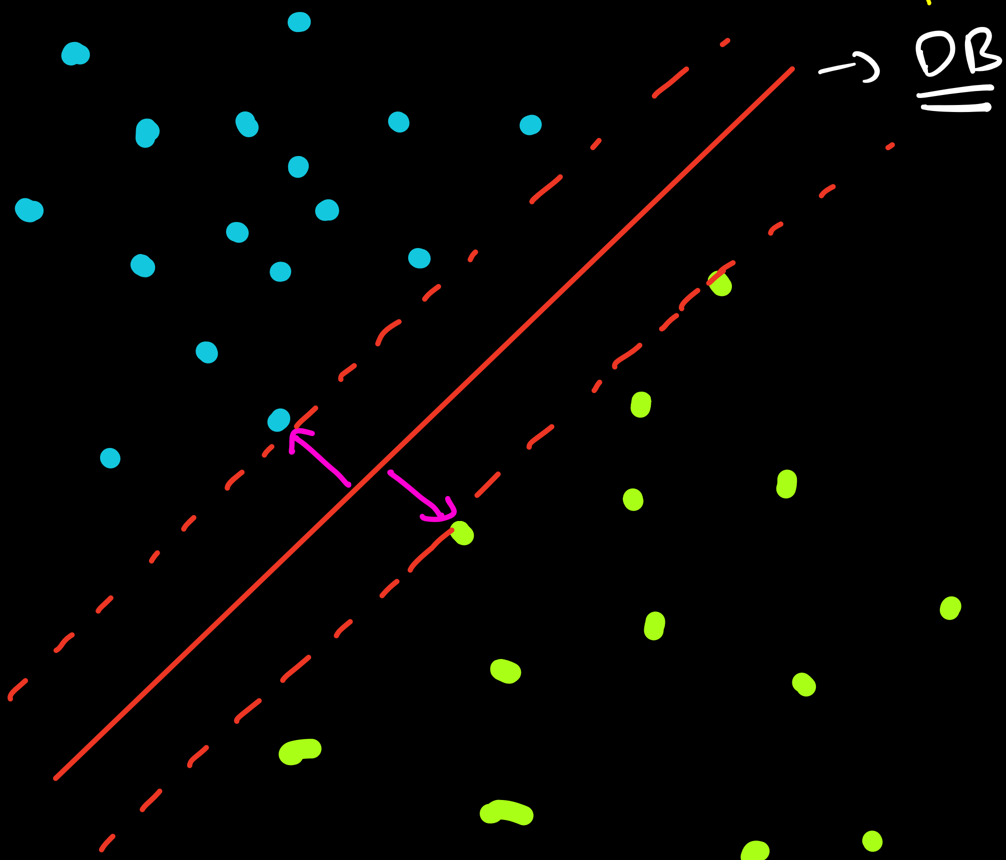
Support Vector Machines

[ML-1]

