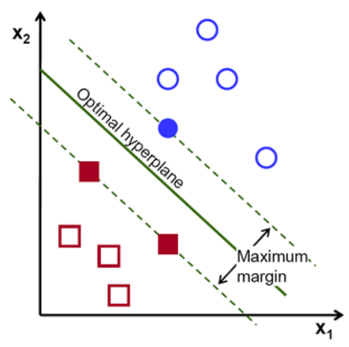
Support Vector Machine

--- Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges.

--- In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

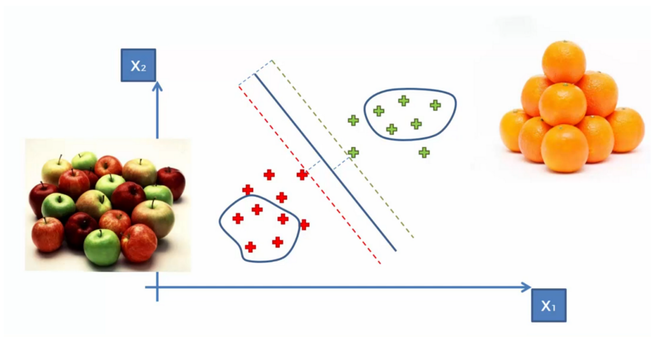
1. Linear SVM.

Key Idea : Find the planes which can separates Positive and negative points as widely as possible.



Support Vectors :

Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

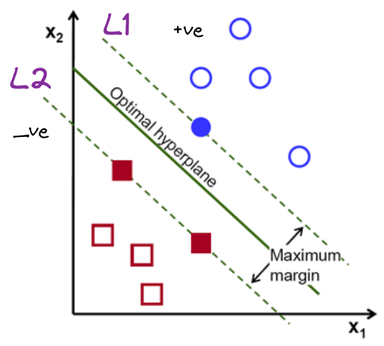


Mathematical Expression for SVM

$L1 = w^T x_i + b$

$L0 = w^T x_i + b$

$L2 = w^T x_i + b$



As we know :

$y_i * (w^T x_i + b) > 0$

$\rightarrow$ $y_i * (w^T x_i + b) = 1$

$\rightarrow$ $y_i * (w^T x_i + b) = -1$

Distance Between two parallel vector:

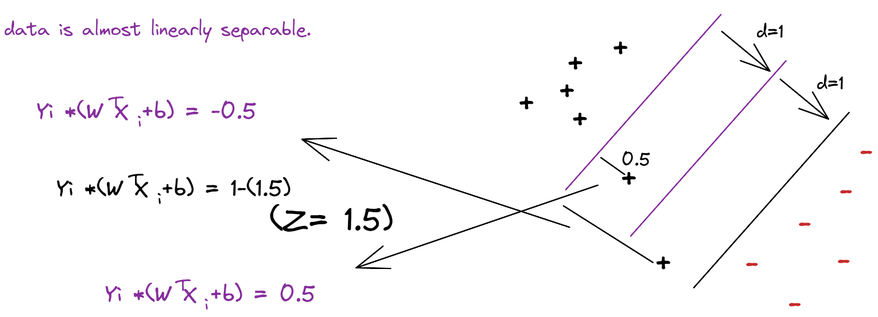
$$d = \frac{c1 - c2}{\| w \|}$$

After calculation :

$$(w^*, b^*) = Argmax \left(\frac{2}{\| w \|} \right)$$

s.t. $y_i * (w^T x_i + b) >= 1$ for all x_i

what if data is almost linearly separable.



$y_i * (w^T x_i + b) = -0.5$

$y_i * (w^T x_i + b) = 1 - (1.5)$

$(Z = 1.5)$

$y_i * (w^T x_i + b) = 0.5$

$y_i * (w^T x_i + b) = 1 - (0.5)$

$(Z = 0.5)$

Z - Zeta (Error in the measurement)
if Zeta is high means points have been incorrectly classified.

Z= 0 for correctly classified pts

Final Equation :

Avg dist for missclassified data ponits

$$(w^*, b^*) = Argmax \left(\frac{2}{\| w \|} \right) + C * 1/n \sum_{i=1}^{n} Zi$$

C - Hyperparameter

if C is high ---- Tendency to make mistake less on train data -- Chances of overfitting will be high
if C is less then underfit model can be there

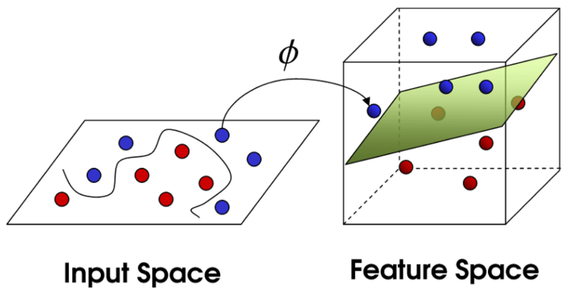
$$C = \frac{1}{\lambda}$$

Notes : We are taking +1 or -1 here to make computation easy.

$$W^T X + b = K$$

The Kernel Trick in Support Vector Classification

The kernel trick provides a solution to this problem. The "trick" is that kernel methods represent the data only through a set of pairwise similarity comparisons between the original data observations x (with the original coordinates in the lower dimensional space), instead of explicitly applying the transformations ϕ(x) and representing the data by these transformed coordinates in the higher dimensional feature space.



Kernel Function :

Our kernel function accepts inputs in the original lower dimensional space and returns the dot product of the transformed vectors in the higher dimensional space.

Kernel Definition

- A function that takes as its inputs vectors in the original space and returns the dot product of the vectors in the feature space is called a *kernel function*
- More formally, if we have data $\mathbf{x}, \mathbf{z} \in X$ and a map $\phi : X \to \Re^N$ then

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

is a kernel function

1. Polynomial Kernel :

-- Increase the dimensionality

$$K(X_1, X_2) = (a + X_1^T X_2)^b$$

in the polynomial kernel, we simply calculate the dot product by increasing the power of the kernel.

Example:

Let's say originally X space is 2-dimensional such that

Xa = (a1 ,a2)

Xb = (b1 ,b2)

now if we want to map our data into higher dimension let's say in Z space which is six-dimensional it may seem like

$$Z_a = \phi(X_a) = (1, a_1, a_2, a_1^2, a_2^2, a_1 * a_2)$$
$$Z_b = \phi(X_b) = (1, b_1, b_2, b_1^2, b_2^2, b_1 * b_2)$$

using kernel trick:

$$Z_a^T Z_b = k(X_a, X_b) = (1 + X_a^T X_b)^2$$
$$Z_a^T Z_b = 1 + a_1 b_1 + a_2 b_2 + a_1^2 b_1^2 + a_2^2 b_2^2 + a_1 b_1 a_2 b_2$$

2. Radial basis function kernel (RBF)/ Gaussian Kernel:

RBF kernel is a function whose value depends on the distance from the origin or from some point.
Gaussian Kernel is of the following format;

$$K(X_1, X_2) = exponent(-\gamma \|X_1 - X_2\|^2)$$

||X1 — X2 || = Euclidean distance between X1 & X2

Using the distance in the original space we calculate the dot product (similarity) of X1 & X2.

Hyperparameters:

1. C: Inverse of the strength of regularization.

Behavior: As the value of 'c' increases the model gets overfits.

As the value of 'c' decreases the model underfits.

2. γ : Gamma (used only for RBF kernel)

Behavior: As the value of ' γ' increases the model gets overfits.

As the value of ' γ' decreases the model underfits.