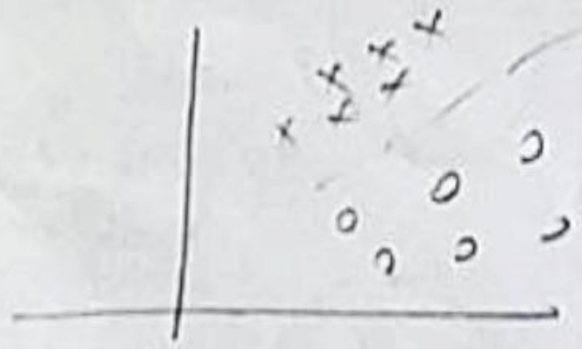
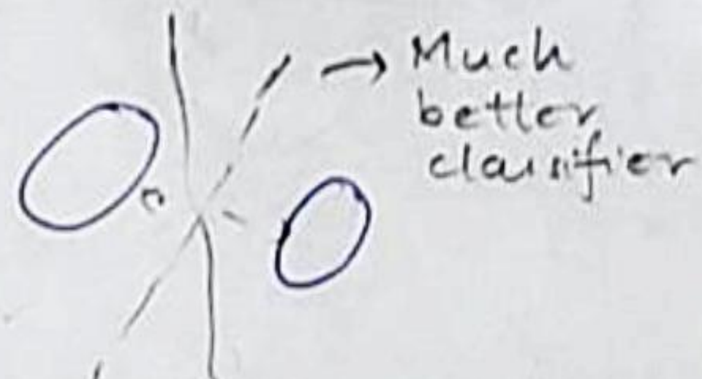
24/9/18

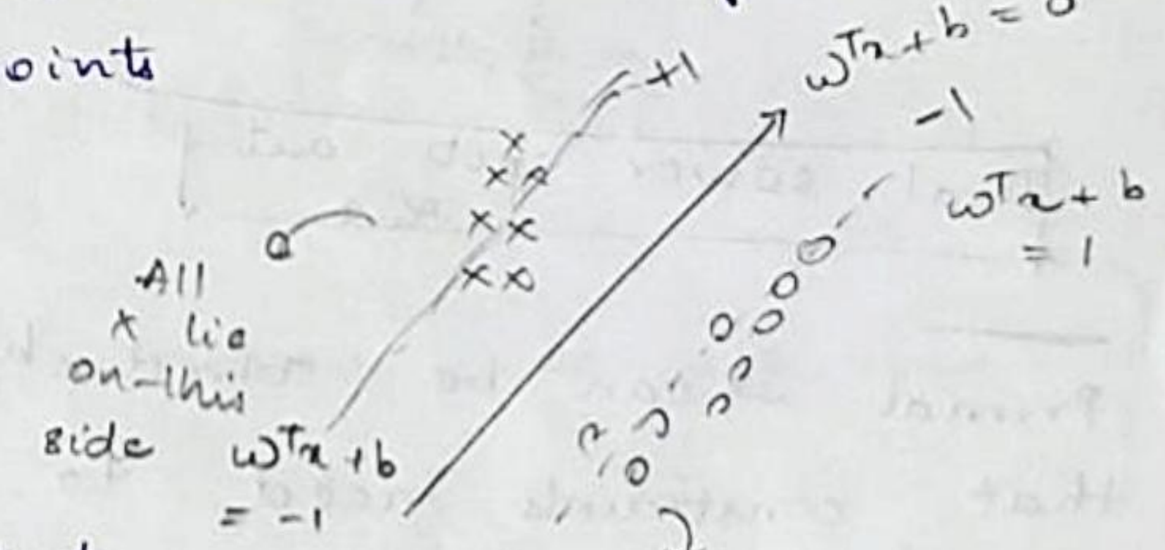
Lec-14



- We were interested in a hyper plane that separates the classes.
- The plane which places training data in the correct location
- We want the hyper plane, that also maximizes distance b/w the classes



- A normal classifier gives a feasible solution, but not enough in practice
- SVM can do that → (that => it can classify w ensure maximum distance b/w points



- Draw 2 lines,
- Now our problem, is maximize margin such that samples are correctly classified

- ±1 dont have a significance.

