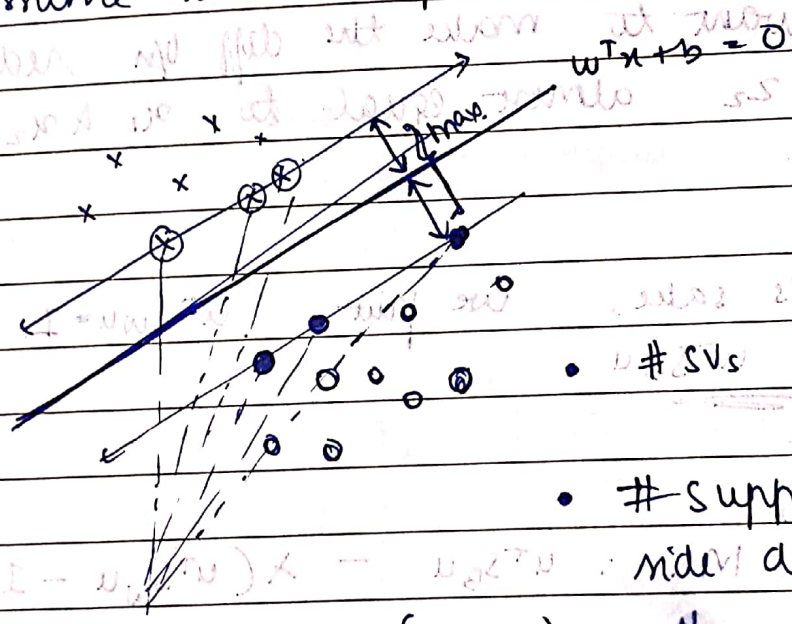
Lecture 14

Support Vector Machine

* Possibility of wrong test data if line is too close to the samples

+ Want to maximize distance b/n +ve & -ve samples with the line. (Illar to logistic regression)

o Assume linear separability.

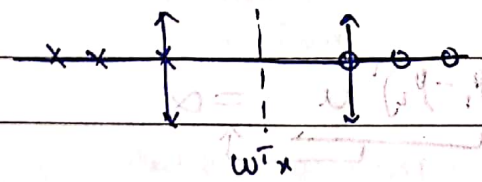


- # SVs on either side can't be 0
- # support vectors on either side don't have to be equal.

(6) Support vectors (points)

– non-support vectors (points) don't contribute to final soln.

(?). Can have 1 SV on each side



SVM → best linear classifier

Scanned by CamScanner

For line $ax + by + c = 0$, dist from origin $= \frac{c}{\sqrt{a^2+b^2}}$

$\therefore$ for $w^T x + b = 0$, dist $= \frac{b}{\|w\|}$ } maximize

Illy, $w^T x + b = \alpha$

$\Rightarrow w^T \frac{x}{\alpha} + \frac{b}{\alpha} = 0$

$\Rightarrow$ ① $w^T x + b = -1$ ← ② $w^T x + b = 1$

$\Rightarrow$ max: $\frac{-1-b}{\|w\|}$ max $= \frac{1-b}{\|w\|}$

② − ①,

maximize: $\frac{1-b}{\|w\|} - \frac{(-1-b)}{\|w\|} = \cancel{\left\| \frac{0 - 2b}{\|w\|} \right\|}$ $\cancel{\left\| \frac{2b}{\|w\|} \right\|}$

$= \boxed{\frac{2}{\|w\|}}$

If y_i is +ve, $w^T x_i + b > 1$

If y_i is −ve, $w^T x_i + b < -1$

} Generally, $y_i(w^T x + b) \geq 1$

> 1

→ constraint optimization, convex.

- not sensitive to initialization.

~~Minimize $\frac{1}{2} w^T w$~~

Wake up!

~~$y_i (w^T x + b) \geq 1$, $\forall i \in 1 \ldots N$~~

~~where $y_i = 1$ or -1~~

Minimize $\frac{1}{2} w^T w$

where $y_i (w^T x + b) \geq 1$ $\forall i \in 1 \ldots N$

$y_i = \pm 1$

(N constraints)

* DUAL

maximize $$J = \frac{1}{2} W^T W - \sum_{i=1}^{N} \alpha_i \left[y_i (W^T x_i + b) - 1 \right] \quad \text{—(1)}$$

$$\frac{\partial}{\partial W} = 0$$

so that : $$W - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

$$\Rightarrow \boxed{W = \sum_{i=1}^{N} \alpha_i y_i x_i}$$

where α_i are 0 for non-support vectors

∴ W is initially a lin. combination of x_i's
& we said that only ~~supp~~ support vectors affect the sol'n
W → line of separation.

Subs val. of W in (1),

$$J = \frac{1}{2} \left(\sum_{i=1}^{N} \alpha_i y_i x_i \right)^T \cdot \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right) - \sum_{i=1}^{N} \alpha_i \times \left[\left(y_i \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right)^T x_i + b \right) - 1 \right]$$

$$\Rightarrow J = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{N} \alpha_i + b \underbrace{\sum_{i=1}^{N} \alpha_i y_i}_{0}$$

Scanned by CamScanner

Diff ① wrt b,

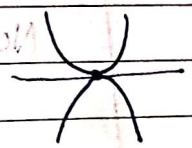
$$\frac{\partial}{\partial b} = 0 - \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0$$

Subs this result in prev eqn,

$$\Rightarrow J = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

☆ Primary problem: $\underset{w}{\text{Min}} \ \underset{\alpha}{\text{Max}} \ J$ (P)



$\Updownarrow$

Duality gap = 0 ∵ same sol'n for both if it's a convex optimization problem

$\underset{w}{\text{Max}} \ \underset{\omega}{\text{Min}} \ J$ (D) : Dual problem

Q. Why did we convert to a dual problem? (when we probably could have solved the primal problem w/ grad. descent etc.)

A. Pay attention next class ☺ {when we get samples as dot prods}

☆ $\frac{\#SV}{N}$ is an estimate of error rate

(∵ in LOO, if we leave out a sv & test on it, there is a chance of making a mistake)

Hard margin – we don't allow any samples from either class b/n support vectors of each class.

Soft margin – we allow a few samples in the boundary. (relax the constraints for a few samples.

Let the 'special' sample have a ε_i of 1. Everything else has $\varepsilon_i = 0$.