- → Recap SVM-1
- → Dual form
- → Kernel Trick

Recap



$w^T x_i + w_0 = 0$

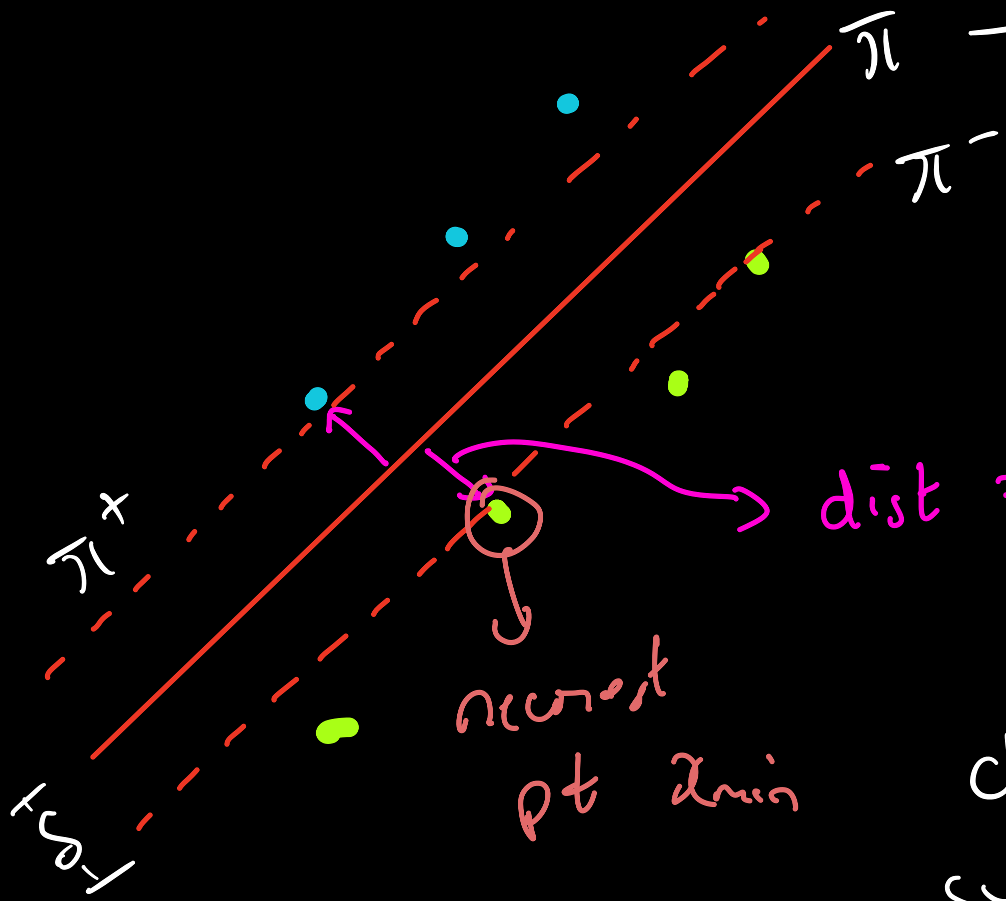
→ DB (linear)

(perfectly separable

DB should be as far as possible from either of the nearest point

Maximise margin

→ We take nearest points because they deserve to have a say. Further pts, may have outliers.



$\pi \longrightarrow w^T x_i + w_0 = 0$

$w^T x_i + w_- = 0$

$$\text{dist} = \frac{|w_0 - w_-|}{||w||}$$

change $||w||$ and w_0 such that $|w_0 - w_-| = 1$

Change s.t. min dist,

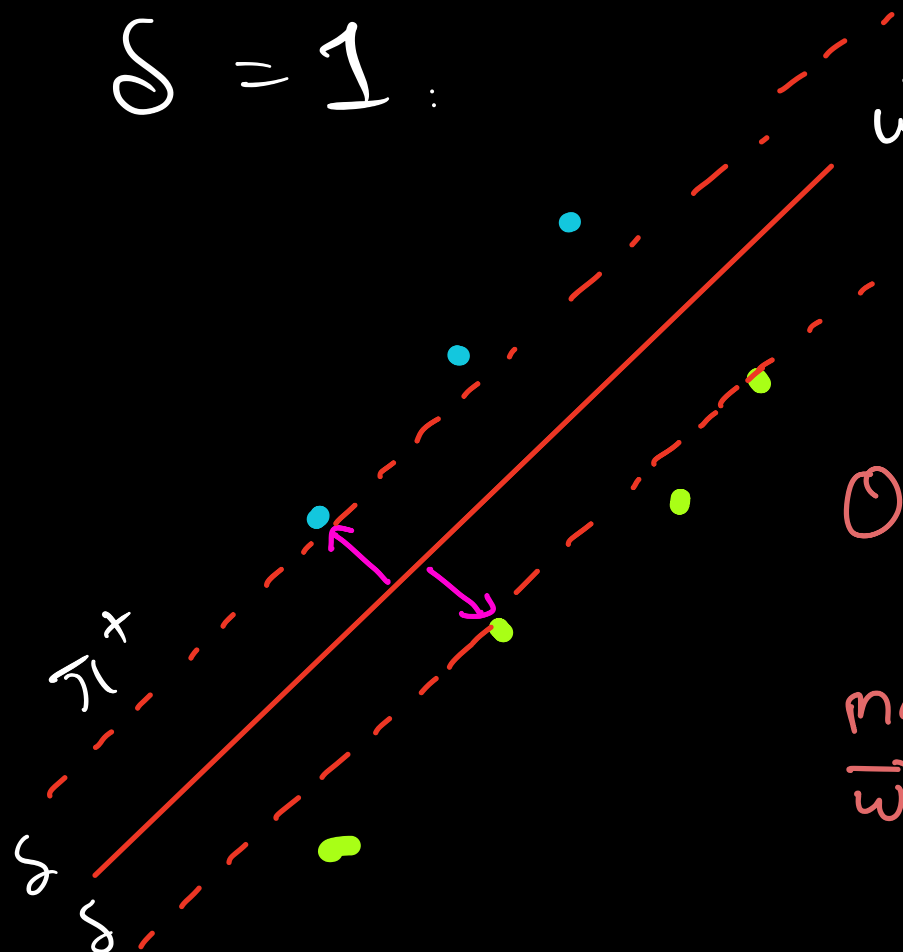
i.e $\delta = 1$.

