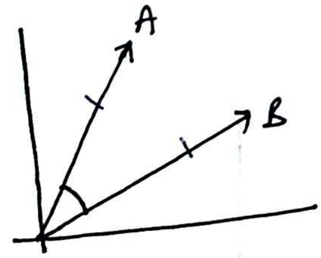
The Similarity Perspective

consine similarity $= \dfrac{A \cdot B}{\|A\| \, \|B\|}$



$\|A\| = 1$ $\quad \|B\| = 2$

$= \boxed{A \cdot B} \rightarrow$ dot product of A and B

* Find the similarity between two vectors we use consine similarity

$$\max_{\alpha_i} \underbrace{\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j}_{} \boxed{(x_i \cdot x_j)}$$

$\rightarrow$ similarity

$\rightarrow$ maximize the similarity of SV based on main sign

Kernel SVM

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \, K(x_i, x_j)$$

$\rightarrow$ Kernel SVM

$K(x_i, x_j) \rightarrow$ kernel $\rightarrow$ similar between x_i and x_j

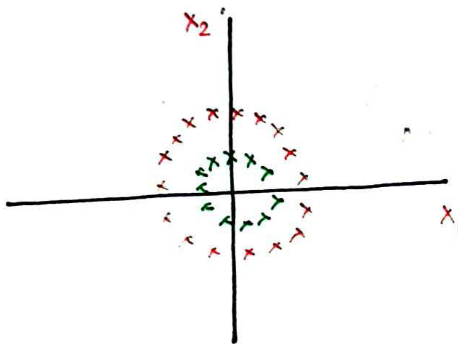
$\rightarrow \boxed{x_i \cdot x_j} \rightarrow$ Linear SVM

Polynomail

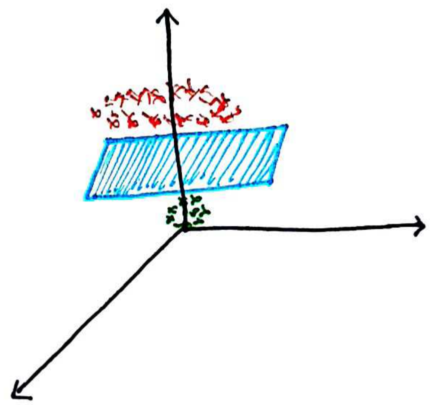
RBF

$\boxed{x_i \cdot x_j}$ $\rightarrow$ Creative Version

Polynomial Kernel



$\xrightarrow{x_1^2 + x_2^2}$



	x_1	x_2
$x_i \rightarrow$	x_{11}	x_{12}
$x_j \rightarrow$	x_{21}	x_{22}

$(x_{11}x_{21} + x_{12}x_{22})$

$$K(x_i, x_j) = (\gamma + x_i \cdot x_j)^d \qquad \gamma = 1 \quad d = 2, 3$$

$$= \boxed{1 + x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}x_{21} + 2x_{12}x_{21} + 2x_{11}x_{21}x_{12}x_{22}}$$

↳ Polynomial term

Kernal Trick

x_1 and x_2 not convert into other form (3d or 4d..)

x_1 and x_2 use Polynomial term in current form.

The trick

$$\boxed{1 + x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}x_{21} + 2x_{12}x_{22} + 2x_{11}x_{21}x_{12}x_{22}}$$

dot product of 2 Vector

$[1 \quad x_{11}^2 \quad x_{12}^2 \quad \sqrt{2}x_{11} \quad \sqrt{2}x_{12} \quad \sqrt{2}x_{11}x_{21}] \rightarrow$ 6d Vector x_i

$[1 \quad x_{22}^2 \quad \cancel{\sqrt{2}} x_{21}^2 \quad \sqrt{2}x_{21} \quad \sqrt{2}x_{22} \quad \sqrt{2}x_{12}x_{22}] \rightarrow$ 6d Vector $\rightarrow x_j$

1 method $\rightarrow$

$x_i (x_{11}\ x_{12}) \xrightarrow{ft} x_i\ (6d)$

$x_j (x_{21}\ x_{22}) \rightarrow x_j\ (6d)$

$\rightarrow x_i' \cdot x_j' \rightarrow$ expression

2 method $\quad x_i, x_j \rightarrow k(x_i, x_j) \rightarrow$ expression

* 1 method occupy much space to store the vector

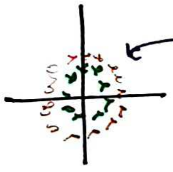
* ✓ 2 method not change into higher dimension just use polynomial term calculate expression.

let take 10 dimensional data if we solve with method 1 then occupy large space bcz change into higher dimension.

but in 2 method not change into higher dimension and not occupy any space.