Now,

$$\text{if } y_i = 1, \quad (w^T x + b) > 1 - \varepsilon_i$$

$$\text{if } y_i = -1, \quad (w^T x + b) < -1$$

$\varepsilon_i \rightarrow$ +ve scalar representing dist from point (non-negative) to SVs.

$$\text{Min. } \sum_{i=1}^{N} \varepsilon_i$$

$\therefore$ Soft margin $\Rightarrow$

$$\text{Minimize } \frac{1}{2} w^T w + \overset{C}{\sum_{i=1}^{N}} \varepsilon_i$$

$$y_i (w^T x + b) \geq 1 - \varepsilon_i$$

we can specify # violations by using L_0 norm, i.e. $\|\varepsilon\|_0$

Scanned by CamScanner

Hard margin – we don't allow any samples from either class b/n support vectors of each class.

Soft margin – we allow a few samples in the boundary. (relax the constraints for a few samples.

Let the 'special' sample have a ε_i of 1. Everything else has $\varepsilon_i = 0$.

Now,

If $y_i = 1$, $(w^T x + b) > 1 - \varepsilon_i$

If $y_i = -1$, $(w^T x + b) < -1$

$\varepsilon_i \rightarrow$ +ve scalar (non-negative) representing dist from point to SVs.

$$\text{Min. } \sum_{i=1}^{N} \varepsilon_i$$

∴ soft margin ⇒

$$\text{Minimize } \frac{1}{2} w^T w + \overset{C}{\sum_{i=1}^{N}} \varepsilon_i$$

$$y_i (w^T x + b) \geq 1 - \varepsilon_i$$

we can specify # violations by using L_0 norm, i.e. $\|\varepsilon\|_0$.

Scanned by CamScanner

- Regularizing constant 'c'

If c is v. small (~ 0),
very lax with violations
S.Vs can be very far apart.

If c is v. big ($\sim \infty$)
becomes a hard margin.

where $\min \frac{1}{2} w^T w + C \sum_{i=1}^{N} \varepsilon_i$

Scanned by CamScanner

Date

Lecture 15

SVM problem :

a) Primal problem

$$\min \frac{1}{2} w^T w$$

can be solved with gradient descent etc.

b) Dual

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

we dont always need / want to compute w.

ie : $\because w = \sum_{i=1}^{N} \alpha_i y_i x_i$

we want sign $(w^T x + b)$ → order of α d

or, sign $\left(\sum_{i=1}^{N} \alpha_i y_i x_i^T x + b \right)$

↓

sparse ($\because$ few SVs)

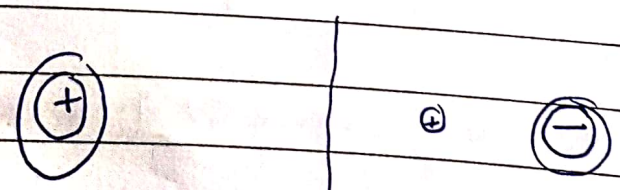
$\Rightarrow O(|SV| \times d)$

↳ as we don't compute w earlier.

$\Rightarrow$ slow computation

Can allow misclassification during training process for a larger margin.

eg :



→ controlled by C in : $\min \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$

We don't need (have) α for test sample derived from training samples.

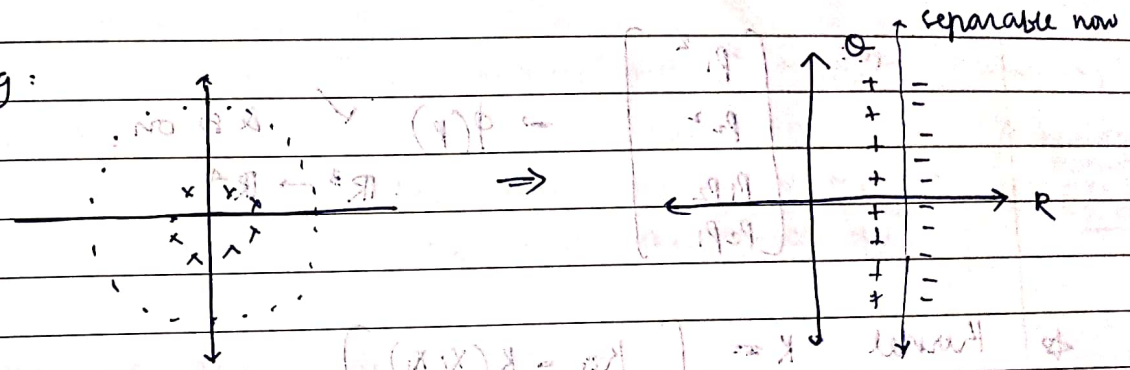
KERNEL

Create a new feature space where problem becomes well behaved / separable

↳ transform feature space (can increase dim #)

$$x \Rightarrow \phi(x)$$

eg:



Cartesian ⇒ Polar space.

Define a kernel $K(p,q) = \phi p^T \phi(q)$ ← kernel

↳ feature map

$p, q \rightarrow$ 2 samples in 2D space

eg: quadratic kernel

$$p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \quad q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

scalar $\rightarrow (p^T q)^2 = (p_1 q_1 + p_2 q_2)^2$

$$= p_1^2 q_1^2 + p_2^2 q_2^2 + 2 p_1 p_2 q_1 q_2$$

$$= \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2} p_1 p_2 \end{bmatrix}^T \begin{bmatrix} q_1^2 \\ q_2^2 \\ \sqrt{2} q_1 q_2 \end{bmatrix}$$

$$= \phi(p)^T \phi(q)$$

We don't need to know ϕ, as long as we get the dot prod.

We don't need (have) α for test sample. Derived from training samples.

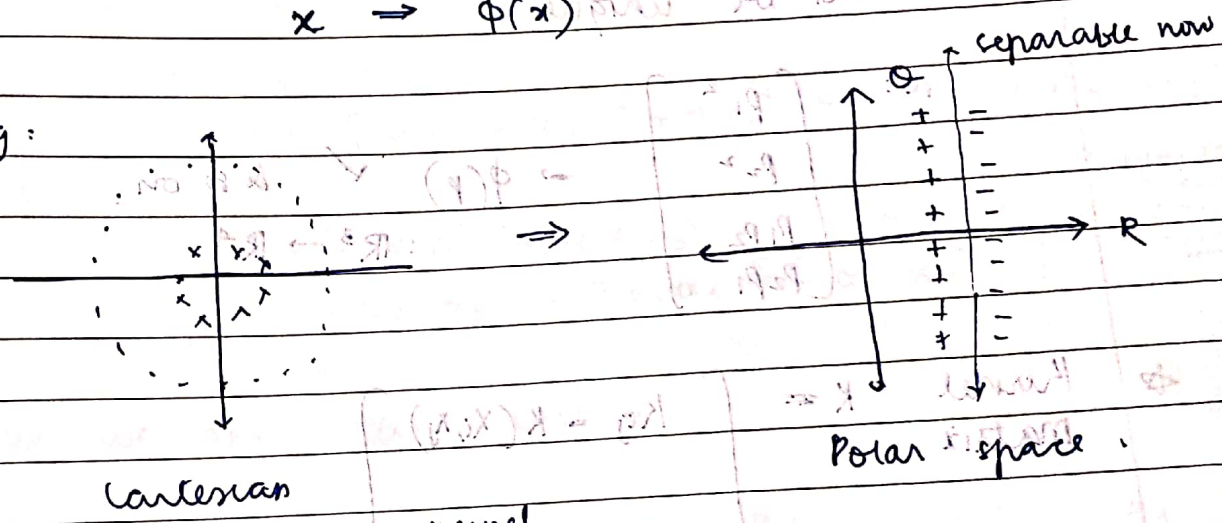
KERNEL

Create a new feature space where problem becomes well behaved / separable

↳ transform feature space (can increase dim #)

$$x \rightarrow \phi(x)$$

eg:



Define a kernel $K(p,q) = \phi p^T \phi(q)$ ← kernel

↳ feature map

$p, q \rightarrow$ 2 sam in 2D spa

eg: quadratic kernel

$$p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \quad q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

scalar $\rightarrow (p^T q)^2 = (p_1 q_1 + p_2 q_2)^2$

$$= p_1^2 q_1^2 + p_2^2 q_2^2 + 2 p_1 p_2 q_1 q_2$$

$$= \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2} p_1 p_2 \end{bmatrix}^T \begin{bmatrix} q_1^2 \\ q_2^2 \\ \sqrt{2} q_1 q_2 \end{bmatrix}$$

$$= \phi(p)^T \phi(q)$$

We don't need to know ϕ, as long as we get the dot prod.

Wherever we have dot product,

eg : $w^T x$, put = kernel $K(w, x)$.

$$\phi : \mathbb{R}^2 \to \mathbb{R}^\infty$$

to get ∞, $e^{-p^T q}$, use Taylor series etc.

From prev. eg : ϕ corresponding to a kernel need not be unique.

ie :
$$\begin{bmatrix} p_1^2 \\ p_2^2 \\ p_1 p_2 \\ p_2 p_1 \end{bmatrix} \to \phi(p) \checkmark \quad \text{& so on.}$$
$$: \mathbb{R}^3 \to \mathbb{R}^4$$

☆ Kernel matrix $K = \begin{bmatrix} K_{ij} = K(x_i, x_j) \\ \\ \end{bmatrix}_{N \times N}$

eg 2) : EXOR problem

	O	
X		O
O		X

p_1	p_2	y :
+1	+1	-1
+1	-1	+1
-1	+1	+1
-1	-1	-1

Notice that products of samples in same class are equal.

$$K(p,q) = (1 + p^T q)^2$$
$$= (1 + p_1 q_1 + p_2 q_2)^2$$
$$= (1 + p_1^2 q_1^2 + p_2^2 q_2^2 + \dots \quad 2(p_1 p_2 + p_1 q_1 + p_2 q_2 + q_1 q_2 \dots))$$

(PTO)

$K_{ij} = (1 + x_i^T x_j)^2$

$$\Rightarrow \phi(p) = \begin{bmatrix} 1 \\ p_1^2 \\ p_2^2 \\ \sqrt{2}\, p_1 p_2 \\ \sqrt{2}\, p_1 \\ \sqrt{2}\, p_2 \end{bmatrix} \qquad K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}_{4\times 4}$$

$$\max J(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$$

$$- \frac{1}{2} \Big[9\alpha_1^2 + 9\alpha_2^2 + 9\alpha_3^2 + 9\alpha_4^2 \quad (\text{when } i = j)$$

$$+ (-2\alpha_1\alpha_2) + (-2\alpha_1\alpha_3) + 2\alpha_1\alpha_4 + \quad (\because \text{symmetry } y_1 y_2 \text{ and } y_2 y_1 \text{ exists})$$

$2, 9 \rightarrow y_i y_j$

$$2\alpha_2\alpha_3 + (-2\alpha_2\alpha_4) + (-2\alpha_3\alpha_4) \Big]$$

When we diff $J(\alpha)$ wrt α_1,

$$1 - 9\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 \quad \text{—} ① \quad \alpha_1$$

$$1 - 9\alpha_2 + \alpha_1 - \alpha_3 + \alpha_4 \quad \text{—} ② \quad \alpha_2$$

and so on ..

We finally get $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8}$

Now, we want

$$W = \sum_{i=1}^{4} \alpha_i y_i \phi(x_i)$$

$$\therefore W = \frac{1}{8} \left[-\phi(x_1) + \phi(x_2) + \phi(x_3) - \phi(x_4) \right]$$

$$= \frac{1}{8} \left[- \begin{bmatrix} 1 \\ p_1^2 \\ \vdots \\ 1 \end{bmatrix} + \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} - \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} \right] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \boxed{\text{non-0}} \\ 0 \\ \vdots \end{bmatrix}$$

p_1, p_2 $\sqrt{2}\, p_1 p_2 \neq 0$ for all 4

$K_{ij} = (1 + x_i^T x_j)^2$

$$\Rightarrow \phi(p) = \begin{bmatrix} 1 \\ p_1^2 \\ p_2^2 \\ \sqrt{2}\, p_1 p_2 \\ \sqrt{2}\, p_1 \\ \sqrt{2}\, p_2 \end{bmatrix} \qquad K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}_{4\times 4}$$

$$\max J(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$$

$$- \frac{1}{2} \Big[9\alpha_1^2 + 9\alpha_2^2 + 9\alpha_3^2 + 9\alpha_4^2 \quad (\text{when } i = j)$$

$$+ (-2\alpha_1\alpha_2) + (-2\alpha_1\alpha_3) + 2\alpha_1\alpha_4 + \quad (\because \text{symmetric, } y_1 y_2 \text{ and } y_2 y_1 \text{ exists})$$

$$2\alpha_2\alpha_3 + (-2\alpha_2\alpha_4) + (-2\alpha_3\alpha_4) \Big]$$

$2, 9 \rightarrow y_i y_j$

When we diff $J(\alpha)$ wrt α_1,

$$1 - 9\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 \quad \text{— (1)} \quad \alpha_1$$

$$1 - 9\alpha_2 + \alpha_1 - \alpha_3 + \alpha_4 \quad \text{— (2)} \quad \alpha_2$$

and so on ..