For simplicity

$w^T x + (w_0 + 1) = 0$

$w^T x + w_0 = 0$

$w^T x + (w_0 - 1) = 0$



Objective:

$$\max_{\vec{w},\, w_0} : \frac{2}{\|w\|}$$

Hard Margin SVM

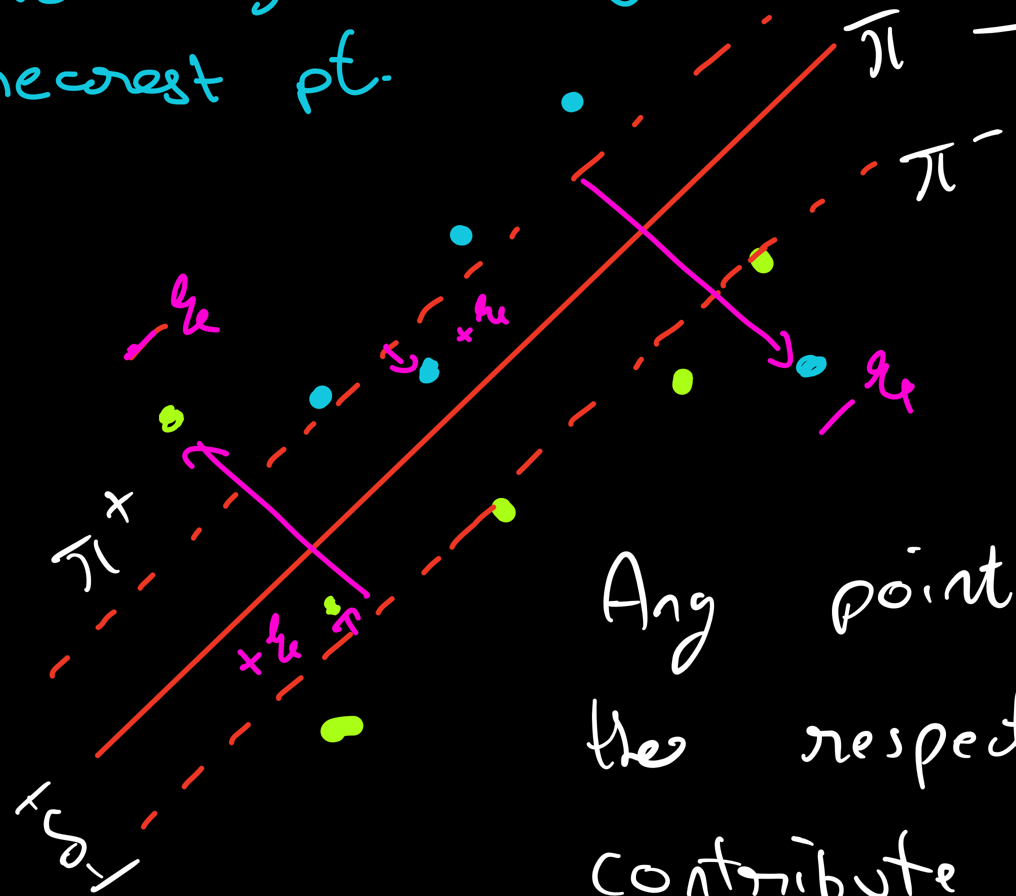
$$\max_{\vec{w}, w_0} \frac{2}{\|w\|} \; ; \text{ s.t. } (w^T x_i + w_0) y_i \geq 1 \quad \forall i : 1 \to n$$

