SUPPORT VECTOR MACHINE

Support Vector Machines (SVM) is a powerful and widely used algorithm in machine learning, particularly for classification tasks. SVM is known for its ability to handle high-dimensional data and can be effective even with limited training samples.

Imagine you have a dataset with two classes of points, and your goal is to find a line that can separate these two classes as best as possible. SVM is an algorithm that helps you do just that.

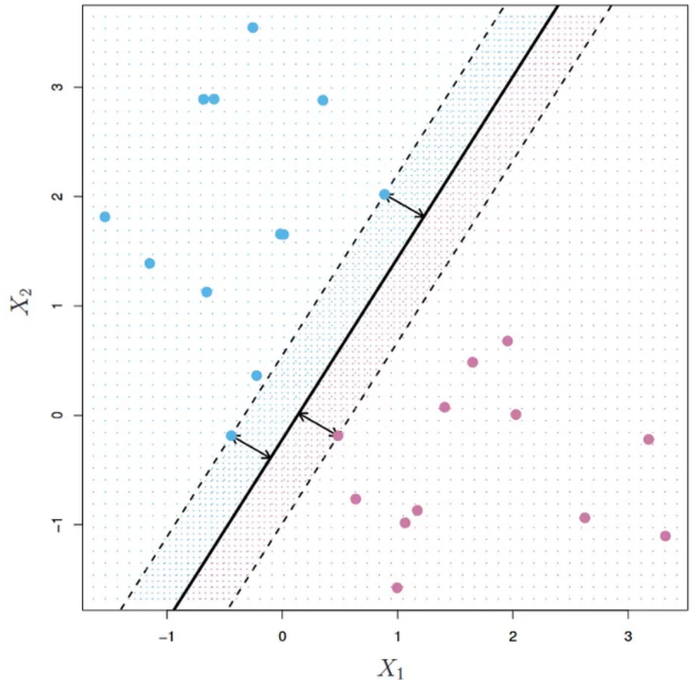
Data Preparation:

You start with a set of data points, each with features (attributes) and a class label indicating their class (e.g., red or blue).

SVM works best when the data is linearly separable, meaning you can draw a straight line to separate the classes. However, SVM can handle some degree of overlapping as well.

Finding the Optimal Hyperplane:

