(1)

Deep Learning

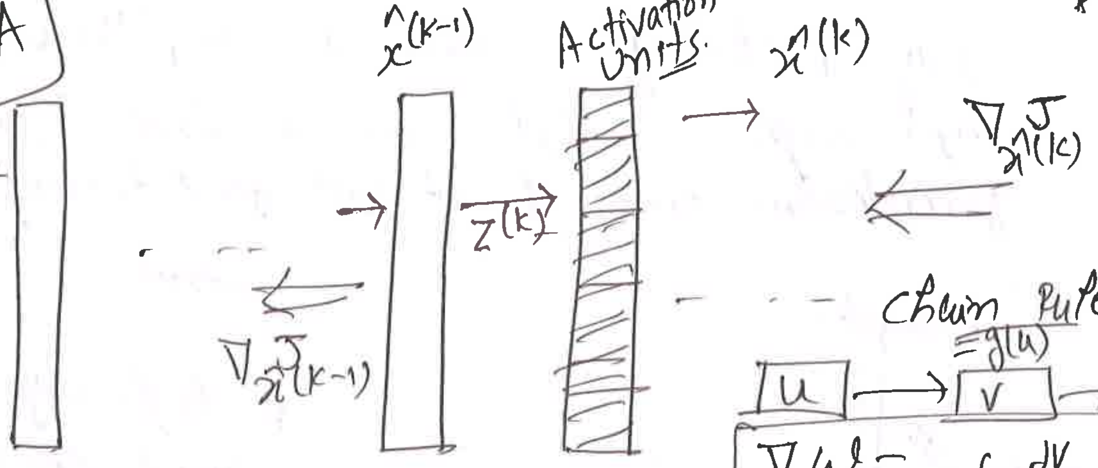
Chain Rule

$$\left(\frac{\partial W}{\partial u}\right)_{jk} = \frac{\partial v_j}{\partial u_k}$$

$W = Au$

$$\boxed{\frac{\partial W}{\partial u} = A}$$

$v_j = \sum_k A_{jk} u_k$



Jacobian

Chain Rule:-

$$\boxed{u} \longrightarrow \boxed{v = g(u)} \longrightarrow \boxed{w = h(v)}$$

$$\boxed{\nabla_u w = \left(\frac{\partial v}{\partial u}\right)^T \nabla_v w}$$

Forward Computation:-

Jacobian $\hat{z}^{(k)} = W^{(k)T} \hat{x}^{(k-1)} + b^{(k)}$

$\hat{x}^{(k)} = g^{(k)} \left(\hat{z}^{(k)}\right)_i$

Notes:-

(A)

$$\boxed{\hat{z}^{(k)}} \longrightarrow \boxed{\hat{x}^{(k)}} \longrightarrow \boxed{J}$$

$\to g'(\hat{z}^{(k)})$

$\Leftarrow \nabla_{\hat{x}^{(k)}} J$

$\hat{W}^{(k)}$

$$\begin{bmatrix} \hat{W}^{(k)}_{:1} & \cdots & \hat{W}^{(k)}_{:\hat{n}(k)} \\ | & & | \\ | & & | \end{bmatrix}$$

$\hat{z}^{(k)}_j = \hat{W}^{(k)T}_{:j} \hat{x}^{(k-1)} + b^{(k)}$

$$\boxed{W^{(k)}} \xrightarrow{\text{Linear Transformation}} \boxed{\hat{z}^{(k)}} \longrightarrow J$$

$$\boxed{\hat{x}^{(k)}} \xrightarrow{\text{Linear trans.}} \boxed{\hat{z}^{(k)}} \longrightarrow J$$

Gradient Computation:-

(1) $\nabla_{\hat{z}^{(k)}} J = \underbrace{\left(\frac{\partial \hat{x}^{(k)}}{\partial \hat{z}^{(k)}}\right)^T}_{\text{Diagonal matrix}} \cdot \nabla_{\hat{x}^{(k)}} J$