Distance b/w the seperators

$$\frac{b+1}{\|w\|} - \frac{b-1}{\|w\|} = \frac{2}{\|w\|} \rightarrow \boxed{\text{needs to be maximized}}$$

and $\begin{cases} w^T x_i + b \geq 1 & \text{if } y_i = +1 \\ w^T x_i + b \leq 1 & \text{if } y = -1 \end{cases}$

Combine → $y_i(w^T x_i + b) \geq 1, \quad \forall i$ → Converting

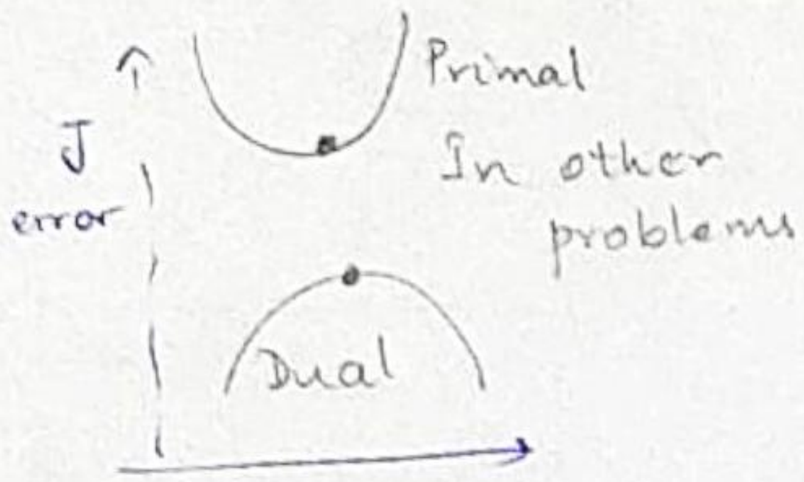
minimize $\rightarrow \frac{\|w\|}{2} = \frac{w w^T}{2}$; with condition $y_i(w^T x_i + b) \geq 1$

↓ primal

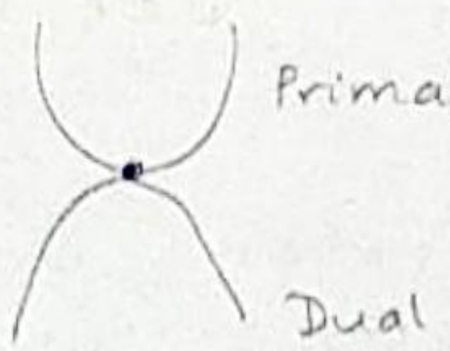
$y_i \in \{+1, -1\}$.

- SVM is actually, practised as per the theoretical basis //

- Dual x Primal



In other problems



SVM

↓

Both have the same error, as SVM is a well-behaving problem.

Dual : $\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} (\alpha_1 \alpha_2 y_1 y_2 x_i^T x_i)$

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

↓

Lagrangian multiplier

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

Dual solver spits out α_i's

Primal → can be solved by G.D., (Keeping in mind that constraints need to be satisfied).