We finally get $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8}$

Now, we want

$$W = \sum_{i=1}^{4} \alpha_i y_i \phi(x_i)$$

$$\therefore W = \frac{1}{8} \left[-\phi(x_1) + \phi(x_2) + \phi(x_3) - \phi(x_4) \right]$$

$$= \frac{1}{8} \left[- \begin{bmatrix} 1 \\ p_1^2 \\ \vdots \end{bmatrix} + \begin{bmatrix} \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \\ \\ \end{bmatrix} - \begin{bmatrix} \\ \\ \\ \end{bmatrix} \right] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \boxed{\text{non-0}} \\ 0 \\ \vdots \end{bmatrix}$$

$\sqrt{2}\, p_1 p_2 \neq 0$ for all 4

re:- all ~~other~~ dimensions other than $\sqrt{2}p_1p_2$ are zero.

$\Rightarrow$ p_1p_2 is a feature space where the problem is clearly linearly separable.

Q. How do we decide K?

A. We can have kernels defined over vectors (here), strings, graphs etc.
when lin. combination of kernels is a valid kernel.
RBF is generally a powerful kernel to use.

- In prev problem, α's were all equal :- all 4 samples were support vectors.

☆ Kernel matrix is always a PSD

Disadvantage: [if not, large no of SVs, ~~so~~ if so.]
$\because$ α's are sparse, we still have to compute kernels for all possible samples, which is expensive $\Rightarrow$ slow at test time
$\Downarrow$
probably more than computing 'w'.

- Approximate $\phi()$ to help this problem.

Scanned by CamScanner

Lecture 16

If we can rewrite the algo as dot products, we can convert to kernel.

i.e. replace $x^T y$ with $K(x, y)$

$\Downarrow$

ϕ (inc. dimensions usually)

$\Downarrow$

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

expressive power of non-linearity + stability of linearity

Assume $K(x_i, x_j)$ measures some kind of similarity.

Old perceptron algo:

If $\text{sign}(y_i (w^T x_i)) < 0$

err $\Rightarrow$ fix / refine $\bar{w}$

$w^{k+1} \leftarrow w^k - \eta\, y_i x_i$

If $w_0 = 0$, w^t can be expressed as a linear combination of $y_i x_i$.

$$\boxed{w^* = \sum_{i=1}^{N} \alpha_i x_i}$$

Dual

Dual

$\therefore$ Finding a d-dim w $\Leftrightarrow$ finding 'N' α's

Primal

(N-dim α)

Pick either based on which is smaller.

$\alpha \to 0$, if no error in ~~sample~~ sample.

$\alpha \sim$ sparse vec

Scanned by CamScanner

Kernels in PCA

Eigenvectors are lin. combinations of samples

$$C = \sum_{i=1}^{N} x_i x_i^T \quad \text{—— ①} \quad \text{when mean} = 0$$

$$Cv = \lambda v$$

From ①,

$$\sum_{i=1}^{N} x_i \underbrace{x_i^T v}_{\alpha_i} = \lambda v$$

$$\therefore \boxed{v = \sum_{i=1}^{N} \alpha_i x_i} \quad \text{(merging } \lambda\text{)}$$

Naive method,

$$C = \sum_{i=1}^{N} \phi(x_i)\, \phi(x_i)^T$$

But if feature space is ∞ earlier, hard to find eigvals

(eigvector) $v = \sum_{i=1}^{N} \alpha_i \phi(x_i)$

Subs this in eqn: $Cv = \lambda v$

$$\therefore \frac{1}{M} \sum_{i=1}^{M} \phi(x_i)\, \phi(x_i)^T \sum_{j=1}^{M} \alpha_j \phi(x_j) = \lambda \sum_{i=1}^{M} \alpha_i \phi(x_i)$$

We want to get even # ϕs on both sides to get rid of easier.

Mult. both sides by $\phi()^T$, (premultiply $\phi(x_k)$ on both sides, where x_k is one of the samples)

$$\phi(x_k)^T \cdot \frac{1}{M} \sum_{i=1}^{M} \phi(x_i)\, \phi(x_i)^T \sum_{j=1}^{M} \alpha_j \phi(x_j) = \lambda\, \phi()^T \sum_{j=1}^{M} \alpha_i \phi(x_i)$$

Scanned by CamScanner

$\phi()^T \phi() = K()$ $\rightarrow$ scalar

$$\therefore \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{M} \underbrace{\phi(x_k)^T \phi(x_i)}\, \alpha_j \underbrace{\phi(x_j)^T \phi(x_j)}$$

$$= \lambda \sum_{i=1}^{M} \underbrace{\phi(x_k)^T \phi(x_i)}$$

$$\therefore \boxed{\frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{M} K_{ki} K_{ij} \alpha_j = \lambda \sum_{i=1}^{M} K_{ki} \alpha_i}$$ run for $k = 1..M$

$$\therefore \boxed{K^2 \vec{\alpha} = M \lambda K \vec{\alpha}}$$

where K (kernel matrix) $= \begin{bmatrix} K_{ij} \\ \downarrow \\ K(x_i, x_j) \end{bmatrix}$ $M \times M$

☆ and α^k is k^{th} eigenvector of K ($\because$ M eigvectors)

$\because$ $K\alpha = (M\lambda)\alpha$

$\Rightarrow$ non-linear version of PCA.

lower dim. subspace $\Rightarrow$ non-linear subspace.

Normal PCA gives us eigvectors & we further find the projections. Here, we directly find the projection.

If data wasn't centred i.e.

$$C = \sum_{i=1}^{M} [x_i - \mu][x_i - \mu]^T$$

Kernel matrix for centred data & kernel matrix for non-centred data are related.

$\phi()^T \phi() = K()$ $\rightarrow$ scalar

Page No
Date / /

$$\therefore \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{M} \underbrace{\phi(x_k)^T \phi(x_i)}\, \alpha_j \underbrace{\phi(x_j)^T \phi(x_j)}$$

$$= \lambda \sum_{i=1}^{M} \underbrace{\phi(x_k)^T \phi(x_i)}$$

$$\therefore \boxed{\frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{M} K_{ki} K_{ij} \alpha_j = \lambda \sum_{i=1}^{M} K_{ki} \alpha_i}$$ run for $k = 1 \ldots M$.

$$\therefore \boxed{K^2 \vec{\alpha} = M \lambda K \vec{\alpha}}$$

where K (kernel matrix) $= \begin{bmatrix} K_{ij} \\ \downarrow \\ K(x_i, x_j) \end{bmatrix}_{M \times M}$

★ and α^k is k^{th} eigenvector of K ($\because$ M eig vectors)

$$\because K\alpha = (M\lambda)\alpha$$

$\Rightarrow$ non-linear version of PCA.

lower dim. subspace $\Rightarrow$ non-linear subspace.

Normal PCA gives us eig vectors & we further find the projections. Here, we directly find the projection.

If data wasn't centered ie.

$$C = \sum_{i=1}^{M} (x_i - \mu)(x_i - \mu)^T$$

Kernel matrix for centred data & kernel matrix for non-centred data are related.