But, this does not work if there is any mix / overlap of classes.

$$= \min_{\vec{w}, w_0} \frac{\|w\|}{2}$$

Soft Margin SVM

No longer taking nearest pt.



Here we care about dist from margin lines, not DB.

Any point which is outside the respective margin line will contribute in computing the DB.

$\xi > 0$ correctly classified support vectors

$\xi < 0$ incorrectly classified support vectors

$$\min_{\vec{w},\, w_0,\, \xi_i} \frac{\|w\|}{2} + C \sum_{i=1}^{n} \xi_i$$

n → only support vectors

$\frac{\|w\|}{2}$ → max margin

$\sum \xi_i$ → min crossing of margin

max margin ⟷ min crossing of margin

balance

$$\text{st:} \quad (w^T x_i + w_0)\, y_i \geq 1 - \xi_i$$

$$\xi_i > 0$$

$$\forall i : 1 \rightarrow n$$

Solving it in this form is possible, however converting it to a dual form would make it possible to introduce non-linearity

Dual Form of Objective Func^n

Principal in optimisation

my problem ⟶ another equivalent problem in a diff universe

→ solve this

$$\max_{\alpha_i} \quad \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{s.t} \quad \left.\begin{array}{l} 0 \le \alpha_i \le c \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{array}\right] \forall i : 1 \to n$$

Here,

prediction eq^n : $\rightarrow$ $f(x_q) = \underbrace{\sum_{i=1}^{n} \alpha_i y_i x_i^T x_q}_{\text{support vectors}}$ query point

Observe that $\vec{x}$ never appears alone anymore!

its always $x_a^T x_b$

$\alpha_i = 0 \rightarrow$ non support vect

$\alpha_i > 0 \rightarrow$ support vect

Full Picture

Man Maximisation

$\downarrow$

Soft margins $\rightarrow \xi_i$

$\downarrow$

Optimisation

$\downarrow$

Dual $\rightarrow \alpha_i$

Kernel SVM

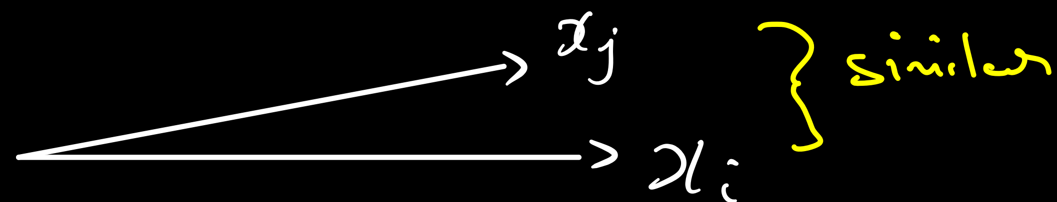
$$\max_{\alpha_i} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \underbrace{x_i^T x_j}_{\text{Dot product}}$$

Let us re-look this!

