Lecture-7

(4) Posing the optimal margin classifier Problem:

$$\max \gamma \quad \text{s.t} \quad \frac{y^{(i)}(w^T x^{(i)} + b)}{\|w\|} \geq \gamma$$

- $\max \gamma$ ← maximize the worst case geometric margin
- $\frac{y^{(i)}(w^T x^{(i)} + b)}{\|w\|} \geq \gamma$ ↙ "every example has geometric margin atleast γ"

(4) $y = g(w^T x + b)$

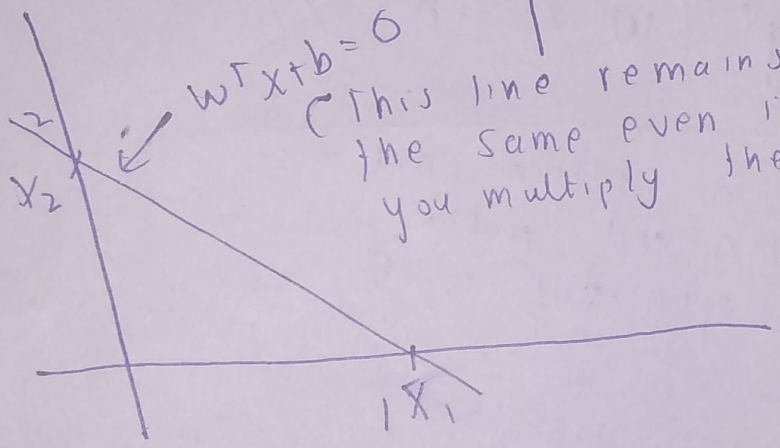
$= g\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}^T x + b\right) \rightsquigarrow$

(*) Example of a classifier

$$= g\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}^T x + b - 2 \right) \} \; g(w^T x + b)$$

You can scale up the values of w and b, in this case by 10.

$$= g\left(\begin{bmatrix} 20 \\ 10 \end{bmatrix}^T x - 20 \right)$$



(*) You can multiply the equation $g(w^T x + b)$ with any constant without changing the decision boundry.

(*) Now Returning back:-
→ lets say you choose to normalize the equation by $\|w\| = \frac{1}{\gamma}$

then:-

$$\max \frac{1}{\|w\|}$$

$$\text{s.t } y^{(i)}(w^T \cdot x^{(i)} + b) \geq \gamma$$

~~Max~~ $\min \frac{1}{2}\|w\|^2$

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

(*) Suppose $w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$ (w can be written as a linear combination of training examples).

(*) Why? (Reasonable Assumption above):

→ Intuition #1:

- For logistic Regression

$\theta_i = 0$

Gradient descent:

$\theta = \theta - \alpha (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$.

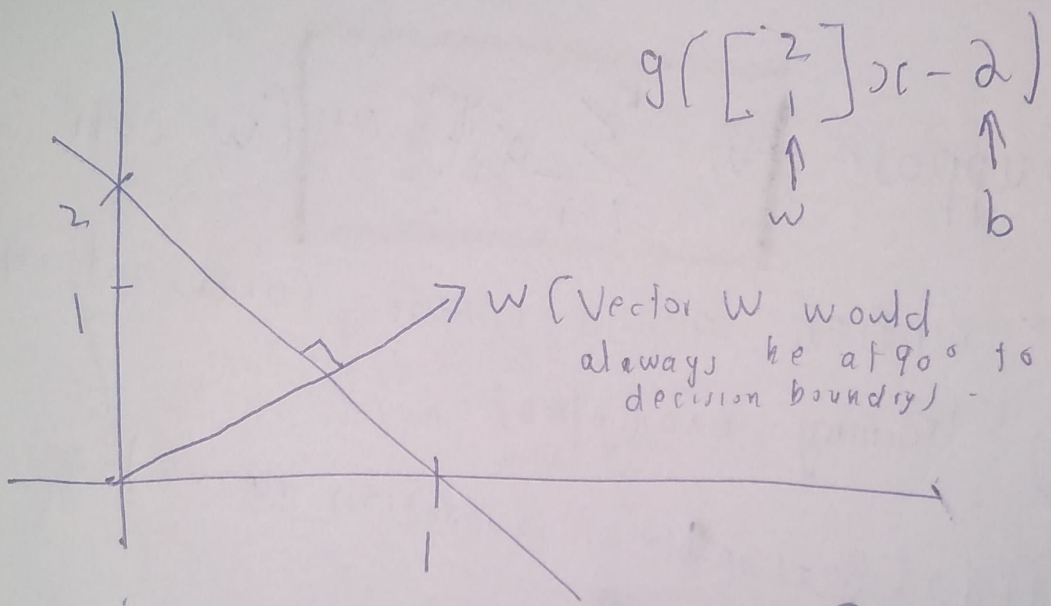
→ θ is a linear combination of your training example for logistic regression. (Proof by induction).