Scanned by CamScanner

$$\therefore \; C = \left[\phi(x_i) - \frac{1}{M} \sum_{i=1}^{M} \phi(x_i) \right] \left[\phi(x_i) - \sum_{i=1}^{M} \phi(x_i) \times \frac{1}{M} \right]$$

. . . .

Keep computing kernel of uncentred data.

We get α, get v from it (eigen vector)

Dot product of v with a sample.

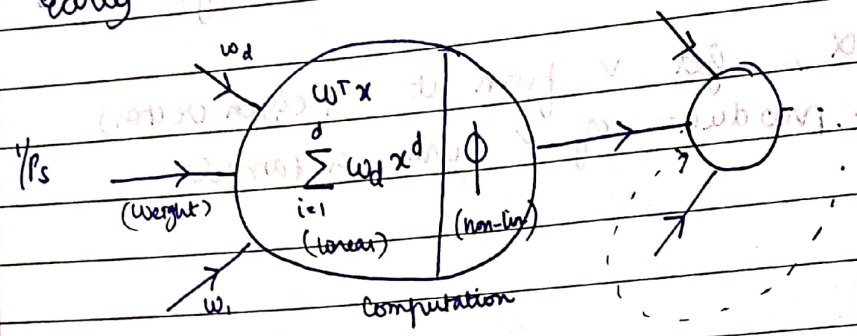
Scanned by CamScanner

10/10/19

Lecture 17

Neural Networks – non-linear methods

Early NN model :



- 2 major classes :-
  - i) Feed forward : no cycles, O/p doesn't travel back to previous nodes (CNNs)
  - ii) Feed back : has cycles (RNNs)

In SVMs, Illan to kernels

$$w^T x \longrightarrow \phi(w)^T \phi(x)$$

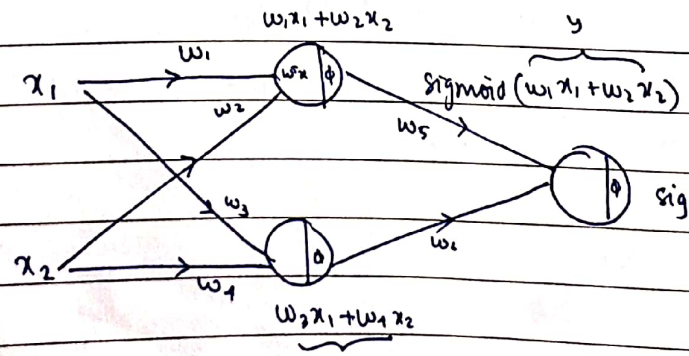
Representation / features / embeddings

became more advanced (enriched)

$$\Psi(w, x) \qquad \Psi(w, \phi(x))$$

(MLP) (DL) Deep learning

eg :



ϕ: $\frac{1}{1+e^{-(\pm w_i \ldots)}}$

$\text{sig}(w_5\, \text{sig}(y) + w_6\, \text{sig}(z))$

parameterized by $w_1 \ldots w_6$.

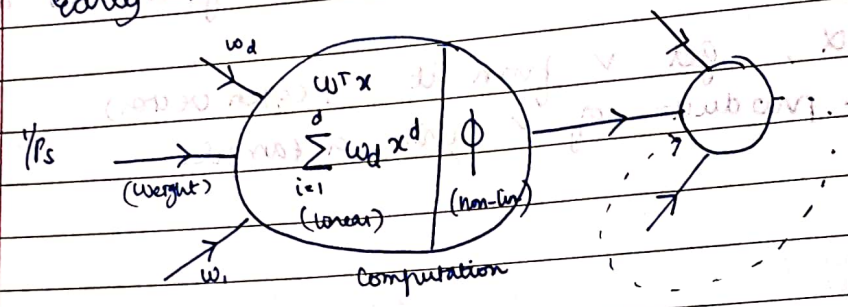
Scanned by CamScanner

10/10/19

Lecture 17

Neural Networks — non-linear methods

Early NN model :



- 2 major classes :-
  - i) Feed forward : no cycles, O/p doesn't travel back to previous nodes (CNNs)
  - ii) Feed back : has cycles (RNNs)

In SVMs,

Illan to kernels

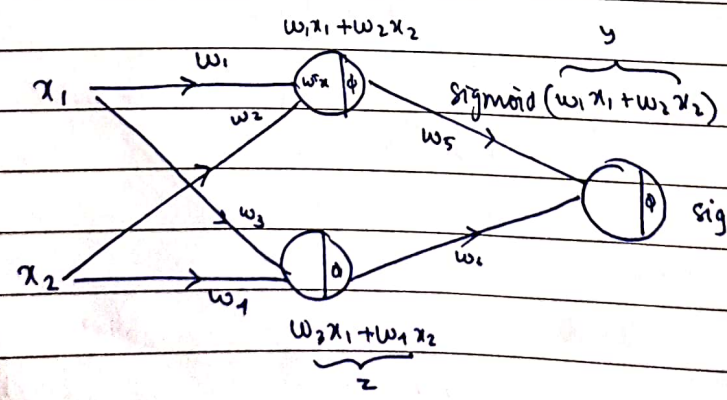
$$w^T x \longrightarrow \phi(w)^T \phi(x)$$

Representation / features / embeddings became more advanced (enriched)

$$\Psi(w, x) \qquad \Psi(w, \phi(x))$$

(MLP) (DL) Deep learning

eg :



$sig(w_5\, sig(y) + w_6\, sig(z))$

$\frac{1}{1 + e^{-(\ldots)}}$

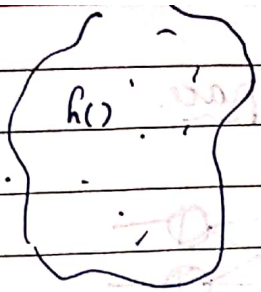
parameterized by $w_1 \ldots w_6$.

Scanned by CamScanner

(Req.)

$g()$

unknown function $g()$.



$w[\ ]$ (params)

(Empirical loss on training data)

Find best $h()$ such that $\min_i g \sim h_i$.

For prev eg: h = NN, MLP with (2, 2, 1) — y_p, hidden neurons, o/p

We can represent $\circ \to \circ$ of NNs

$$[\ ] = \phi\left(\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$$

helps does calc. parallely.

Q10 Reliable software stack

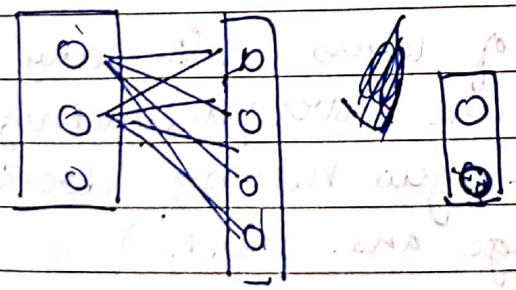
~~Can increase depth & width to fit~~

~~More ~~para~~ parameters ⇒ more ~~exactness~~ samples, func. versus poly~~

Can increase depth (of NN) & width.