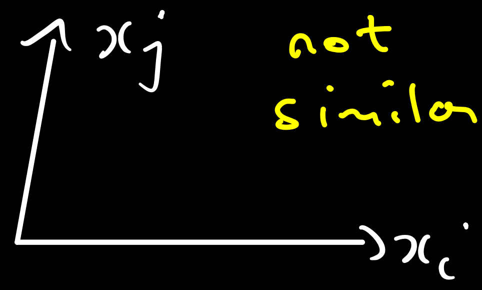
"Dot product" can be interpreted as similarity

Mathemetitians realised that they can change this to any notion of similarity

- → cosine
- → custom ..
- → etc..



Now we can write:



$$\sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i \alpha_j y_i y_j \underbrace{K(x_i, x_j)}_{\text{Kernel Function}}$$

→ predict func^n : $\sum_{i=1}^{n} \alpha_i y_i K(x_i, x_q)$

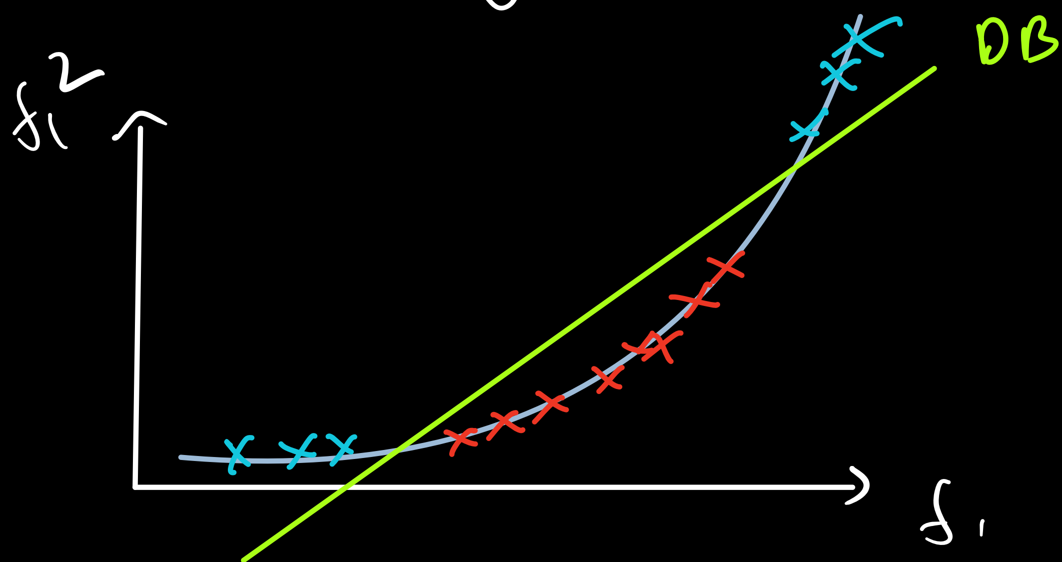
The beauty here is that the kernel function helps provide a convinient place to introduce non-linearity.

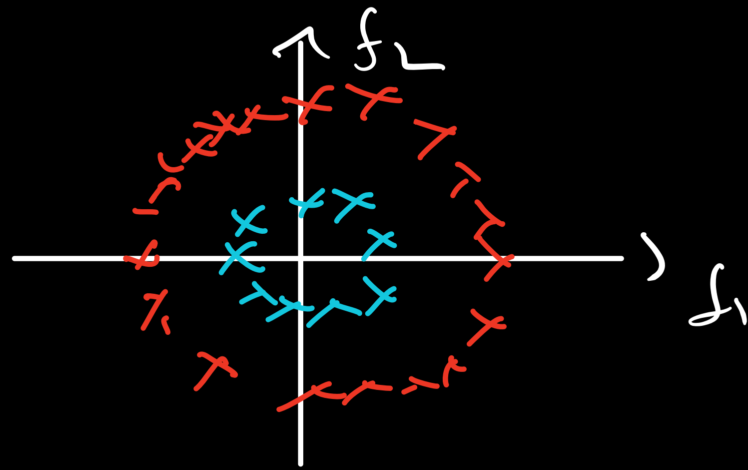
Examples:

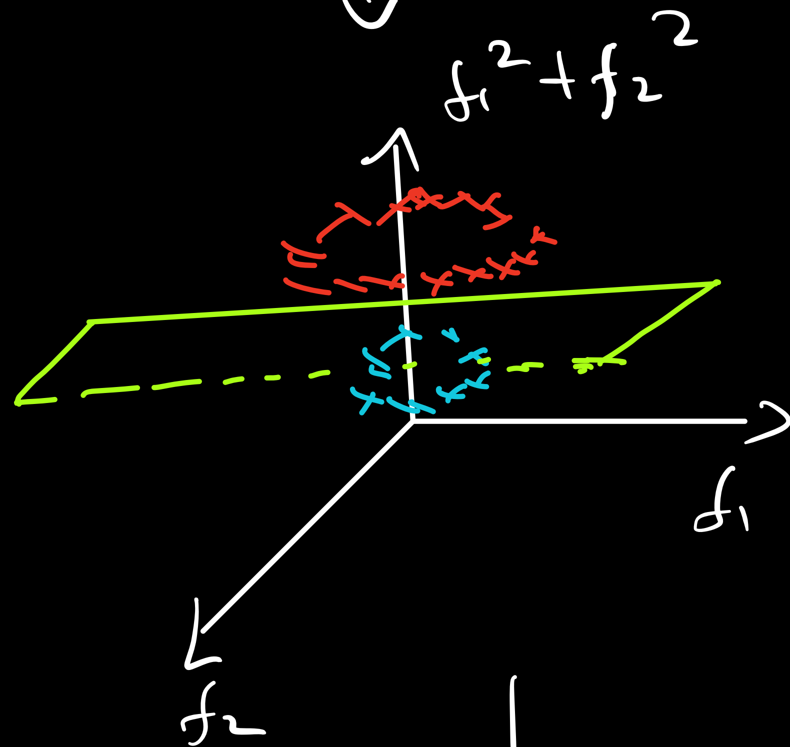
Low - Dim

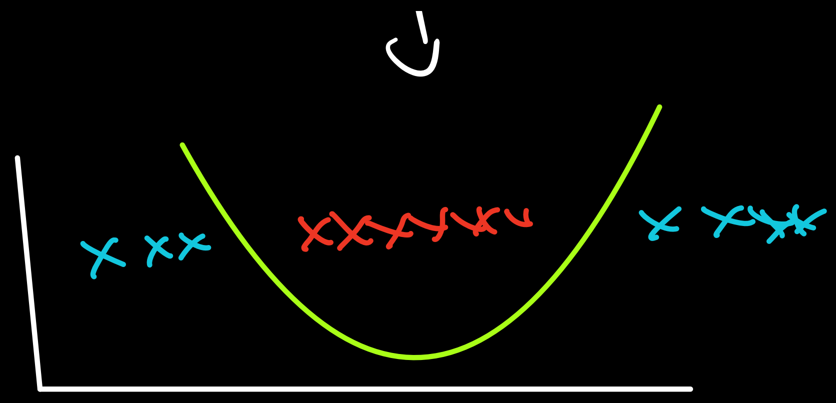
not seperable

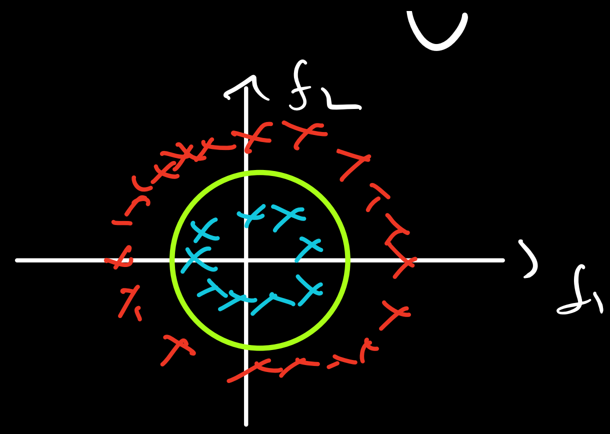
High - Dim











So kernels may be used to create more features (non-linear) where SVM can now try a linear decision boundary in a **high dimensional space**.