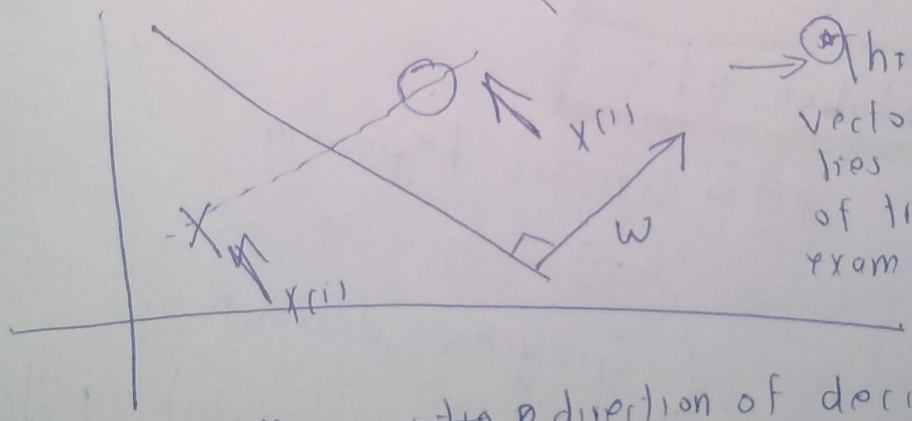
- (True for Batch Gradient Descent).

(*) You can optimize SVM by gradient descent and see like logistic regression

Parameter are linear combination of training examples.

→ **Intuition 2** :-

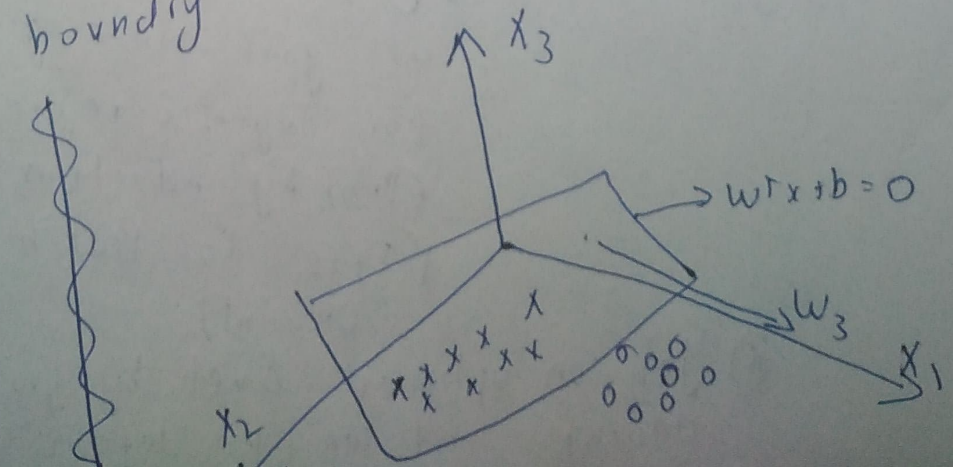




→ ⊛ This show vector w lies in span of training example

⊛ w set the ~~pos~~ direction of decision boundry while b shifts the decision boundry

→



→ The above diagrams shows that our training examples lies in three dimensional space but each training example third dimension is zero. The plane has been show and vector normal to plane would be w which would also have third dimension as zero like training example which shows we given training examples.

→ $w = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} x^{(i)}$ (Plugging in below equations)

$\min_{w,b} \frac{1}{2} \|w\|^2$ → $w^T \cdot w$

s.t $y^{(i)} [w^T x^{(i)} + b] \geq 1$.