More parameters → more training samples needed.

eg 2)

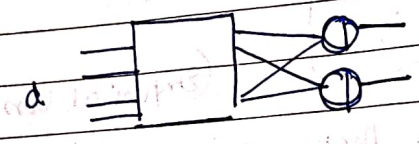


3 4 2

$\left(\frac{4\times 3}{w_1}\right) = \left(\frac{2\times 4}{w_2}\right)$

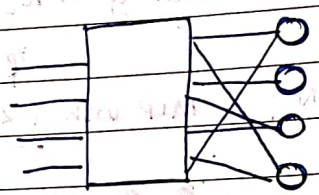
Encoding output space

a)



(makes strong assumption about data / classes that may/not be true)

b)

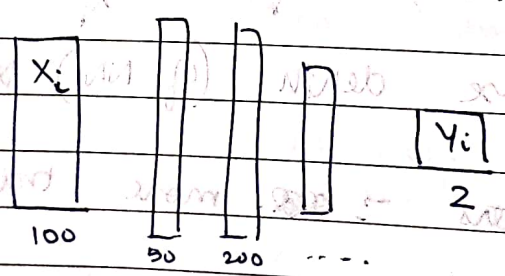


(explodes if # classes is v. high)

Time series data :

$$x(n) = \alpha_1 x(n-1) + \alpha_2 (x(n-2)) + \alpha_3 \cdot x(n-3)$$

Q. If we want to convert X to Y, how many layers do we need? What size?



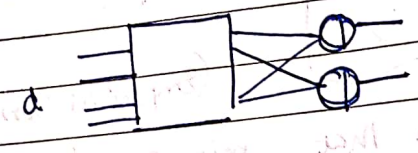
A. We can use any # of layers with any # of size.
∵ neural networks are over parameterized.
(so failure of a few neurons doesn't really change ans. much)

⇒ architecture doesn't really matter.
We are not looking for most reduced / compact sol'n.

Scanned by CamScanner

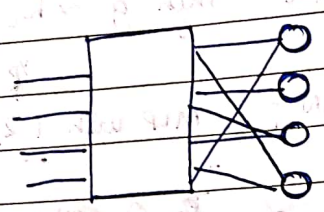
Encoding output space

a)



(makes strong assumption about data / classes that may/not be true)

b)

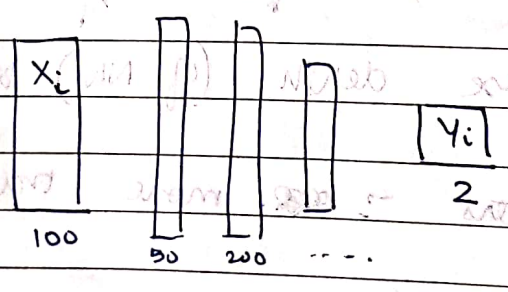


(explodes if # classes is v. high)

Time series data :

$$x(n) = \alpha_1 x(n-1) + \alpha_2 (x(n-2)) + \alpha_3 \cdot x(n-3)$$

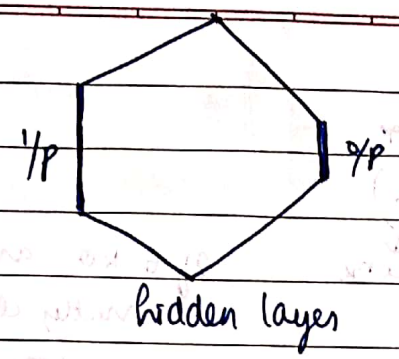
Q. If we want to convert X to Y, how many layers do we need? What size?



A. We can use any # of layers with any # of size, ∵ neural networks are over parameterized. (so failure of a few neurons doesn't really change ans. much)

⇒ architecture doesn't really matter.
We are not looking for most reduced / compact sol'n.

Scanned by CamScanner



advisable arch.

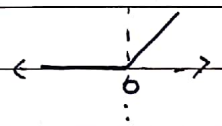
Drawback of over parameterization → overfitting

* Activation function / non-linear functions increases / changes the expressive nature of a neural network.

A deep neural network w/ no non-linearity is only as good as a single layer perceptron.

* Rectified Linear Unit (ReLu) (piecewise linear activation function)

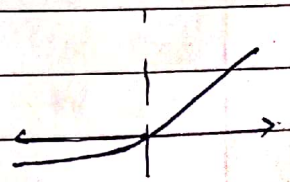
$y = 0 \quad x < 0$
$y = x \quad x \geq 0$



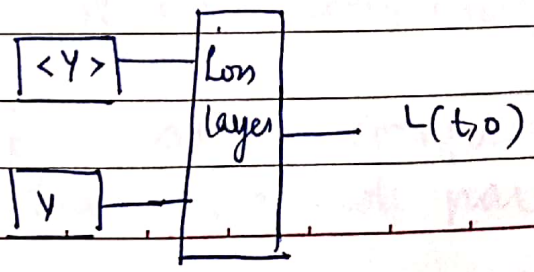
(+) gradients don't saturate (unlike sigmoid & tanh)

Leaky ReLu (improvement)

y = v. small -ve quantity $\quad x < 0$
$y = x \quad x \geq 0$



* Loss layer compares predicted o/p with actual o/p.



eg: MSE, cross entropy, hingeloss (svm)

Scanned by CamScanner

o/p, target

eg: Hinge loss

$$\max(\overset{\text{zero}}{0}, 1 - \underset{\pm 1}{t} \cdot \underset{w^T x}{o})$$