What about the other Polynomial terms

$$1 + \underbrace{x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2}_{\text{Circular shape}} + \underbrace{2x_{11} x_{21} + 2x_{12} x_{22} + 2x_{11} x_{21} x_{12} x_{22}}_{\text{Other shape (Conic section)}}$$



RBF Kernel

Radial Basic function {Normal distribution}

- Popular
- Best out of the box kernel
- Powerful

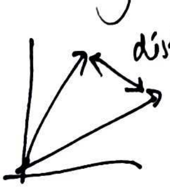
euclidian distance

$$k(x_i, x_j) = \left\{ e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}} \right\}$$

$k \propto \frac{1}{\text{distance}}$

↓

similarity



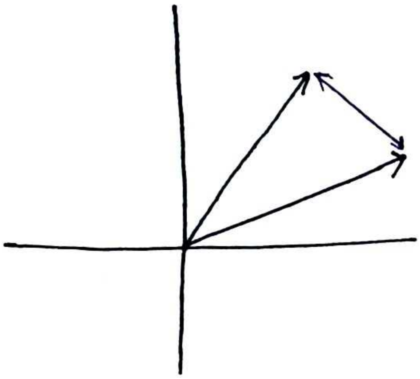
distance

$$e^{-\gamma \|x_i - x_j\|^2}$$

$\boxed{\gamma = 1/2\,\sigma^2}$

gamma

hyperparameter



$$k(x_i, x_j) = e^{-\frac{dist^2}{2e^2}}$$

$$k \propto \frac{1}{distance}$$

↑

Situation

- **Non-linear Transformations** : The RBF kernel enables the use of non-linear transformations, which can map the original feature space to a higher-dimensional space where the data becomes linearly seperable. This is particularly useful for problems where the decision boundary is not linear.

- **Local Descision** : Unlike some other kernels, the RBF kernel make "local" decisions. That is, the effect of each data point is limited to a certain region around that point. This can make the model more robust to outliers and create complex descision boundary.

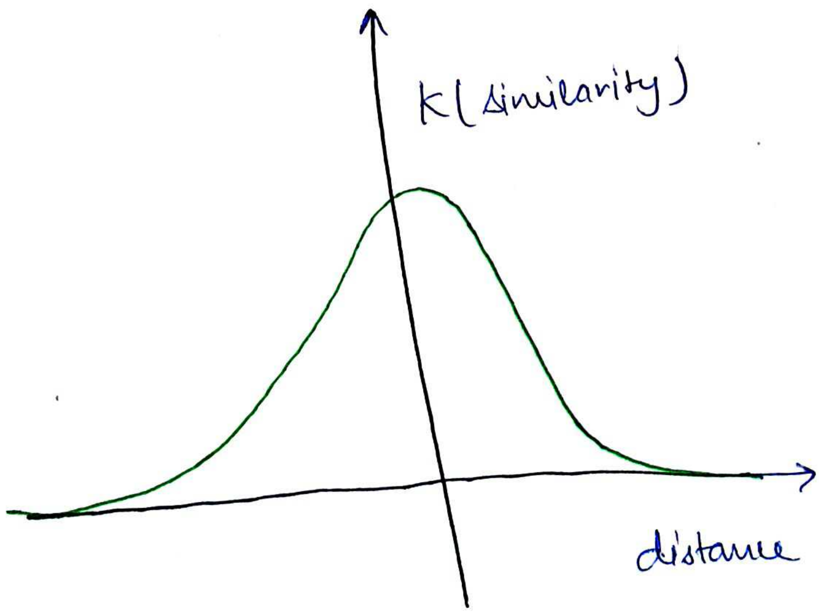
- **Flexiblity** : The RBF kernel has a parameter γ(related to the standard deviation of the Gaussian distribution) that determines the complexity of the decision boundary. By tuning this parameter, we can adjust the trade-off between bias and variance, allowing for a flexible range of decision boundaries.

- **Universal Approximation Property:** The RBF kernel has a property known as the universal approximate property, meaning it can approximately any continous function to a certain degree of accuracy given enough data points. This makes it highely versatile and capable of modeling a wide variety of relationship in data.

- **General purpose:** The RBF kernel does not makes any strong assumption about the versatile, general-purpose kernel.