$i = 1, \ldots m$

→ $\min_{w,b} \frac{1}{2} \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T \left(\sum_{j=1}^{m} \alpha_j y^{(j)} x^{(j)} \right)$.

$= \min \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \underbrace{x^{(i)T} x^{(j)}}_{\langle x^{(i)}, x^{(j)} \rangle}$

$\langle x, z \rangle = x^T z$ is inner product

(*)

s.t $y^{(i)}\left(\sum_j \alpha_j y^{(j)} x^{(j)}\right)^T x^{(i)} + b\right) \geq 1.$

s.t $y^{(i)}\left(\sum_j \alpha_j y^{(j)} \langle x^{(j)}, x^{(i)} \rangle + b\right) \geq 1.$

→ So Our **updated equation** are:-

$$\min \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle$$

s.t $y^{(i)}\left(\sum_j \alpha_j y^{(j)} \langle x^{(j)}, x^{(i)} \rangle + b\right) \geq 1.$

- Feature vectors only appear in the inner product. But you can compute inner product without actually mapping feature vector to higher dimension.

(*)

e) You further simplify the above equations to the following (Simplification is complicated)

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

s.t $\alpha_i \geq 0$ "Dual optimization problem"

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

1) Solve for α_i's, b

2) To make prediction compute

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

$$= g\left(\left(\underbrace{\sum_i \alpha_i y^{(i)} x^{(i)}}_{w}\right)^T x + b\right)$$

$$= g\left(\sum_i \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b\right)$$

↳ You can even make predictions only using inner products.

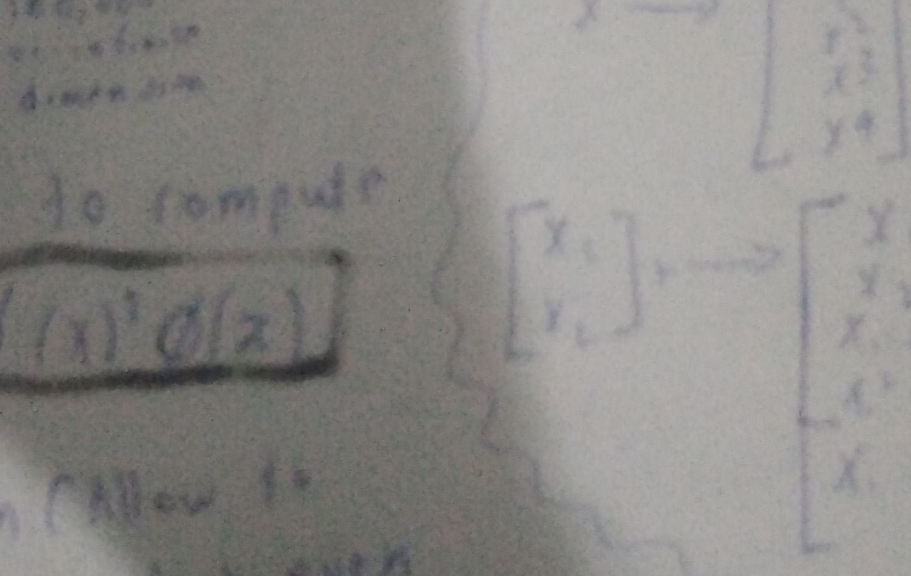
Kernels Trick

1) Write algorithm in terms of $\langle x^{(i)}, x^{(j)} \rangle$ or $\langle x, z \rangle$

2) Let there be some mapping from

$x \longrightarrow \phi(x)$ → higher dimensional

2D



3) Find way to compute

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

↓

Kernel function (Allow to compute dot product even when $\phi(x)$ & $\phi(z)$ are in very high dimensions)

4) Replace $\langle x, z \rangle$ in algorithm with $K(x, z)$.