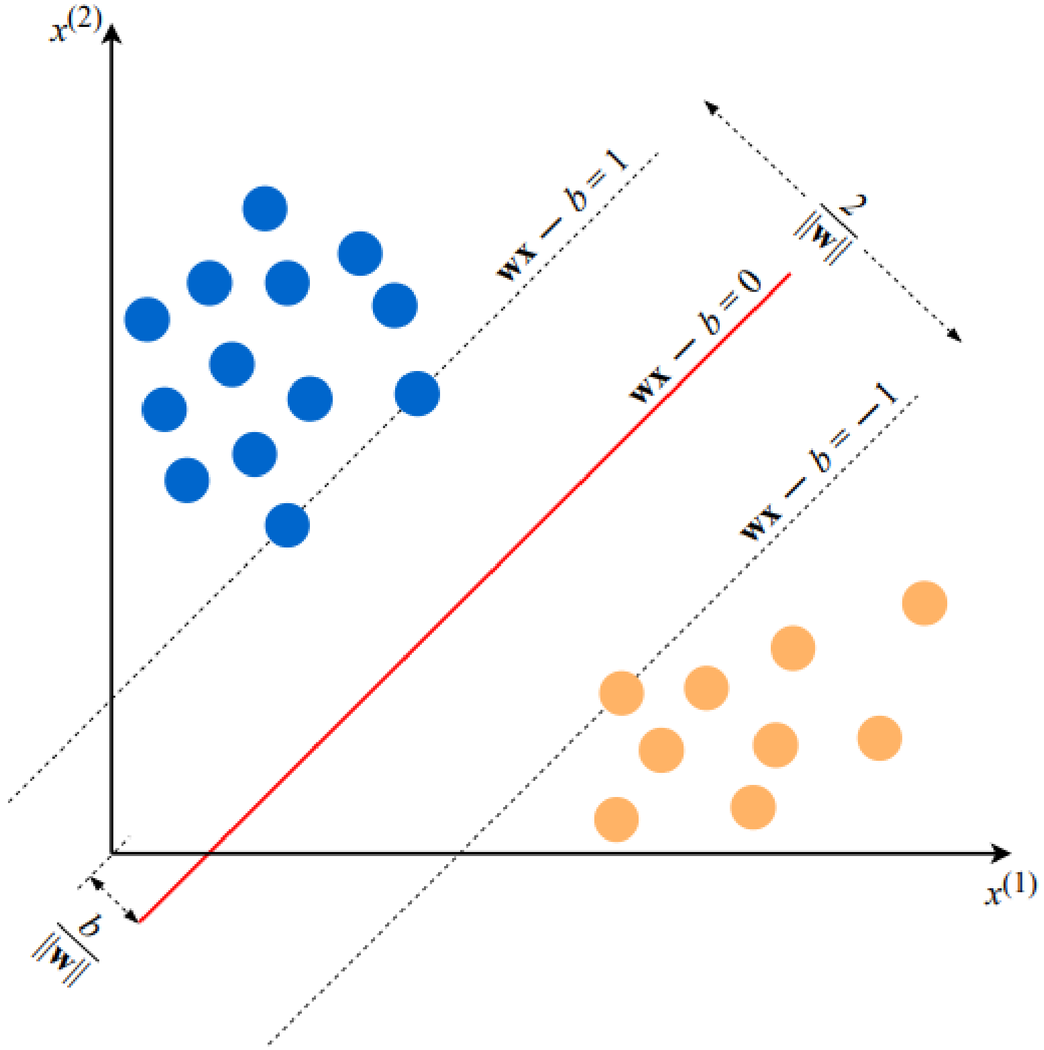
SVM aims to find the best line, called the hyperplane, that separates the classes while maximizing the margin. The margin is the distance between the hyperplane and the closest data points from each class. The larger the margin, the better the separation. SVM selects the hyperplane that has the maximum distance to the closest data points from both classes. These data points are called support vectors.



The equation of the hyperplane is given by two parameters, a real-valued vector w of the same dimensionality as our input feature vector x, and a real number b like this:

$$wx - b = 0,$$

where the expression wx means w(1)x(1) + w(2)x(2) + ... + w(D)x(D), and D is the number of dimensions of the feature vector x.



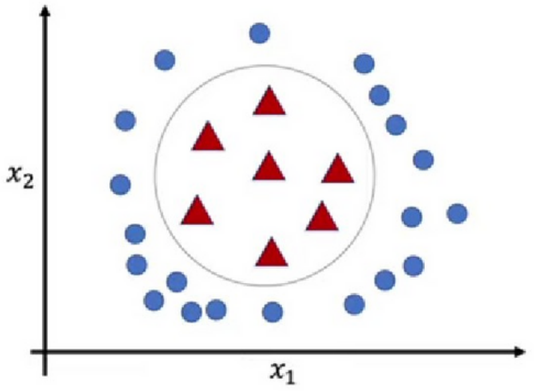
- wxi - b >= 1 if yi = +1, and
- wxi - b <=-1 if yi = -1

yi(wxi+b)>=1

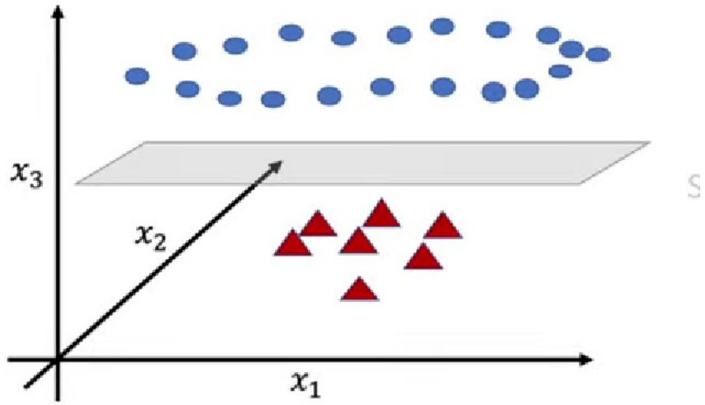
SVM-Kernels:

Sometimes, the original data cannot be separated by a straight line. In such cases, SVM uses kernel tricks to transform the data into a higher-dimensional space.