o/p

If t & o are +ve, then correctly classified + zero loss

For SVMs, we have a margin

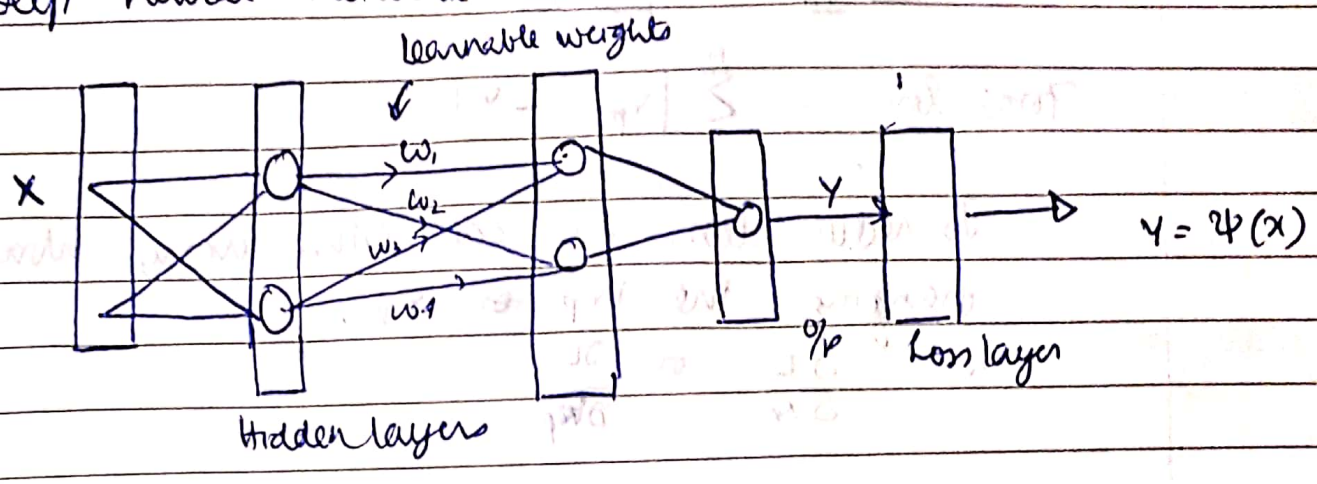
eg: if $t = +1$, $o = 0.5$, we still get a loss of 0.5, even though it was correctly classified ∵ margin < 1.

$$\frac{1}{2} w^T w + C \sum_{i=1}^{N} \max(0, 1 - t_i(w^T x_i))^2$$

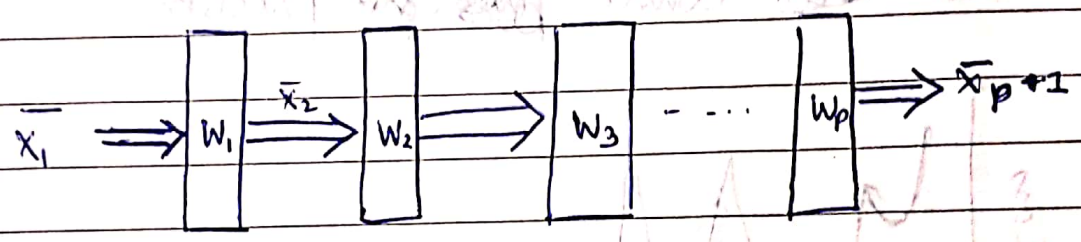
Issue: Loss functions may not be differentiable so hard to perform back prop.

Lecture 18

Deep neural network looks like :



☆ Back propagation



$$x_{p+1} = \Psi(x_1)$$

(Here, weights are inside boxes rather than outside (prev.)).

wrt.

$$\boxed{\underset{n}{x_{q+1}} = \phi(\underset{n\times m}{W}\ \underset{m}{\bar{x}_q})}$$

learn

MLP:

$$\begin{bmatrix} x_2 \end{bmatrix} = \phi\left(\underset{2\times 2}{\begin{bmatrix} W \end{bmatrix}} \begin{bmatrix} x_1 \end{bmatrix}\right)$$

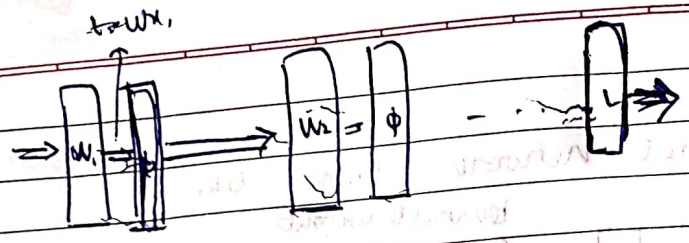
Learning problem → update matrix W

$$W_q^{K+1} \leftarrow W_q^{K} - \eta \nabla J$$, where K is an epoch.

- Loss layer (final) compares $\bar{X}_{p+1}$ with $\bar{Y}$.

Network is a set of computational blocks that may/may not have learnable parameters

Scanned by CamScanner



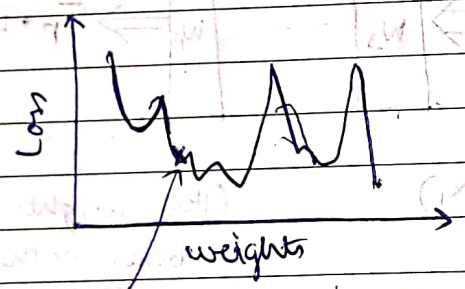
Total loss $= \sum^{N} |x_{p+1} - y|$

To reduce loss, we can either worry about changing the w_p or x_p.

ie $\frac{\partial L}{\partial x_p}$ or $\frac{\partial L}{\partial w_p}$

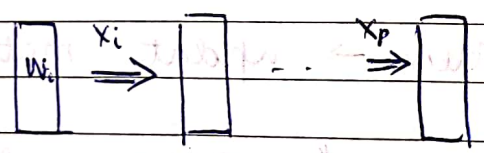
Initially, convex problem ~~[illegible]~~ becomes a non-~~[illegible]~~ convex ~~[illegible]~~ problem

eg:



local minima problem. But it could exist in >2 dimensions. So we could still reach the global minima through gradient descent.

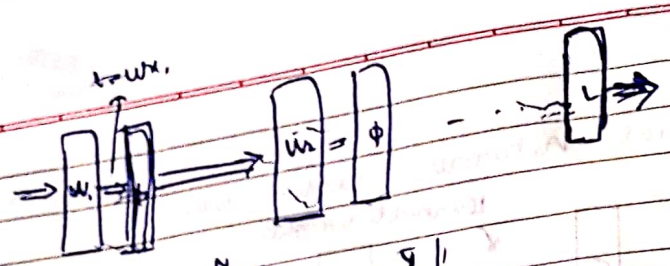
Consider:



$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial x_p} \times \frac{\partial x_p}{\partial x_{p-1}} \times \cdots \frac{\partial x_{i+1}}{\partial x_i}$$ (all values larger than i)

When we define a custom block for our NN, we need to know: $\boxed{\frac{\partial x_{i+1}}{\partial x_i}}$ o/p / i/p

Scanned by CamScanner



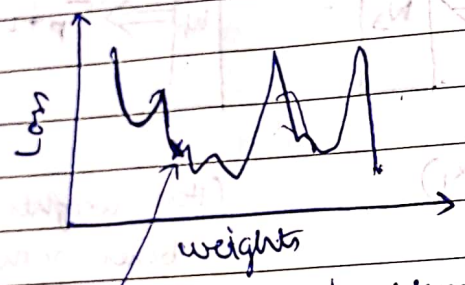
Total loss $= \sum^{N} |x_{p+1} - \hat{y}|$

To reduce loss, we can either worry about changing the w_p or x_p.

i.e $\frac{\partial L}{\partial x_p}$ or $\frac{\partial L}{\partial w_p}$

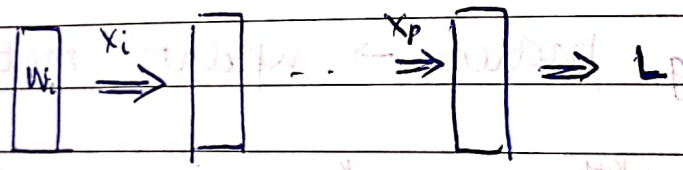
Initially, convex problem becomes a non-convex problem

eg:



local minima problem. But it could exist in >2 dimensions. So we could still reach the global minima through gradient descent.