⑧ Examples of kernel

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad x \in \mathbb{R}^n$$

$$\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_1 \\ x_2 x_2 \\ x_2 x_3 \\ x_3 x_1 \\ x_3 x_2 \\ x_3 x_3 \end{bmatrix} \quad \phi(x) \in \mathbb{R}^{n^2}$$

$$\phi(z) = \begin{bmatrix} z_1 z_1 \\ z_2 z_2 \\ z_1 z_3 \\ z_2 z_1 \\ z_2 z_3 \\ z_2 z_1 \\ z_2 z_1 \\ z_2 z_3 \end{bmatrix} \quad \phi(z) \in \mathbb{R}^{n^2}$$

→ Now because there are n^2 elements you need $O(n^2)$ time to compute $\phi(x)$, or $\phi(x)^T\phi(z)$ explicitly

→ $K(x,z) = \phi(x)^T\phi(z)$

$= (x^Tz)^2$ → This is an $O(n)$ time operation

$x \in \mathbb{R}^n$, $z \in \mathbb{R}^n$

→ Let me proof this statment

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

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

x^Tz $\quad$ x^Tz

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

$= \sum_{i=1}^{n}\sum_{j=1}^{n} (x_i x_j)(z_i z_j)$ → Dot product of the vector which we mapped into higher dimension by polynomial pairs

$(x_i x_j) \rightarrow \phi(x)$ → Polynomial pairs

$(z_i z_j) \rightarrow \phi(z)$ → Polynomial pairs

→ We have been to compute the higher dimension dot product $\phi(x)^T \phi(z)$ by remaining in lower dimension through the kernel trick or known by the example (Computationally Efficient).

→ Other Examples of Kernel

② $K(x,z) = (x^T z + c)^2$

$c \in \mathbb{R}$

This correspond to modifying your features as follow

$\phi(x) =$ [$x_1 x_1$, $x_1 x_2$, $x_1 x_3$, $x_2 x_1$, $x_2 x_2$, $x_2 x_3$, $x_3 x_1$, $x_3 x_2$, $x_3 x_3$, $\sqrt{2c}\, x_1$, $\sqrt{2c}\, x_2$, $\sqrt{2c}\, x_3$, c]

→ Some change to $\phi(z)$.

→ So this example $K(x,z)$ can compute the dot product of transformed $\phi(x)$ & $\phi(z)$ without any mapping

→ ~~3- $K(x,z) = (\phi^T z + c)$~~

→ 3- $K(x,z) = (x^T z + c)^d$ → $O(n)$ time complexity

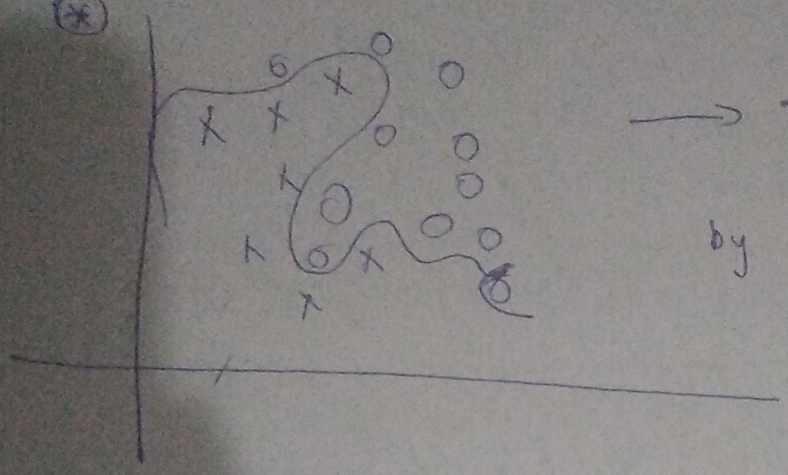
↳ This corresponds to taking dot product of

- $\phi(x)$ and $\phi(z)$ has all $\binom{n+d}{d}$ features of monomial upto order d

→ So SVM is the optimal margin classifier and applying kernel trick to it

→ SVM = Optimal Margin Classifier + Kernel Trick

(*)



→ This a linear boundry learned by SVM in high-dimensional space but when you project back to original space it becomes non-linear.

(*) How to make kernels:

→ If x, z are "similar" $K(x,z) = \phi(x)\phi(z)$

is large

- If x, z are "dissimilar", $K(x,z)$ is small

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

→ You can see that kernel are basically similarity mapping function so $K(x,z)$ basically takes this idea where if x and z are similar its value would be high or else vice-versa. (Satisfies the criteria)

- You can use the Kernel function such as the one suggested above only if there exist ϕ s.t $K(x,z) = \phi(x)^T \phi(z)$

→ Also $K(x,x) = \phi(x)^T \phi(x) \geq 0$

↳ Dot product is also always greater than zero

→ Proof to outline when Kernel is valid

Let $\{x^{(1)} \dots x^{(d)}\}$ any d points

Let $K \in R^{d \times d}$ → "Kernel matrix"

$K_{ij} = K(x^{(i)}, x^{(j)})$ → Kernel applied to two of those points. (Every pair of point applied with Kernel and put into the matrix)

(b) Given any vector z.