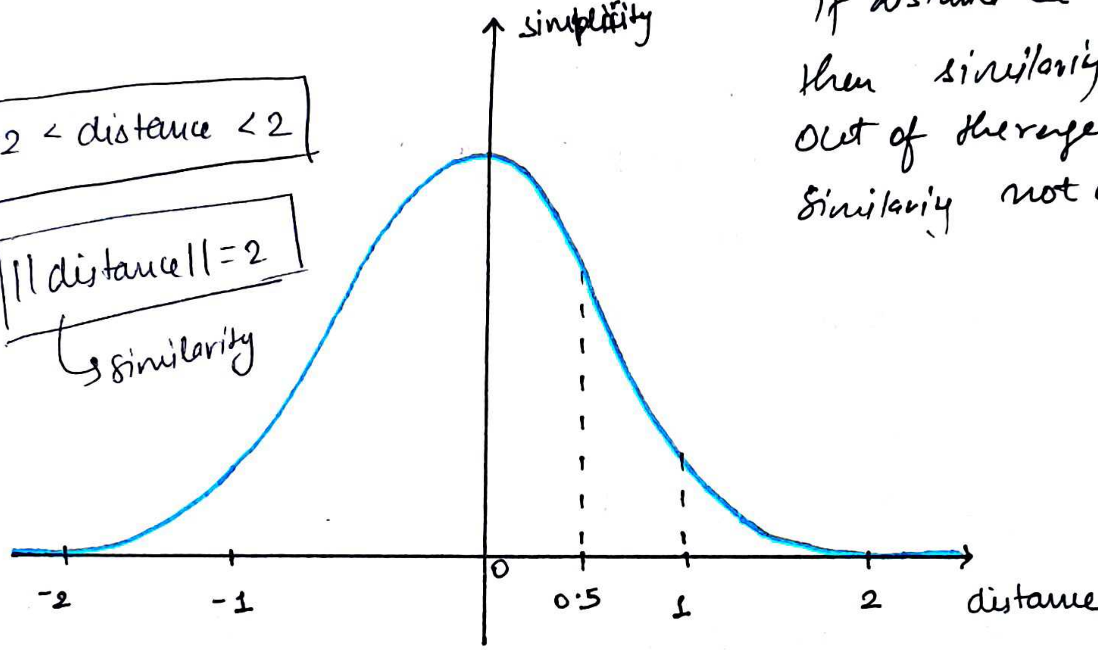
Local Decision Boundary

$$\left[e^{-\text{distance}^2} \right] \rightarrow x$$



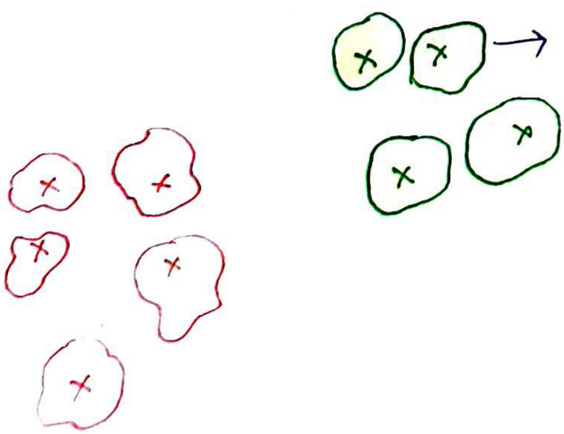
$\boxed{-2 < \text{distance} < 2}$

$\boxed{||\text{distance}|| = 2}$ → similarity



if distance bet^n -2 and 2 then similarity exist out of the range (-2 and 2) similarity not exist.

$e^{-||x_i - x_j||^2}$ → similarity and distance is exponential decrease.



→ $-2 < \text{distance} < 2$

if any point in the area then similar otherwise not similar.

Effect of Gamma

The parameter γ in the Radical Basis Function (RBF) kernel of a support vector Machine (SVM) is a hyperparameter that determines the spread of the kernel and therefore the decision region.

The effect of γ can be summarized as follows:

- If γ is too large, the exponential will decay very quickly, which means that each datapoint will only have an Influence in its Immediate vicinity. The result is a more complex descision boundary, which might overfit the training data.
- If γ is too small, the exponential will decay slowly, which means that each data point will only have an ~~Influence in its Immediate vicinity~~ wide range of influence.

The decision boundary will therefore be smoother and more simplistic, which underfit the training data.

