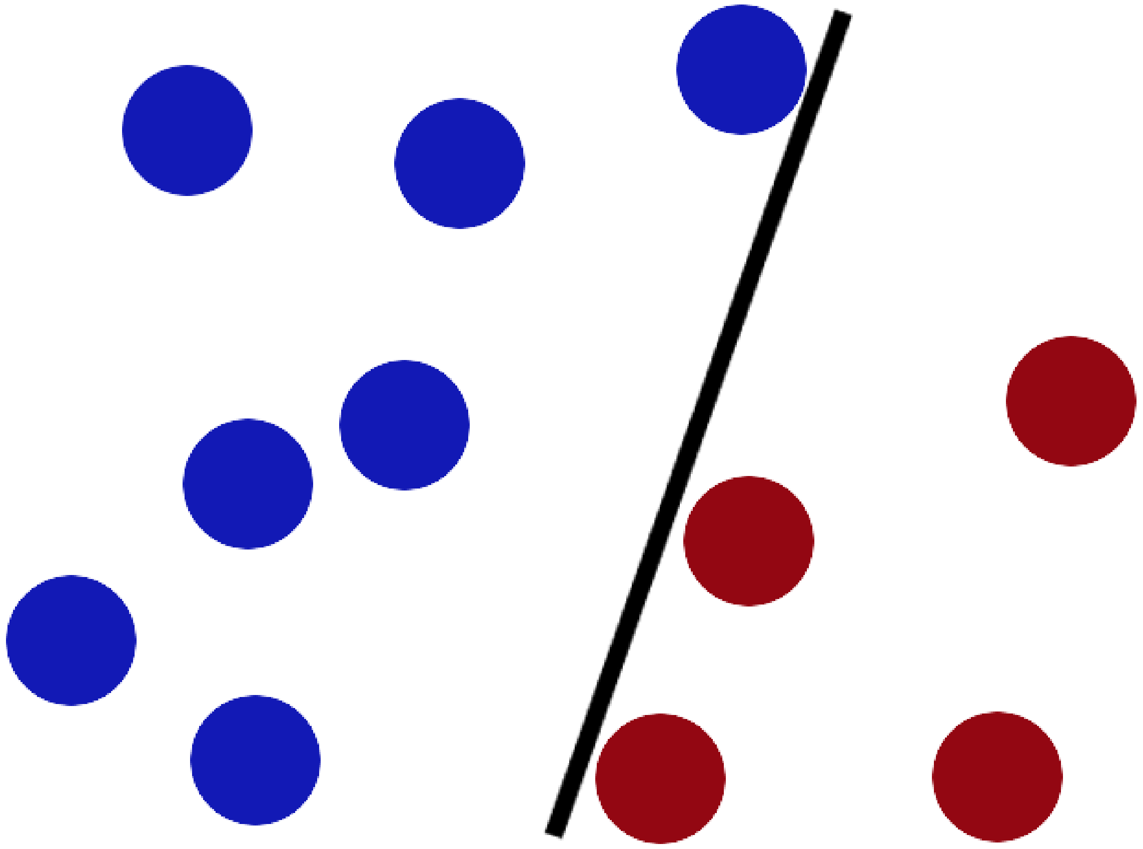
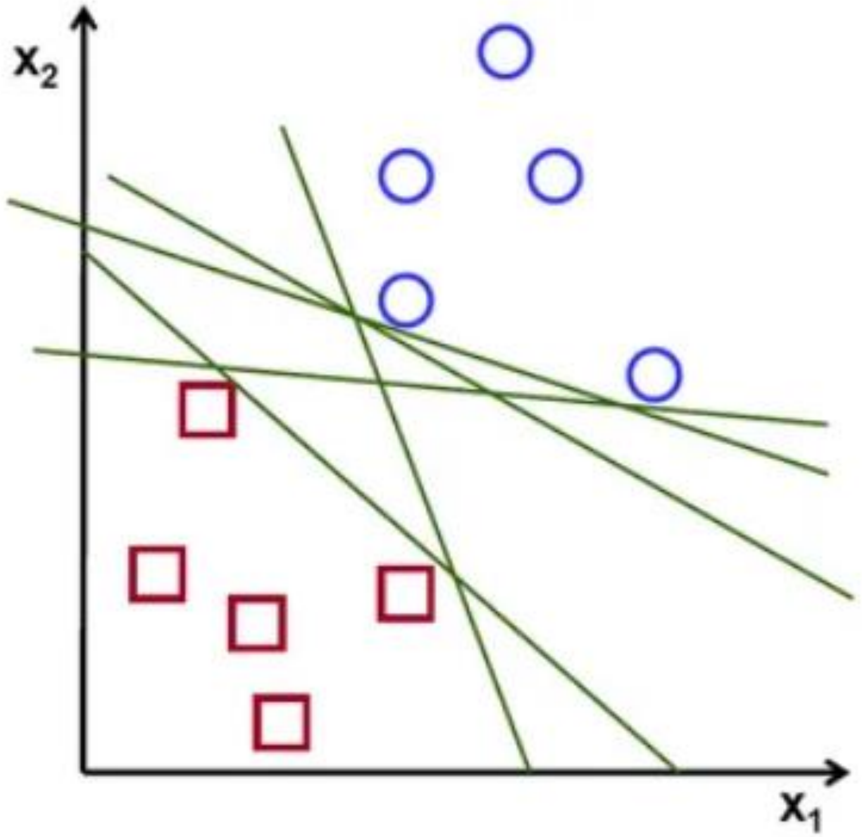


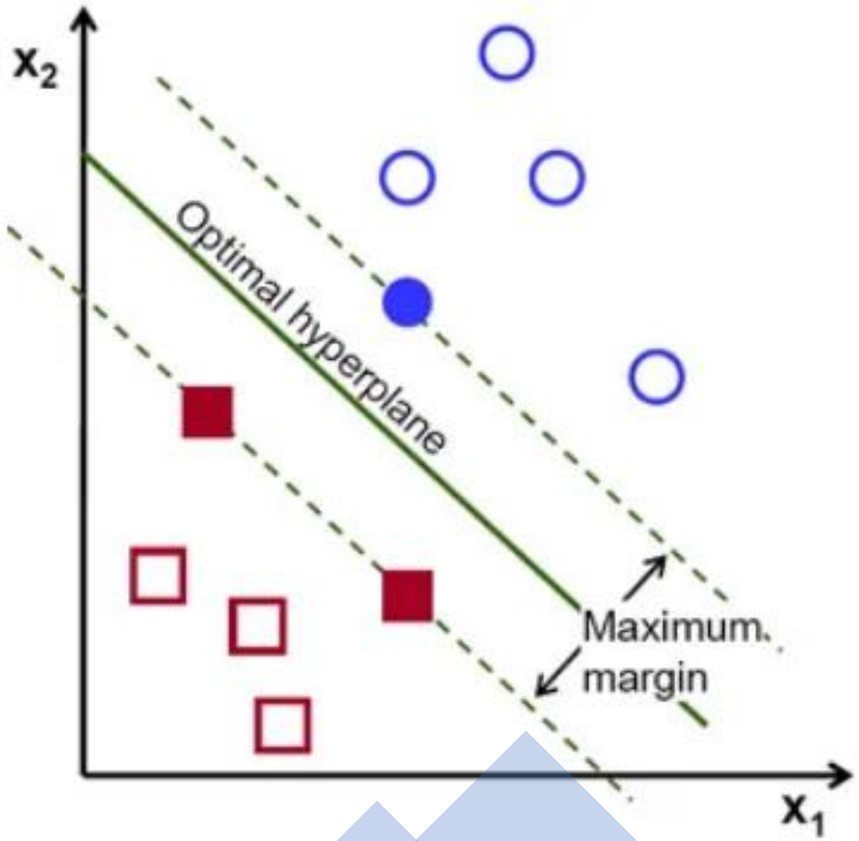