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

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

~~(if k=n)~~

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

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

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

which proves $K \geq 0$ (K is positive semi-definite matrix).

→ Theorm Mercer: K is a valid Kernel function (ie ∃ ∅ st

$K(x,z) = \phi(x)^T \phi(z)$ if and only if for any d points $\{x^{(1)}, \ldots x^{(d)}\}$ the corresponding Kernel matrix $K \geq 0$ (Positive semi-definate).

↳ The theorm also stands in a reverse way ie if $K \geq 0$ then $K(x,z) = \phi^T(x)\phi(z)$ is a valid Kernel

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

↓

Gaussian Kernel

* Different types of Kernel

Linear Kernel

$$K(x,z) = x^T z$$

$$\phi(x) = x$$

Gaussian Kernel

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$ ↔ This also uses all polynomial features

$$\phi(x) \in \mathbb{R}^{\infty} \text{ (Infinite dimension)}$$

- **Polynomial Kernel**

$$K(x,z) = (x^T z + c)^d$$

$$\phi(x) \in \mathbb{R}^{\binom{n+d}{d}} \text{ features}$$

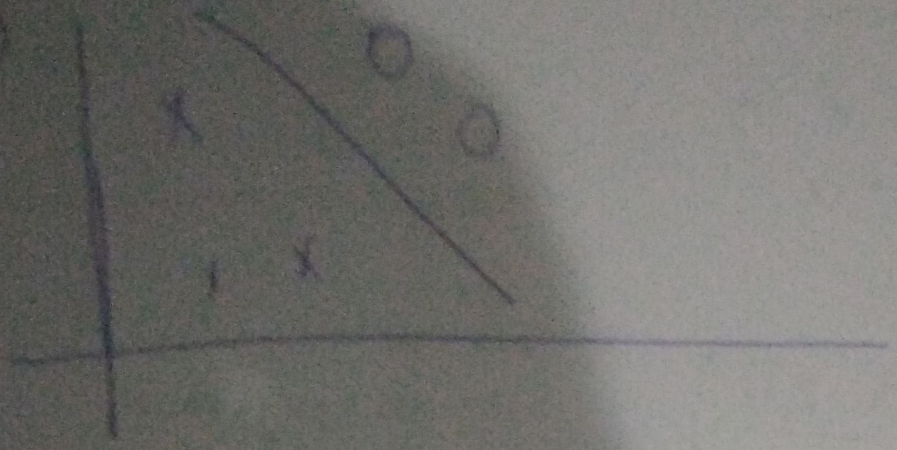
(*) Kernel Trick is more general then SVM :-

→ You can apply Kernel Trick to any training algorithm if it can be rewritten in terms of $\langle x^{(i)}, x^{(j)} \rangle$ and then replace it with Kernel function

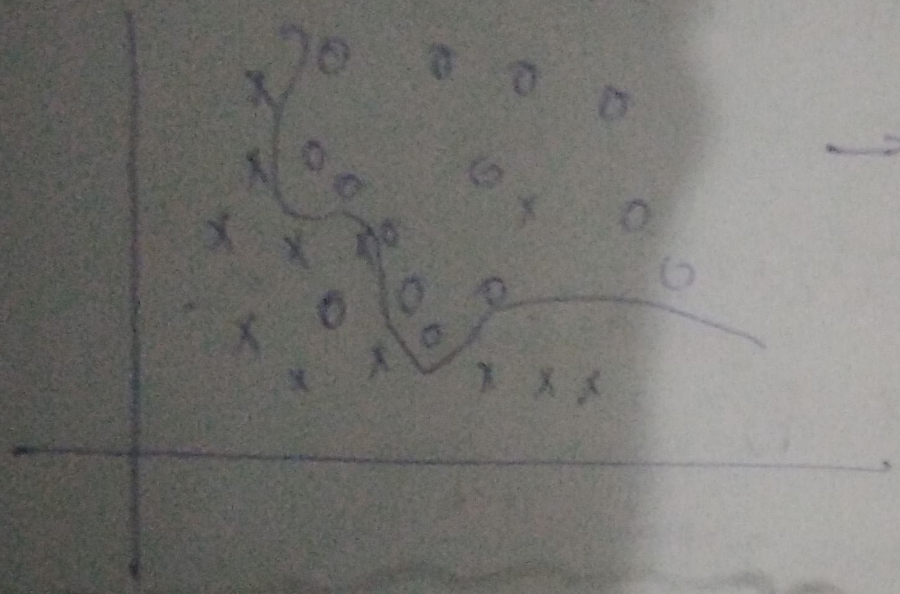
→ All the ~~Kernel functions~~ Discriminative algorithm learned so far can take advantage of Kernel tricks