$= g'(\hat{z}^{(k)}) \odot \nabla_{\hat{x}^{(k)}} J$

$\cancel{\nabla J}$ $\frac{\partial J}{\partial W^{(k)}_{ij}} = \hat{x}^{(k-1)}_i \cdot \frac{\partial J}{\partial \hat{z}^{(k)}_j}$

(2) $\cancel{\nabla_{\hat{W}^{(k)}_{:j}} J = \nabla_{\hat{W}^{(k)}_{:j}}}$

$\cancel{\frac{\partial J}{\partial \hat{W}^{(k)}_{ij}}} =$ $\hat{z}^{(k)}_j = (W^{(k)T})_{ji} \cdot \hat{x}^{(k-1)}_i + b^{(k)}_j$

$\left(\nabla_{\hat{W}^{(k)}} J\right)_{ij} = \hat{x}^{(k-1)}_i \left(\nabla_{\hat{z}^{(k)}} J\right)_j$

$\nabla_{b^{(k)}} J = 1 \cdot \nabla_{\hat{z}^{(k)}} J$

$\nabla_{\hat{x}^{(k-1)}} J = W^{(k)T} \nabla_{\hat{z}^{(k)}} J$

Topic 2:-

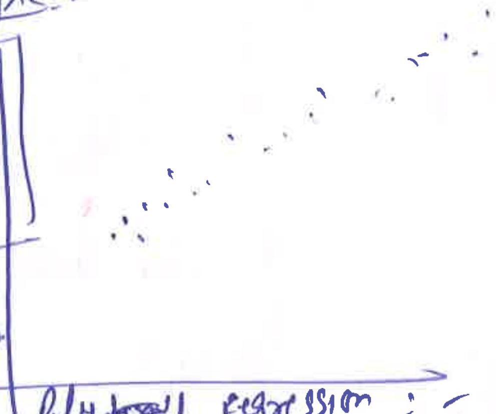
Regularization:-

Key ~~f~~ Challenge:- How to keep the model simple ~~enough~~ so that we ~~a~~ have a good generalization error (& not just good training error).

Example:-

$$\boxed{y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 x^0}$$

what is ~~c~~ the largest n, a_n fo?



~~A high degree polynomial $(n-1)$ fits n points exactly.~~

Fitting a hyperplane in n-dimensions:-

Polynomial regression:-

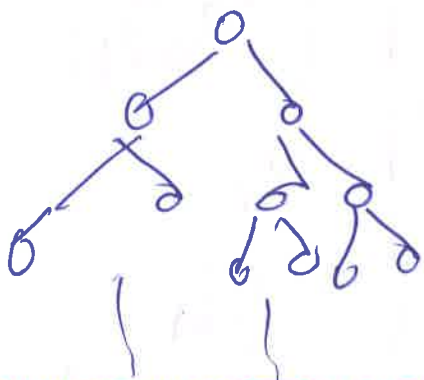
Regularizer:- ① Penalty for having higher degree d

② SVMs:-

$$\min_{w,b} \left[\underbrace{\frac{1}{2} w^T w}_{\text{Regularizer}} + \underbrace{C \sum_{i=1}^{m} \xi_i}_{\text{L2-Norm}} \right] \rightarrow \text{Loss}$$

③ Decision Trees:-

A penalty on number of nodes in the tree



②

For neural networks:-

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha \underbrace{\Omega(\theta)}_{\text{Regularizer}}$$

$\Downarrow$ Regularized Objective / loss Function

$\Downarrow$ LOSS

$\alpha = 0$ No regularization

$\Omega(\theta) = \frac{1}{2} w^T w \rightarrow$ weight vector (Excluding bias term)

Note:- typically do not impose penalty on bias term b (Does not capture correlation among)

Norm of a vector- w:-

$\boxed{p > 0}$

p-norm:- $\left[\sum_{j=1}^{n} \|w_j\|^p\right]^{1/p}$

Different kinds of effects of on parameters

L2- norm:- $p = 2$ $\quad \|W\|_2^2$:- L2-norm square

L1- norm:- $\sum_{j=1}^{n} |w_j|$

L0:- norm:- # of non-zero components of w.

