i=1}^{L} (y_i - (w^T x_i + b))^2 \longrightarrow \text{Elastic-Net}$$

$$\min_{w,b} \frac{\lambda}{2} w^T w + \frac{1}{L} \sum_{i=1}^{L} \|y_i - (w^T x_i + b)\|_1$$

↓

This loss fn is optimal when noise is uniformly distributed.

L1 - Norm Regression model :-

$$\min_{w,b} \frac{\lambda}{2}\|w\|_1 + \frac{1}{L}\sum_{i=1}^{L}\|Y_i - (w^T X_i + b)\|_1$$

$$\min_{w,b} \text{Regularisation} + \frac{1}{L}\sum_{i=1}^{L} L(Y_i, w, b)$$

23/3/23

CLASSIFICATION

If any of the data pts. is very large & all other are very less w.r.t to the largest, then it will effect Euclidean distance & every other operation, so we normalised the data. as,

Normalisation :-

For every variable of transaction x = $\frac{\text{Value of var} - \min^m \text{value of var}}{\max^m \text{value of var} - \min^m \text{value of var}}$

* Relative frequency converge to probability for large number of data points.

Bayes formula

$$P(w_i/x) = \frac{P(x/w_i)\,P(w_i)}{P(x)}$$

$P(x/w_i)$: likelihood, $P(w_i)$: Prior prob., $P(x)$: evidence

if $P(w_1/x) \geq P(w_2/x)$; decide w_2
otw, decide w_2.

if $P(X/w_1)\,P(w_1) \geq P(X/w_2)\,P(w_2)$; decide w_1
otw, decide w_2