(8)

(4)



(5) If your dataset is noisy



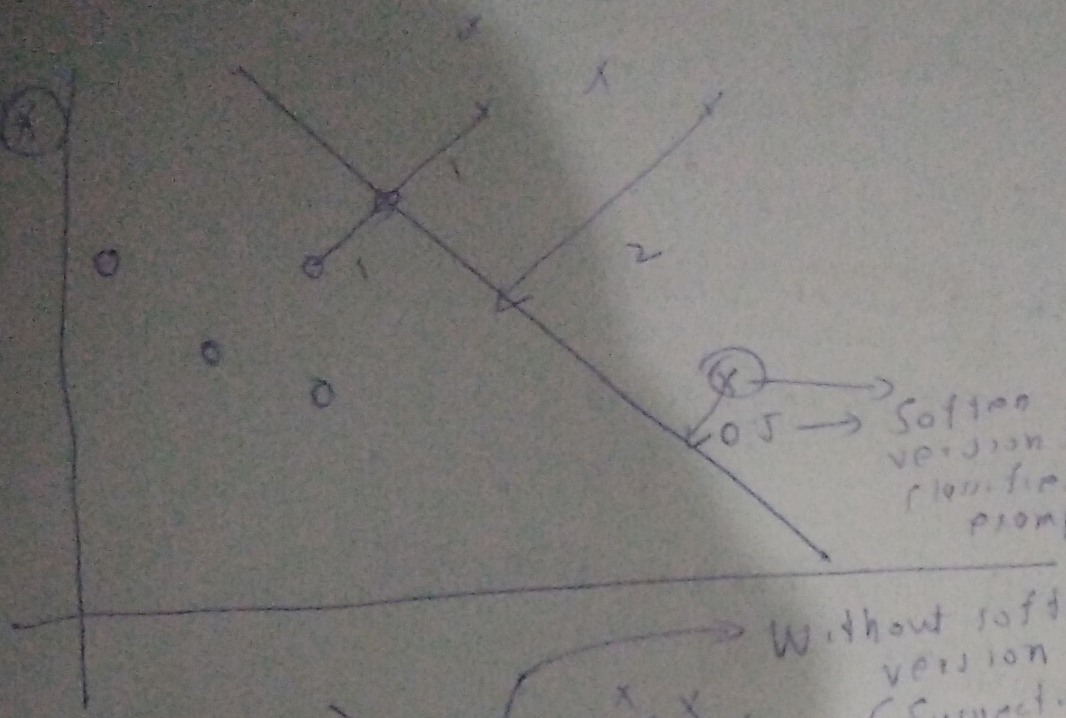
→ You don't want a very strict complicated decision boundary but rather a more soften one so it generalizes better (AKA You don't want a zero training error).