Why minimize $\|W\|^2$?

$J(\theta) =$

L^2 Regularization :-

Simplification: No bias parameters

$$\tilde{J}(\theta; X, y)$$

$$= J(\theta; X, y) + \frac{\alpha}{2} w^T w$$

$$\nabla_w \tilde{J}(w; X, y) = \nabla_w J(\theta; X, y) + \alpha w$$

$\Rightarrow$ Gradient update Rule.

$$w^{new} = w - \eta \cdot \nabla_w \tilde{J}(\theta; X, y)$$

$$= w - \eta [\nabla_w J(\theta; X, y) + \alpha w]$$

$$= w - \eta \alpha w - \eta J(\theta; X, y)$$

$$= w(1 - \eta \alpha w) - \eta J(\theta; X, y)$$

$\Rightarrow$ Equivalent to ~~applying no~~ multiplying weight by $1-\eta$ a constant $(1-\eta\alpha w)$ before applying non-regularized gradient update

Another insight:-

Let $w^* = \underset{w}{\arg\min}\ J(w)$

Approximate $J(\theta)$ using a quadratic approximation around w^*

③

Taylor approximation: $\theta(w, b)$

~~Approx~~

~~Approx~~ $J'(\theta) = J(w^*) + \underbrace{\nabla_w J(w^*)}_{\to 0} \cdot (w - w^*)$,

Since w^* is minima

$+ \frac{1}{2}(w - w^*)^T \underbrace{H(w)}_{\text{Hessian matrix}} (w - w^*)$

$= J(w^*) + \frac{1}{2}(w - w^*)^T H(w^*)(w - w^*)$

(Are semi-definite).

$$H = \begin{array}{cc} & j \\ i & \begin{bmatrix} & & \\ & \dfrac{\partial J(\theta)}{\partial \theta_j \partial \theta_i} & \\ & & \end{bmatrix} \end{array}$$

$$\boxed{f(x) = f(x^*) + (x - x^*) f'(x^*) + \frac{1}{2}(x - x^*)^2 f''(x^*)}$$

Quadratic approximation to a function $f: \mathbb{R} \to \mathbb{R}$ around x^*.

~~Now, let w add the~~ weight decay ~~term to $J'(w)$~~

~~$\Rightarrow$~~ $\nabla_w J'(w) =$ ~~$(w - w^*)$~~ $H(w^*)(w - w^*)$

~~$\Rightarrow$~~ Adding regularization term & taking derivative:-

$$\nabla_w \tilde{J}'(w) = \cancel{(w-w^*)} + \cancel{H(w-w^*)}\ H(w^*)(w-w^*) + \alpha \cdot \nabla_w \tfrac{1}{2} w^T w$$

$$= H(w^*)(w-w^*) + \alpha I w$$

Equating it to zero, we get:-

$$\cancel{H(w^*)w}\ Hw - \cancel{Hw^*} + \alpha I w = Hw^*$$

$$w(H+\alpha I) = Hw^*$$

$$\boxed{w = (H + I\alpha)^{-1} H w^*}$$

if $\alpha = 0$, $\boxed{w = w^*}$ $\boxed{QQ^T = 1}$

H: real & symmetric :- orthonormal basis

$H = Q \Lambda Q^T$ (Eigenvalue decomposition)