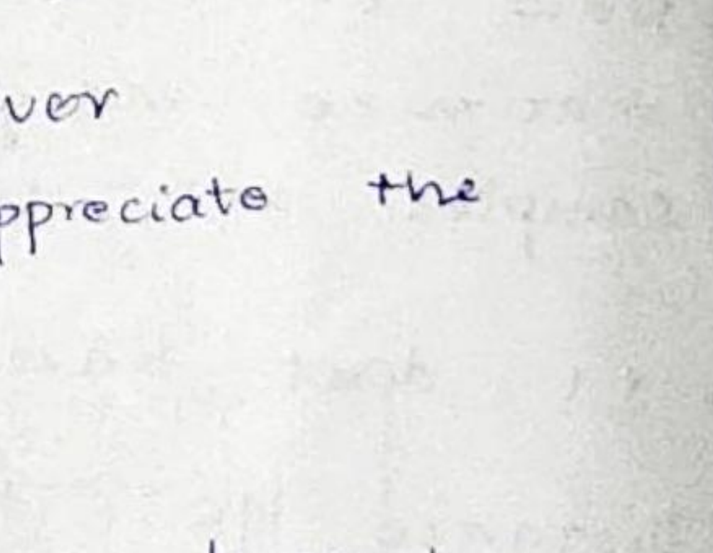
Dual → Solved by a quadratic solver

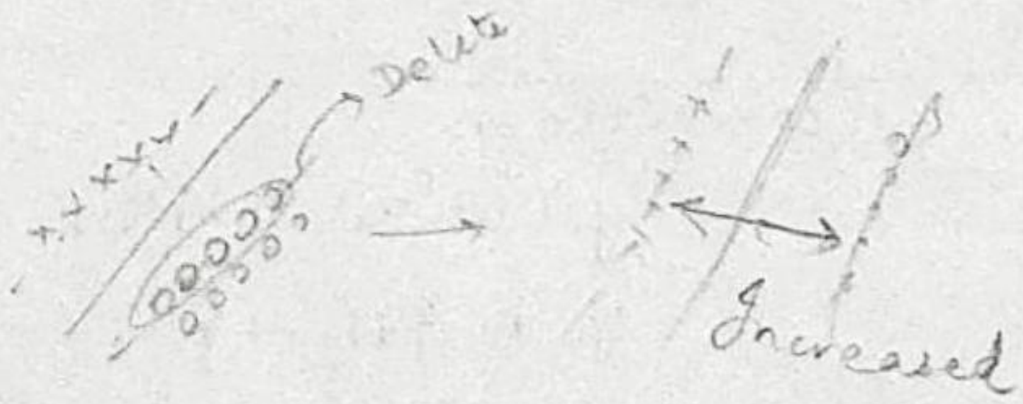
- Why dual is needed -? : We can appreciate the soln better.

- SVM looks for Max-margin seperating hyperplane

- $\alpha = 0$; for samples that don't matter.

So α is non zero for the frontier points





- So $\bar{\alpha}$ is sparse $\alpha = [\alpha_i \text{ from } 1 \text{ to } n] = [\alpha_1, \alpha_2 \ldots \alpha_n]$
  ie for most samples, α is ~~sparse~~ zero.

— Most solvers give non-zero α's.

If ~~as~~ $\alpha \ll n \Rightarrow$

support vectors ⇒ those points with non-zero α.

The final decision depends on S.V.s

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i \xrightarrow{\text{convert to}} w = \sum_{i \in SV}^{N} \alpha_i y_i x_i$$

Toy prob

- If 1M i.ll samples & 100 SVMs ⇒ why do you need others?
- ~~first~~ We don't ~~need to~~ know SVM's in the first place
- Only after all shit is done, we get soln.

Testing data:

$$w^T x + b \gtrless 0$$

Put w.

$$\sum \alpha_i y_i x_i^T x + b \gtrless 0.$$

- Notice that everywhere $x^T x$ is present. So, we will exploit it, later

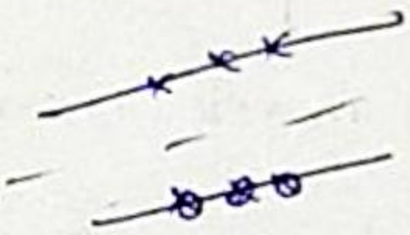
Leave-1-out:

Remove 1 sample & build the classifier

Test on it & find avg

- In SVMs :-
  If non SV ⇒ no error
  If SV ⇒ then, maybe error.
  So, $\frac{\#SV}{N}$ is an upperbound on the error

• If ∃ x, ∄ o are present.