By transforming the data, SVM can find a hyperplane that separates the classes effectively.

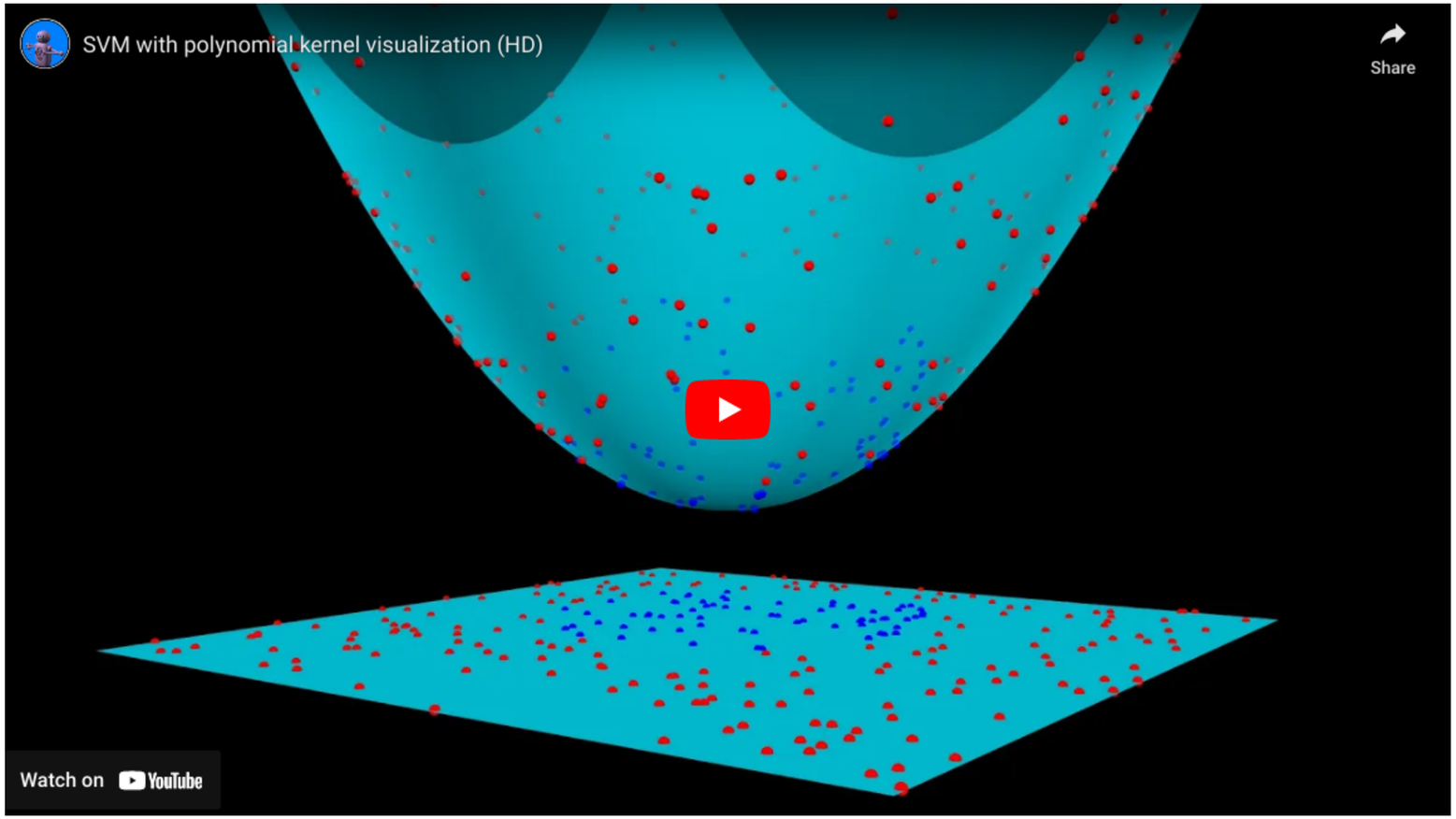


SVM in 2 dimensions



SVM in 3 dimensions

Make from this whatever you want to, you will understand this better once you go through the following slides.