Λ ↳ Diagonal

$$w = (Q\Lambda Q^T + \alpha I)^{-1} Q \Lambda Q^T w^*$$

$$= [Q\Lambda Q^T + Q \alpha I Q^T]^{-1} Q \Lambda Q^T w^*$$

$$= (Q(\Lambda + \alpha I) Q^T)^{-1} (Q \Lambda Q^T) w^*$$

$$= \boxed{Q (\Lambda + \alpha I)^{-1} \Lambda Q^T w^*}$$

Each eigenvalue λ_i is scaled by

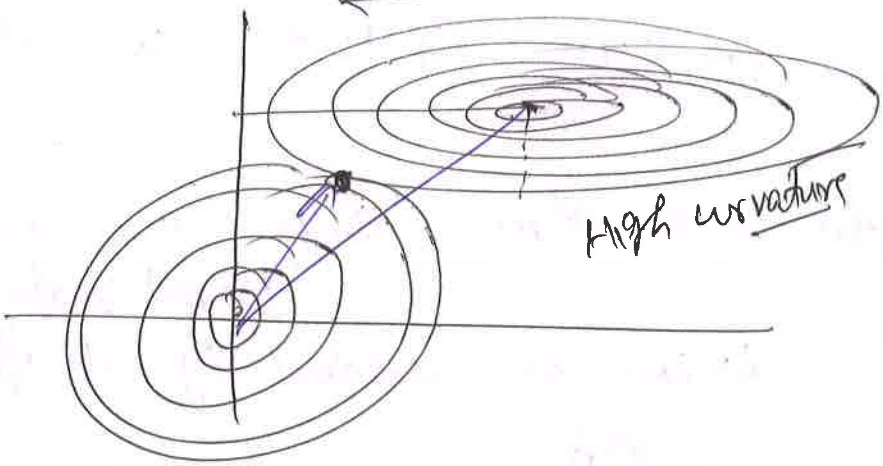
1

scaling each direction (i-th eigenvector) by $\frac{\lambda_i}{\lambda_i + \alpha}$

λ_i : curvature

Low curvature ← λ_i is small

scale by $\frac{\lambda_i}{\lambda_i + \alpha}$



High curvature

L1:- Regularization :-

$$\Omega(\theta) = \|W\|_1 = \sum_{i=1}^{n} |w_i|$$

$$\tilde{J}(w; X, y) = J(w; X, y) + \alpha \|W\|_1$$

$$\nabla_w \tilde{J}(w; X, y) = \nabla_w J(w; X, y) + \alpha \, \text{sign}(w)$$

$\Downarrow$ sub-gradient

Gradient update

old: $\Rightarrow \quad w \leftarrow w - \eta \cdot \nabla_w J(w; X, y)$

new (regularization): $w \leftarrow w - \eta \left[\nabla_w J(w; X, y \pm v\right]$

v ⇓ sign of w

vector of 1's & -1's (or possibly zeros)

$v = \text{sign}(w)$

May not get a closed form

$$J'(w; x, y) = J(w^*, x, y) + (w - w^*) \nabla_w J(w^*; x, y)$$
$\Downarrow$ Quadratic approximation around w^*

$\Downarrow 0$

$$+ (w - w^*)^T \cancel{\nabla_w} H(w^*) (w - w^*)$$

$$\nabla_w J'(w; x, y) = H(w^*)(w - w^*)$$

Another assumption - Data is un-correlated (input features)

↳ can be achieved by doing linear PCA.

$$H = \begin{bmatrix} H_{11} & & & & \\ & H_{22} & & & \\ & & H_{33} & & \\ & & & \ddots & \\ & & & & H_{n,n} \end{bmatrix}$$

$$\Rightarrow \nabla_w \tilde{J}'(w; x, y) = H(w - w^*) + \cancel{\alpha \operatorname{sig}} \; \alpha \sum_{i=1}^{n} |w_i|$$