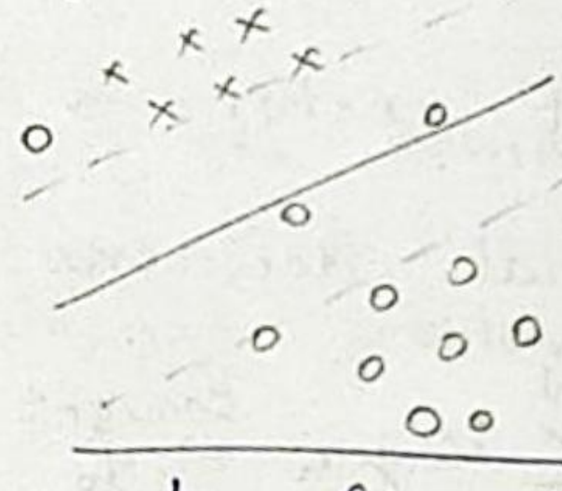


why do you need other SVs?

Ans: It is in 2D, so it looks simple to you

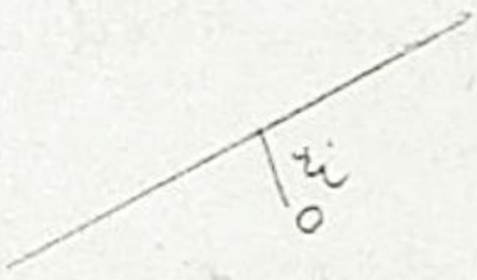
↓

Explanation to this stupid ans:



• If suppose the classification looks like this: (Samples can never be linearly separable (separable with a line)

We want ξ_i to be minimized



$$\boxed{w^T x_i + b \geq +1 - \xi_i \text{ for } y_i = +1} \quad \xi_i \geq 0$$

So minimize $\frac{1}{2} w^T w_i + \sum_{i=1}^{N} \xi_i$

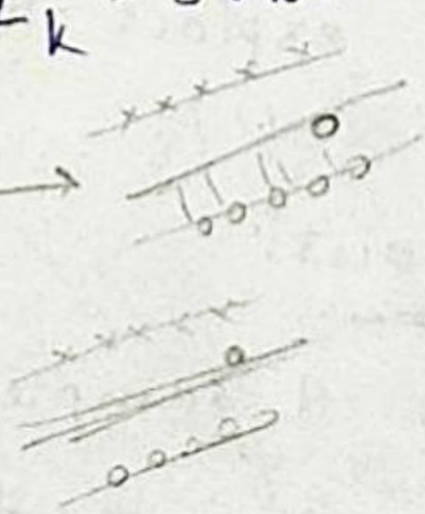
• Prev. prob: Hard margin SVM,
This prob: Soft margin SVM

• Append 1 $\frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$

If C = 'small ⇒ anyone can violate

C needs to be optimized

• Append 2 $\frac{1}{2} w^T w + C \times \sum_{i=1}^{N} \xi_i^k \rightarrow L_k - SVM$

. Minimum number of penality → 

. Minimum amount of penalty

for Some k

↳ strictly convex → ⌴

for others ↳ convex → ∪

Back to primal

$$\frac{1}{2}w^Tw - \sum_{i=1}^{N} \alpha_i [y_i(w^Tx_i)]$$

$$J(w, b, \alpha) = \frac{1}{2}w^Tw - \sum_{i=1}^{N} \alpha_i [y_i(w^Tx_i + b) - 1]$$

2 things $\Rightarrow$ $\alpha_i = 0$ or // Not on wall

$y_i(w^Tx_i + b) = 1 \Rightarrow$ // On wall

$\frac{\partial J}{\partial w} = 0, \quad \frac{\partial J}{\partial b} = 0, \longrightarrow$ Give, $\sum_{i=1}^{N} \alpha_i y_i = 0$.