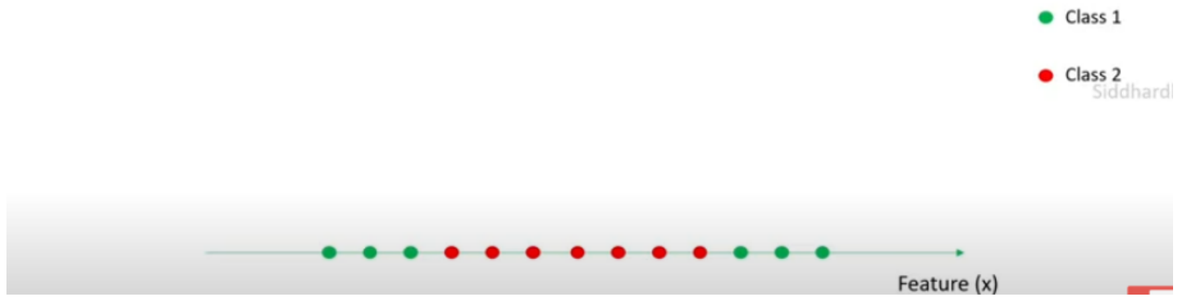


The kernel function generally transforms the training set of data so that a non-linear decision surface can be transformed to a linear equation in higher no. of dimension spaces. It return the inner product between two points in a standard feature dimension.

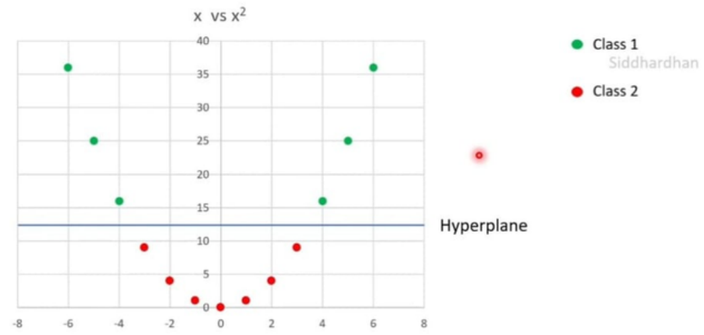
Examples:

1)Linear Kernel

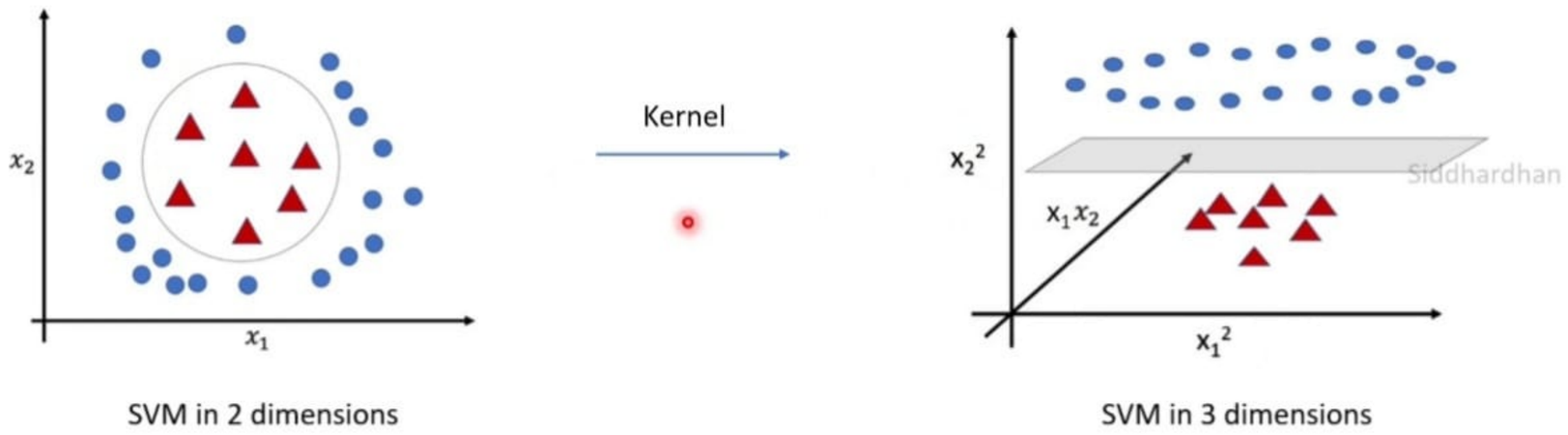
Feature (x)	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6