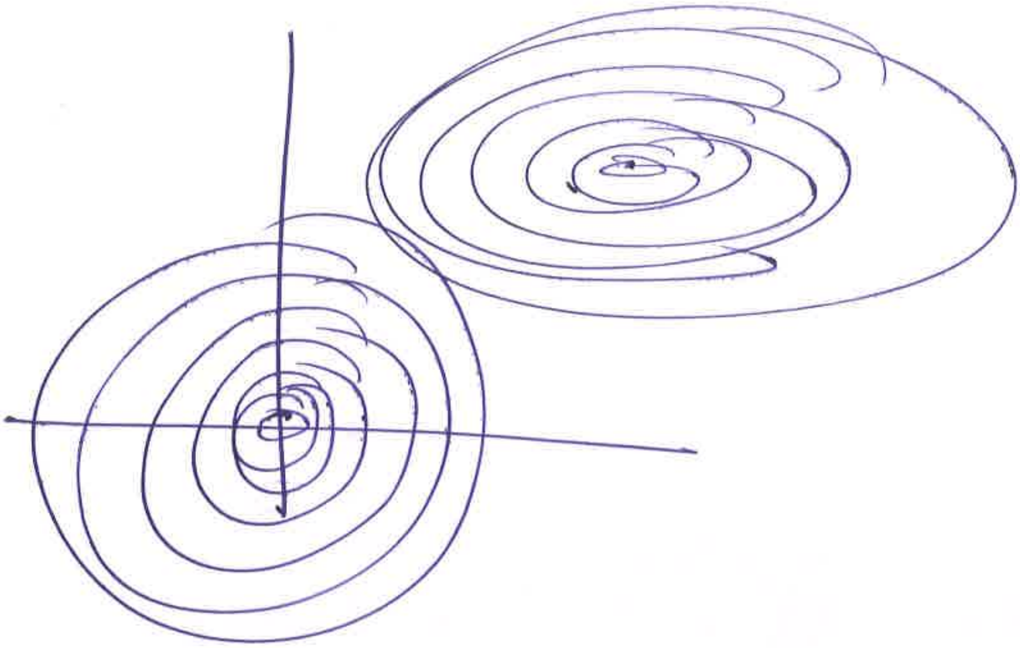
$$\Rightarrow \nabla_w \tilde{J}'(w; x, y) = \sum_i H_{i,i}(w_i - w_i^*) + \alpha \sum_i \operatorname{sign}(w_i)$$

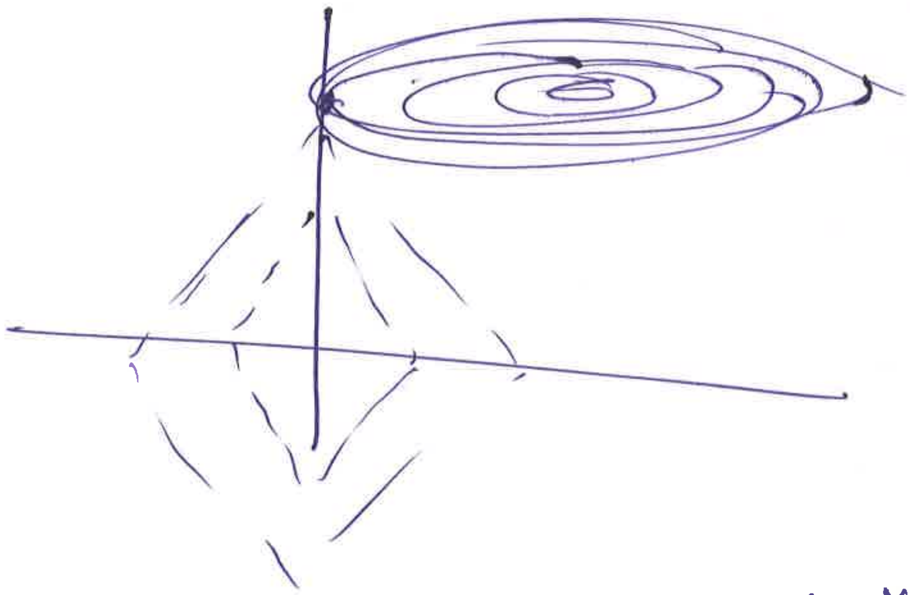
~~Equating to zero:~~

$$\cancel{w_i} \quad \cancel{w_i - w_i^* = \frac{\alpha \operatorname{sign}(w)}{H_{i,i}}}$$

$w_i^* \geq 0$

$$w_i = \quad w_i \geq 0 \quad \boxed{w_i^* + \frac{\alpha}{H_{ii}}} \qquad w_i = w_i^* + \frac{\alpha}{H_{ii}}$$





$$w_i = \mathrm{sign}(w_i^*)\max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}$$