$\hookrightarrow$ Gives $w = \sum_{i=1}^{N} \alpha_i y_i w^T x_i$

$$J = \frac{1}{2}\left[\sum_{i=1}^{N} \alpha_i y_i x_i^T\right]\left[\sum_{j=1}^{N} \alpha_j y_j x_j\right] - \sum_{i=1}^{N} \alpha_i [y_i w^T x_i] -$$

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

$\sum \alpha_i y_i = 0$

$$+ \sum_{i=1}^{N} \alpha_i$$

$$= \frac{1}{2}\left[\sum_{i=1}^{N} \alpha_i y_i x_i^T\right]\left[\sum_{j=1}^{N} \alpha_j y_j x_j\right] - \sum_{i=1}^{N} \alpha_i [y_i w^T x_i] + \sum_{i=1}^{N} \alpha_i$$

Substitute w

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

$\frac{1}{2}L - L$

$= -\frac{1}{2}L$

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

27/9/18

Lec-15

Kernels and Non-linear SVM's

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

If $d < N$

and $\sum\sum \alpha_i \alpha_j y_i y_j x_i^T x_j$ $= \sum_{i=1}^{N} \alpha_i w^T x$ $\rightarrow O(d)$

If $w^T x$ is already known

computation

Else $\sum\sum \alpha_i \alpha_j y_i y_j x_i^T x_j$ $\rightarrow d \times \#SV$.

$X \Rightarrow X' = MX$ Linear dimensionality reduction

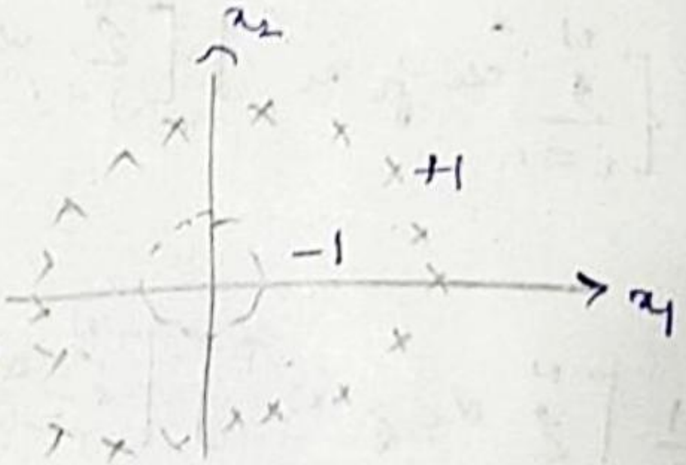
- Kernels do a transformation

$x \rightarrow \phi(x)$ generally, it increases dimensionality

↓

Some non-linear mapping

A linear hyperplane is not separating.

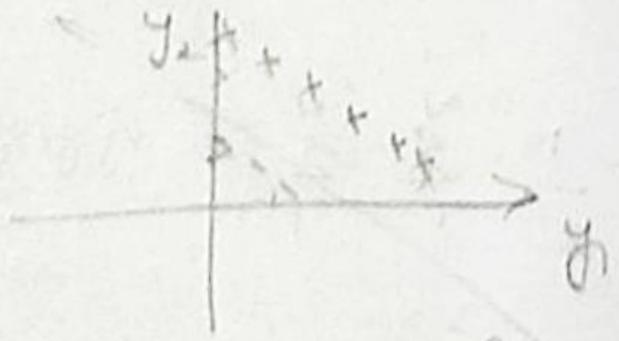


Toy example

But a circle can do it beautifully

Do a feature transform

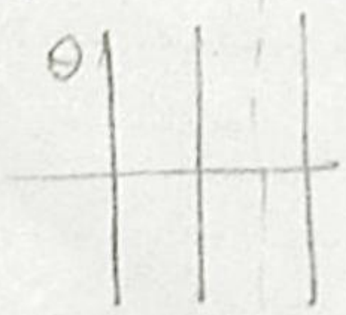
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{matrix} y_1 \\ y_2 \\ y_3 \end{matrix} \begin{bmatrix} x_1^2 \\ x_2 \\ x_2^2 \end{bmatrix}$$