Feature (x)	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
x^2	36	25	16	9	4	1	0	1	4	9	16	25	36

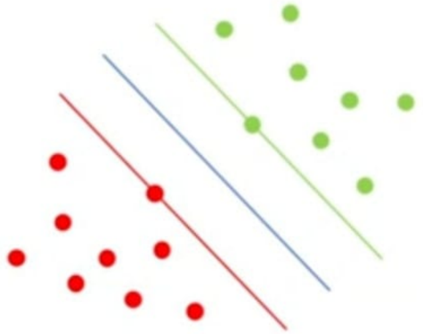


2)Polynomial Kernel

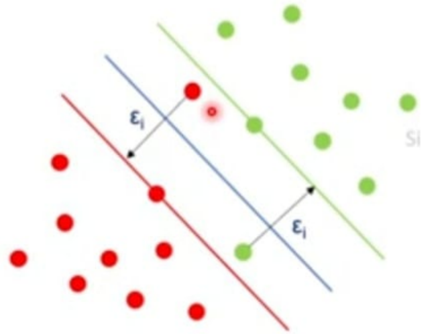


Soft Margin Classification:

In real-world scenarios, the data might not be perfectly separable. There could be outliers or noise that makes it difficult to find a hyperplane that separates the classes perfectly.
SVM introduces a concept of a soft margin that allows for some misclassifications. It balances between maximizing the margin and minimizing the misclassifications.
The algorithm finds the optimal hyperplane that minimizes the errors while maximizing the margin. The points that fall within the margin or are misclassified contribute to the error.



Hard Margin



Soft Margin

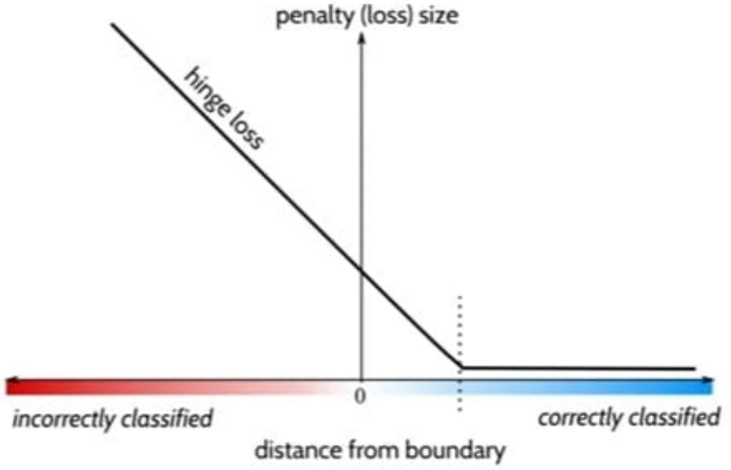
Hinge Loss

Hinge Loss is one of the types of Loss Function, mainly used for maximum margin classification models.