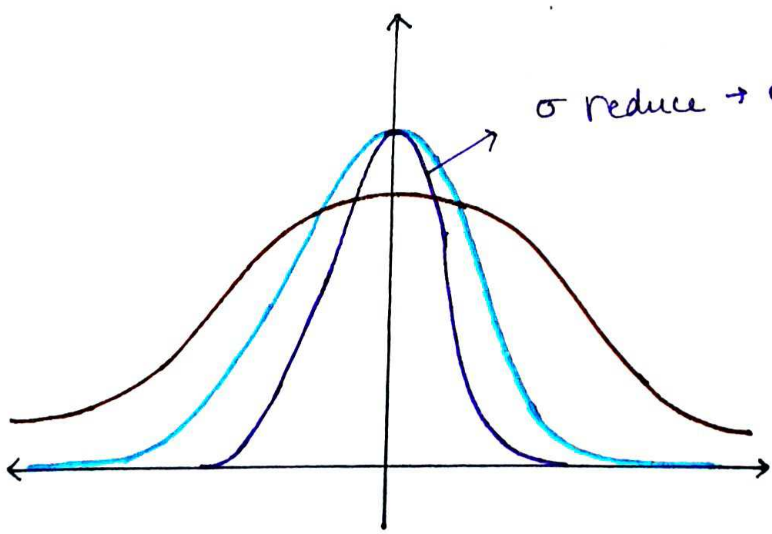
In a sense, γ in the RBF kernel plays a role similar to that of the inverse of the regularization parameter. It controls the trade-off between bias (underfitting) and variance (~~offer~~ overfitting). High γ values can lead to high variance (overfitting) due to more flexiblity in shaping the decision boundary, while low γ values can lead to high bias (underfitting) due to a more rigid, simplistic decision boundary.

Tuning the γ parameter using cross-validation or a similar technique a crucial step when training SVMs with an RBF kernel.

$0.1 \rightarrow$ small
$1 \rightarrow$ medi-
$10 \rightarrow$ large
} gamma

σ reduce $\rightarrow 0.1$

$\sigma \uparrow$ range$\uparrow$ accomodation



Bias - variance tradeoffe

$\sigma \rightarrow$ hyperparameters

$\sigma \downarrow$ or $\gamma \uparrow \rightarrow$ overfiting

$\gamma \downarrow \rightarrow$ locality $\uparrow$

$\sigma \propto \frac{1}{\gamma}$

$\sigma \uparrow$ or $\gamma \downarrow \rightarrow$ Underfitting

$\gamma \uparrow \rightarrow$ locality $\downarrow$

Relation Between RBF and Polynomial Kernel

Infinite Dimensional Mapping: The RBF kernel implicity maps input data to an infinite dimensional feature space, which allows for even greater flexibility in forming decision boundaries.

In polynomial

$$x_1 \quad x_2 \xrightarrow{\text{degree} \to 2} x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 x_2$$

In RBF $\rightarrow$ ∞ dim Vector

$x_1 \quad x_2$ $\longrightarrow$ ∞ dim Vect

$x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 x_2 \rightarrow d=2$

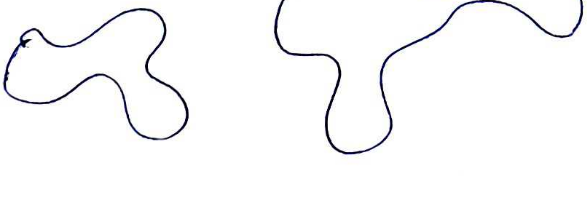
$x_1^3 \quad x_1^2 \quad x_1^2 x_2 \quad x_2^2 x_1 \rightarrow d=3$

$x_1^4 \quad x_2^4 \quad x_1^3 x_2^1 \quad x_1^2 x_2 \quad x_2^3 x_1^1 \rightarrow d=4$

⋮

$d = \infty$

* RBF $\rightarrow$ Because of ∞ dimension vector RBF make any type of decision boundary.