Separable

- If we knew, how to transform, Kernels can do stuff, Linear Classifiers can never do

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



→ So only r can seperate

- We would like to map 2D space to ∞D space

$$P = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}, \quad Q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

$$P \cdot Q = P^T Q = p_1 q_1 + p_2 q_2$$

Apply $\phi(P) \Rightarrow \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2}\, p_1 p_2 \end{bmatrix}$;

$$\phi(Q) = \begin{bmatrix} q_1^2 \\ q_2^2 \\ \sqrt{2}\, q_1 q_2 \end{bmatrix}$$

$$\phi(P) \cdot \phi(Q) = \phi(P)^T \cdot \phi(Q) = p_1^2 q_1^2 + p_2^2 q_2^2 + 2 p_1 p_2 q_1 q_2$$

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

$$= (P^T Q)^2 \rightarrow \text{Homogenous Polynomial kernel}$$

we took $P^T Q$ and squared it
$\Rightarrow$ we got dot product $\phi(P) \cdot \phi(Q)$ to 3D.
without going

$$K(P, Q) = (P^T Q)^2$$

↓
Kernel function

$$K(P, Q) = \phi(P)^T \phi(Q)$$

Find ϕ so that $K(P, Q) = (P^T Q + 1)^2 \rightarrow$ Inhomogenous Poly

$$(p_1 q_1 + p_2 q_2 + 1)^2$$

$$p_1^2 q_1^2 + p_2^2 q_2^2 + 1 + 2 p_1 q_1 p_2 q_2 + 2 p_1 q_1 + 2 p_2 q_2$$

$$= \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2}\, p_1 p_2 \\ \sqrt{2}\, p_1 \\ \sqrt{2}\, p_2 \\ 1 \end{bmatrix} \begin{bmatrix} q_1^2 \\ q_2^2 \\ \sqrt{2}\, q_1 q_2 \\ \sqrt{2}\, q_1 \\ \sqrt{2}\, q_2 \\ 1 \end{bmatrix}$$

So 6D Kernel

$e^{-\gamma x^T y}$, $\tanh(x^T y) \rightarrow \underline{\underline{\infty D}}$ ['We generally don't care what these ∞ dimensions are]

- $\phi\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{bmatrix} p_1^2 \\ p_2^2 \\ p_1 p_2 \\ p_2 p_1 \end{bmatrix}$

$\Rightarrow$ $p_1 q_1 + p_2 q_2$

$p_1^2 q_1^2$
$p_2^2 q_2^2$
$p_1^2 p_2^2 q_1^2 q_2^2$
$p_1^2 p_2^2 q_1^2 q_2^2$

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

So many ϕ same Kernel [Faisakk!]

Kernel

if K is PSD, $\exists\ \phi$

$K = \begin{bmatrix} k(x_i, x_j) \end{bmatrix}$ NXN

- RBF: A Kernel that transforms to ∞ space, can be used for most problems

Kernel needn't be a valid vector

$K_2 = K_1() + K_2()$

$\sum_{i=1}^{n} d_i K_i()$

Curse of dimensionality:-

It is bad, to use more dimensions

But Kernels use only dot prods, so no problem

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \underbrace{K(x_i, x_j)}_{\phi(x_i)^T \phi(x_j)}$$

$w^T x + b$
$w = 0$
$b = \sum \alpha_i y_i$

$$\text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x_j) + b \right)$$

$$\text{Sign}(w^T x)$$

$\text{sign}(w^T x + b)$

- while testing (α_i, x_i) are to be carried
  i.e SV

If we are going to calculate linear SVM
α_i, x_i data . store only w.
we can remove

- If we know a feature map ,