Consider:



$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial x_p} \times \frac{\partial x_p}{\partial x_{p-1}} \times \cdots \frac{\partial x_{i+1}}{\partial x_i}$$ (all values larger than i)

When we define a custom block for our NN, we need to know: $\boxed{\frac{\partial x_{i+1}}{\partial x_i}}$ o/p / i/p

So, $\boxed{\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial x_i} \times \frac{\partial x_i}{\partial w_i}}$

For learnable parameters θ, we also need to know

$$\frac{\partial y_p}{\partial \theta}$$

$\frac{\partial o/p}{\partial i/p}$ and $\frac{\partial o/p}{\partial \theta}$ are computable by chain rule.

1. Do forward pass with initialized Ws.
2. Calc Loss at the end & update weights w/ G.D.
3. Iterative also, so repeat.

$$W_q^{k+1} \underset{q=1}{\leftarrow} W_q^{k} - \eta \frac{\partial L}{\partial W_q}$$

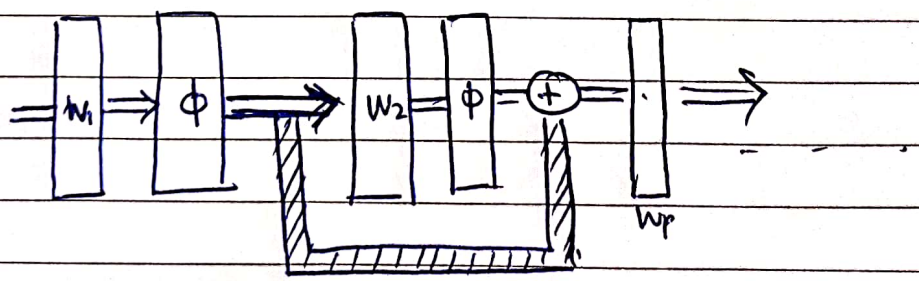
calc this by chain rule

When individual gradients are small, & we keep multiplying them, total product gets very small, vanishing gradient problem.

Skip connections

Add a bypass (addition) between the architecture.

$\frac{\partial L}{\partial w_i}$ can be computed by either chain.



alt. path!

So, $$\boxed{\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial x_i} \times \frac{\partial x_i}{\partial w_i}}$$

For learnable parameters θ, we also need to know

$$\frac{\partial y_p}{\partial \theta}$$

$\frac{\partial o/p}{\partial i/p}$ and $\frac{\partial o/p}{\partial \theta}$ are computable by chain rule.

1. Do forward pass with initialized Ws.
2. Calc Loss at the end & update weights w/ G.D.
3. Iterative also, so repeat.

$$W_q^{k+1} \underset{q=1}{\leftarrow} W_q^k - \eta \frac{\partial L}{\partial W_q}$$

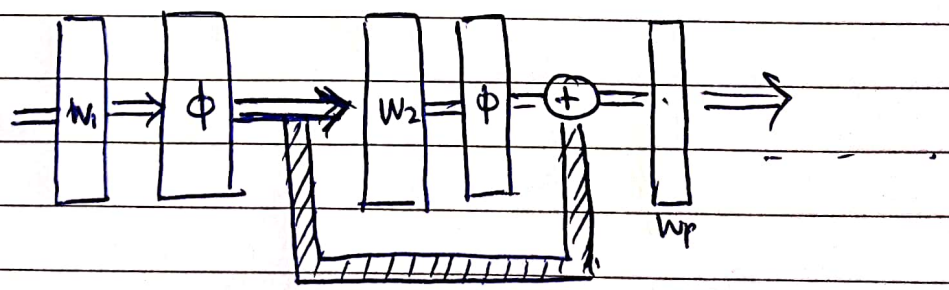
calc this by chain rule

When individual gradients are small, & we keep multiplying them, total product gets very small, vanishing gradient problem.

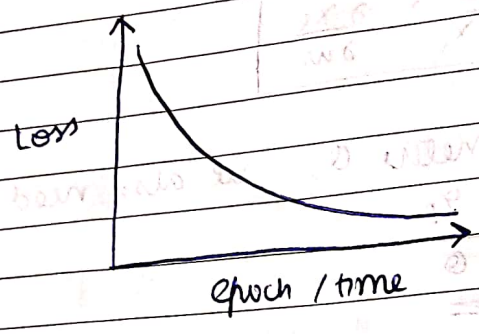
Skip connections

Add a bypass (addition) between the architecture.

$\frac{\partial L}{\partial w_1}$ can be computed by either chain.



alt. path!



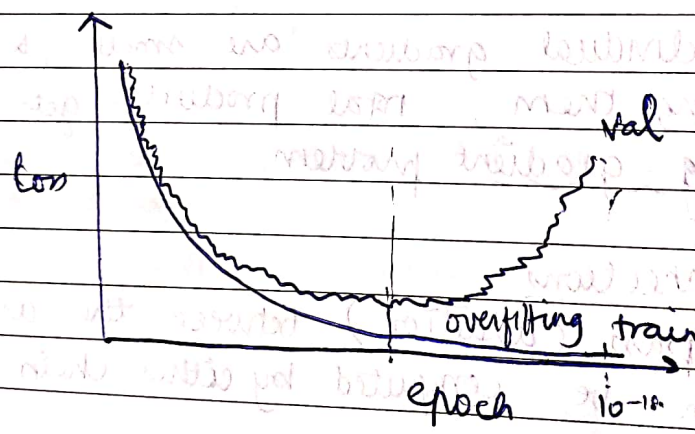
Monotonically non-increasing

(no zigzag ∵ batch g.d.)

Termination criteria :

① when weights don't change

② when $|L^{k+1} - L^{k}| < 10^{-18}$.

↳ NO! Why ?

When we use validation data on the network, (but loss is computed only through training data)