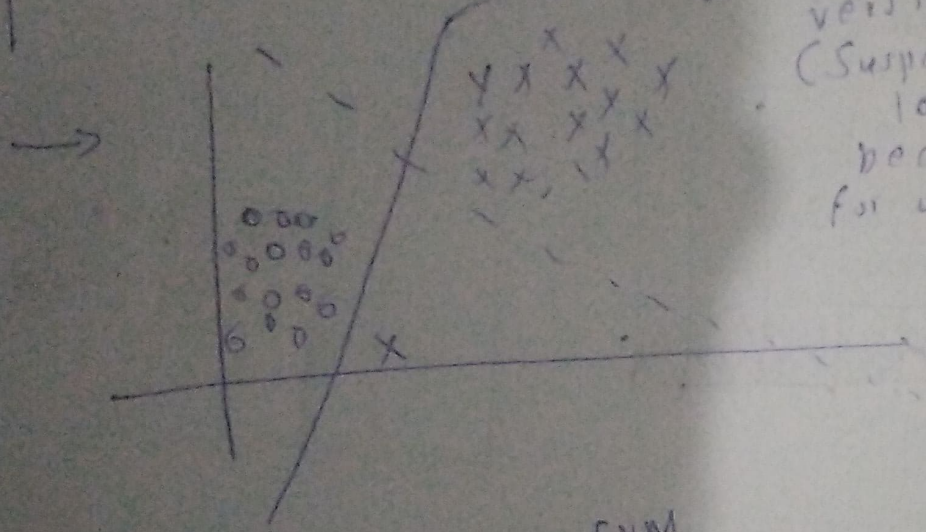
(6) L_1 norm soft margin SVM:

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

$$\text{s.t. } \underbrace{y^{(i)}(w^T x^{(i)} + b)}_{\text{functional margin}} \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad (i = 1, \ldots, m).$$





Without soften version
(Susceptible to outlier because it optimizes for worst case marginal

↳ Soften version
(More Robust)

→ L_1 norm soft margin SVM simplifies to the following:

$$\max \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m}\sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

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

$$\boxed{0 \leq \alpha_i \leq C}$$ → This the change from previous optimization of Hard SVM.

Application of SVM

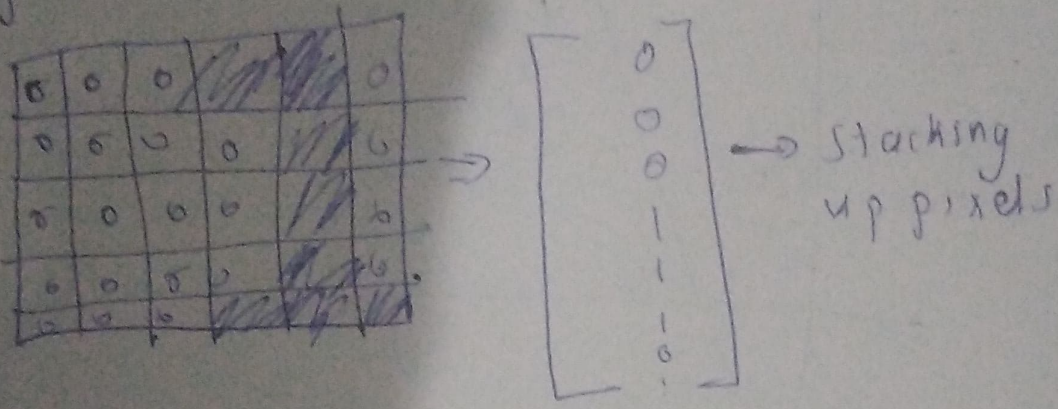
① SVM with **Polynomial kernel**:

$$K(x,z) = (x \cdot z)^d$$

or with **gaussian kernel**:

$$= \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right).$$

Works well with hand written digit recognization (classifier).

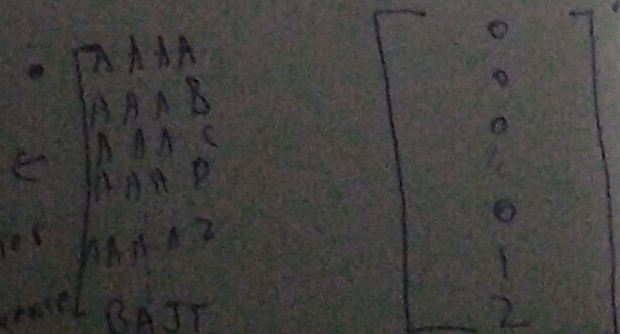


② Protien Sequence classifier:

A .. Z → Denocde amino acid who combination define protien.

- BAJIIAIBAJTIAU → Protien

$\phi(x) = ?$



List of four characters protien sequence ∈

→ Turns out instead of computing
$\phi(x)$ which is d^4 dimensional vector
and consequently computationally
expensive to do so ⇒ you can compute
Kernel through dynamic programming i.e
$\phi(x)^T \phi(z) = K(x, z)$ and then you use
support vector machine
↳ Example of innovative
Kernel.