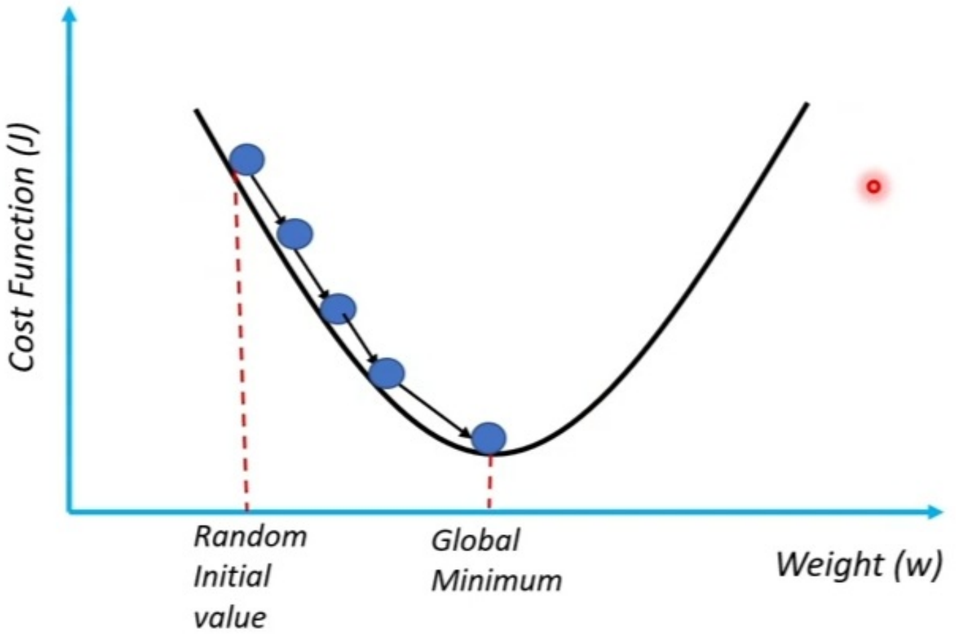
Hinge Loss incorporates a margin or distance from the classification boundary into the loss calculation. Even if new observations are classified correctly, they can incur a penalty if the margin from the decision boundary is not large enough.

$$L = \max(0, 1 - y_i (w^T x_i + b))$$

0 - for correct classification
1 - for wrong classification



Gradient Descent



Gradient Descent

Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithms. It is used for updating the parameters of the learning model.

$$w_2 = w_1 - L * \frac{dJ}{dw}$$

$$b_2 = b_1 - L * \frac{dJ}{db}$$

w --> weight
b --> bias
L --> Learning Rate

$\frac{dJ}{dw}$ --> Partial Derivative of cost function with respect to w

$\frac{dJ}{db}$ --> Partial Derivative of cost function with respect to b

Objective Function:

The objective function of soft margin SVM combines the hinge loss with a regularization term to control the complexity of the model

$$J(w, b) = C * \Sigma[\max(0, 1 - y_i * (w^T * x_i + b))] + 0.5 * ||w||^2$$

where C is the regularization parameter that balances the trade-off between margin maximization and hinge loss minimization. The ||w||^2 term is the L2 regularization term.

The gradient descent algorithm is used to iteratively update the model's parameters based on the gradients of the objective function with respect to w and b.

The gradients of the hinge loss function with respect to w and b are:

$$\partial L(y_i, f(x_i)) / \partial w = -y_i * x_i$$

$$\partial L(y_i, f(x_i)) / \partial b = -y_i$$

The gradients of the regularization term with respect to w and b are:

$$\partial(0.5 * ||w||^2) / \partial w = w$$

$$\partial(0.5 * ||w||^2) / \partial b = 0$$

The gradients of the objective function can be obtained by summing the gradients of the hinge loss and the regularization term over all samples:

$$\partial J(w, b) / \partial w = C * \Sigma[-y_i * x_i] + w$$

$$\partial J(w, b) / \partial b = C * \Sigma[-y_i]$$

Making Predictions:

Once the optimal hyperplane is determined, you can use it to make predictions on new, unseen data points.
Given a new data point, SVM classifies it based on which side of the hyperplane it falls on. If it's on one side, it belongs to one class, and if it's on the other side, it belongs to the other class.