When we come back to original space we get a **non-linear DB**

How does kernel accomplish this?

Polynomial Kernel

$$K(x_i, x_2) = (x_1^T x_2 + 1)^m$$

m = degree of polynomial

Eg: $m = 2$ (quadratic)

$$\vec{x}_1 = \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix}, \quad \vec{x}_2 = \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix}$$

$$\therefore K(x_1, x_2) = \left(\begin{bmatrix} x_{11} & x_{12} \end{bmatrix} \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix} + 1 \right)^2$$

$$= (1 + x_{11} x_{21} + x_{12} x_{22})^2$$

$$(a+b+c)^2 = a^2 + b^2 + c^2 + 2ab + 2bc + 2ca$$

$$= 1 + x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}x_{21} + 2x_{12}x_{22}$$
$$+ 2x_{11}x_{21}x_{12}x_{22} \quad — \quad A$$

Now consider:

$$x_1' = [1, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}, \sqrt{2}x_{12}, \sqrt{2}x_{11}x_{12}]$$
$$x_2' = [\underbrace{1}_{\substack{1 \\ \text{bias}}}, \underbrace{x_{21}^2, x_{22}^2}_{\substack{x_i^2 \\ \text{non-linear}}}, \underbrace{\sqrt{2}x_{21}, \sqrt{2}x_{22}}_{\substack{x_i \\ \text{linear}}}, \underbrace{\sqrt{2}x_{21}x_{22}}_{\substack{x_i x_j \\ \text{non-linear}}}]$$

Magic $\rightarrow$ $\vec{x}_1' \cdot \vec{x}_2' = A$ [2d $\rightarrow$ 6d]

So using this small $K(w_1, w_2)^m$ we are able to get all powers of original features and all multiplicative combinations

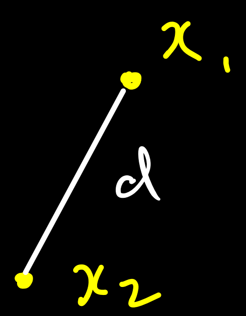
RBF Kernel → Most Popular

(Radial Basis Func^n / Gaussian)

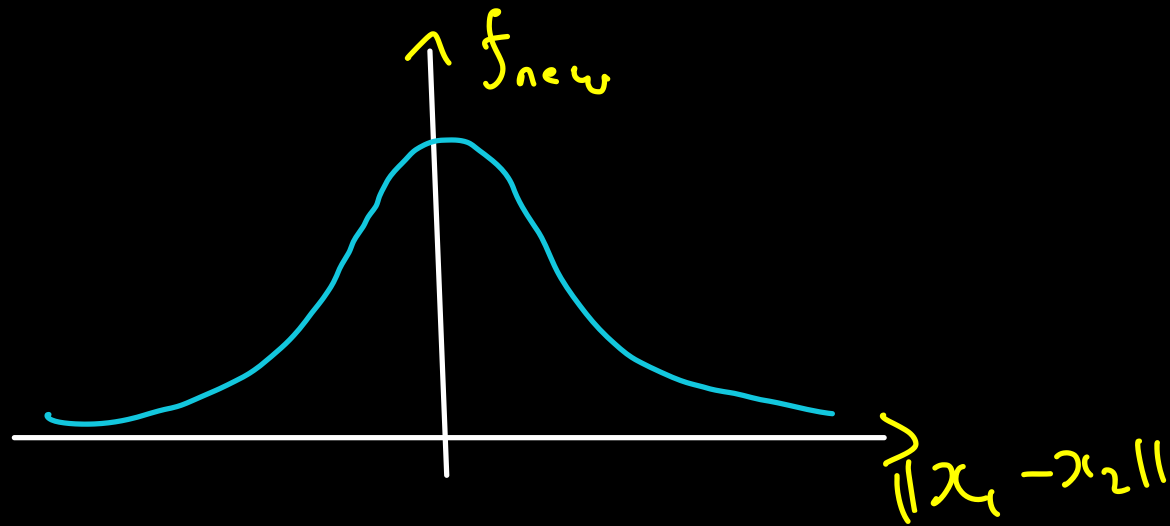
$$K_{RBF}(x_1, x_2) = \exp\left(\frac{-\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