$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$ let $\sigma = 1$ $\sigma^2 = 1$

$$= e^{-\frac{\|x_i - x_j\|^2}{2}}$$

$$= e^{-\frac{(x_i - x_j)^T (x_i - x_j)}{2}} = e^{-\frac{(x_i^T - x_j^T)(x_i - x_j)}{2}}$$

$$= e^{-\frac{[x_i^T x_j - x_j^T x_j - x_j^T x_i + x_j^T x_j]}{2}}$$

$x_i = [x_{11} \ x_{12}]$

$x_j = [x_{21} \ x_{22}]$

$$= e^{-\frac{[x_i^T x_i + x_j^T x_j - x_i^T x_j - x_j^T x_i]}{2}}$$

$x_i^T x_j =$

$x_j^T x_i =$

$$= \underbrace{e^{-\frac{1}{2}[x_i^T x_i + x_j^T x_j]}}_{\text{constant}} e^{x_i^T x_j}$$

$$= C e^{1 + x_i^T x_j - 1}$$

$$= C e^{1 + x_i^T x_j} e^{-1}$$

$$= C' \sum_{k=0}^{\infty} \frac{(1 + x_i^T x_j)^k}{k!} \quad \left[\because e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}\right]$$

$$\hookrightarrow \quad C' \sum_{k=0}^{\infty} \frac{k_{poly}(x_i, x_j)^k}{k!}$$

$$C'\left[1 + \frac{(1 + x_i^T x_j)}{1!} + \frac{(1 + x_i^T x_j)}{2!} + \dots \frac{(1 + x_i^T x_j)^{\infty}}{\infty!}\right]$$

proved

$$\boxed{RBF = \sum_{i=0}^{\infty} k_{poly}(x_i, x_j)}$$

Custom Kernels

1. **String kernel:** These are used for classifying text or sequences, where the input data is not numerical. String kernels measure the similarity between two strings. For example, a simple string kernel might count the number of common substring between two strings.

2. **Chi-Square kernel:** This kernel is often used in computer vision problem, especially for histogram comparism. It's defined as $k(x,y) = \exp(-\gamma \chi^2(x,y))$, where $\chi^2(x,y)$ is the chi-square distance betn the histogram x and y.

3. **Intersection kernel:** This is another kernel commonly used in computer vision, which computes the intersection between two histograms (or generally non-negative feature vectors).

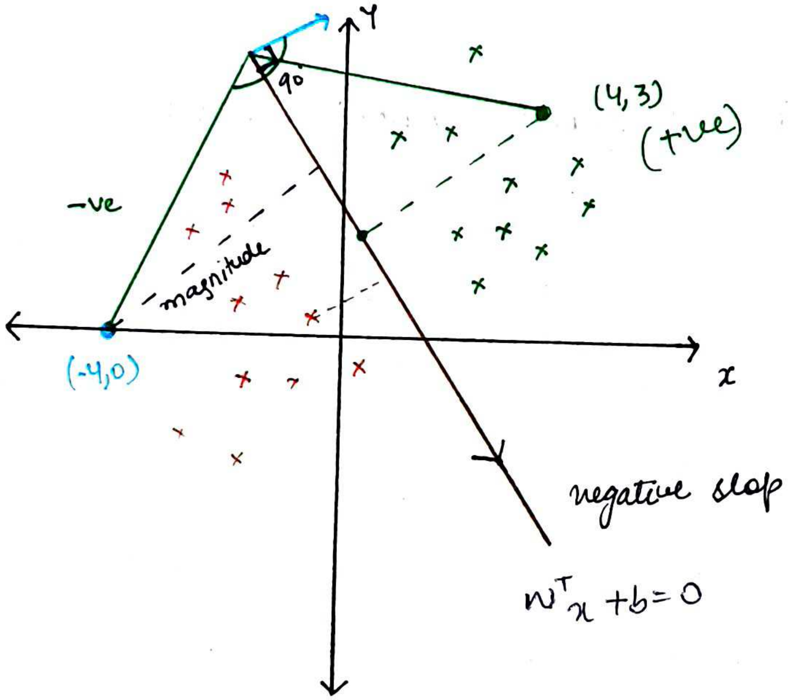
4. **Hellinger's kernel**: Hellinger's kernel, or Bhattacharya kernel, is used for comparing probability distribution and is popular in image recognition tasks.

5. **Radial basis function network (RBFN) kernels**:

These are similar to the standard RBF kernel, but the centers and width of the RBFs are learned from the data, rather than being fixed prior.

6. **Spectral kernels**: These kernels use spectral analysis techniques to compare data points. They can be particularly useful for dealing with cyclic or periodic data.