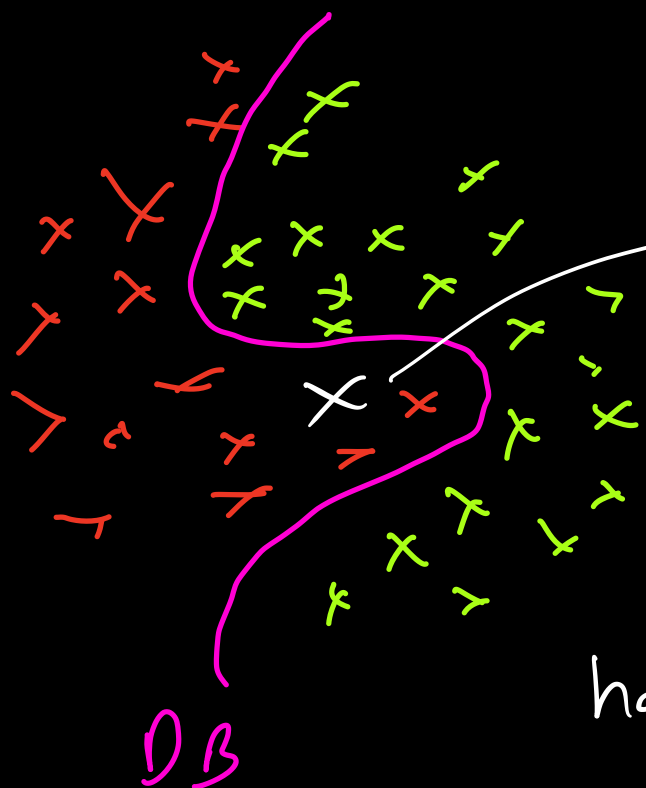
→ euclidean dist

$2\sigma^2$ → H.P



$$K_{RBF}(x_1, x_2) = e^{\frac{-d^2}{2\sigma^2}}$$





→ query point

→ dist b/w nearest points?)

→ close points have high f_{new}.

→ KNN behaviour

Similarity with KNN

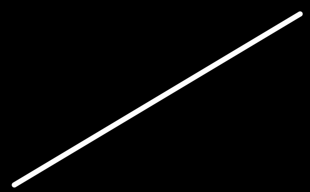
KNN $\rightarrow$ K $\uparrow$ underfit ; K $\downarrow$ } overfit

RBF $\rightarrow$ $\sigma \uparrow$ underfit ; $\sigma \downarrow$

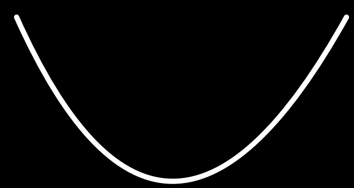
RBF $\sim$ Polynomial $(m \rightarrow \infty)$

How ?

$m = 1$



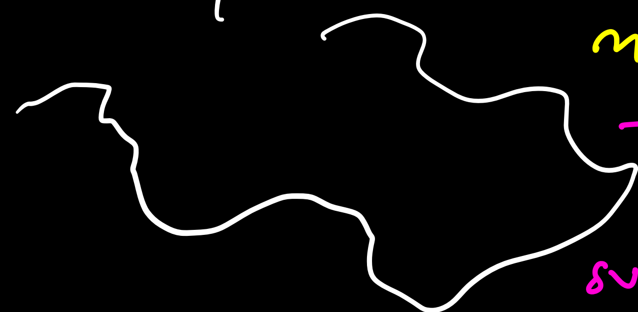
$m = 2$



$m = 3$

$m = 7$

$\rightarrow$



$m \rightarrow \infty$

$\rightarrow$ RBF can make such DB ✓