Another Method to find the point is positive or negative



$w^T x + b = 0$

$$y = mx + c$$

$$h\theta(x) = \theta_0 + \theta_1 x_1$$

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$$y = b + [w_1 x_1 + w_2 x_2 + w_3 x_3]$$

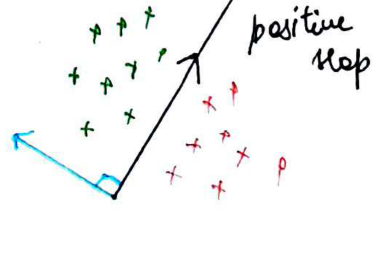
$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

If angle of point is more than 90° is ~~positive~~ negative point

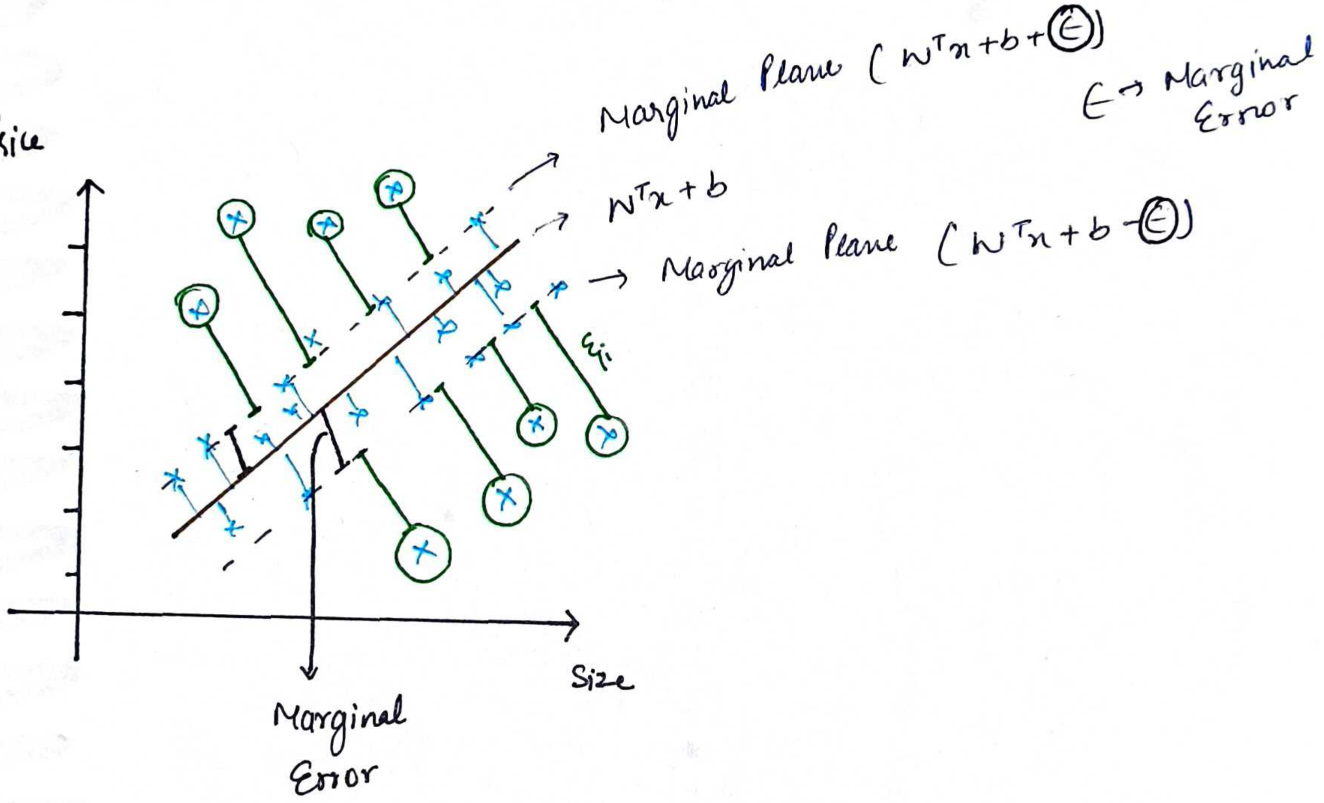
$$w^T = [w_1 \; w_2 \; w_3] \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$w^T x = 0$$

If angle of point is less than ~~or~~ equal to 90° is positive point.



SVR (Support Vector Regressor)



Cost function

$$\min_{w,b} \frac{\|w\|}{2} + \boxed{C \sum_{i=1}^{n} \xi_i} \Rightarrow \text{Hinge}$$

Constrain :

Error $\Leftarrow$ $|y_i - w^T x_i| \leq \epsilon$

* Points under the margin or distance between points and best fit line and error always less than margin

Error $\Leftarrow$ $|y_i - w^T x_i| \leq \epsilon + \xi_i$

Some points outside the margin so, we use ξ_i and add distance between the margin and outside point.