SUPPORT VECTOR MACHINES

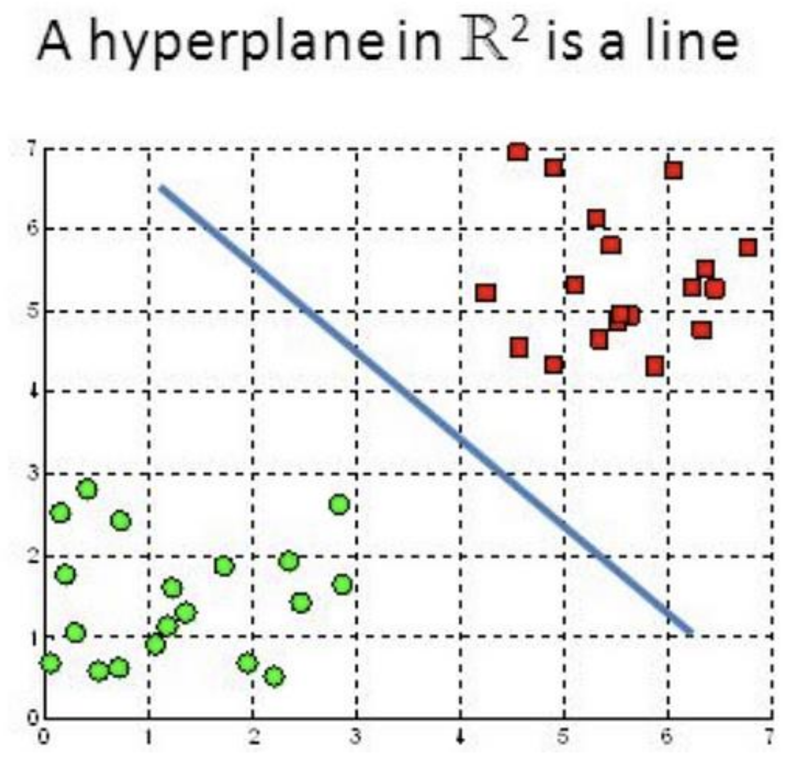
Support Vector Machines

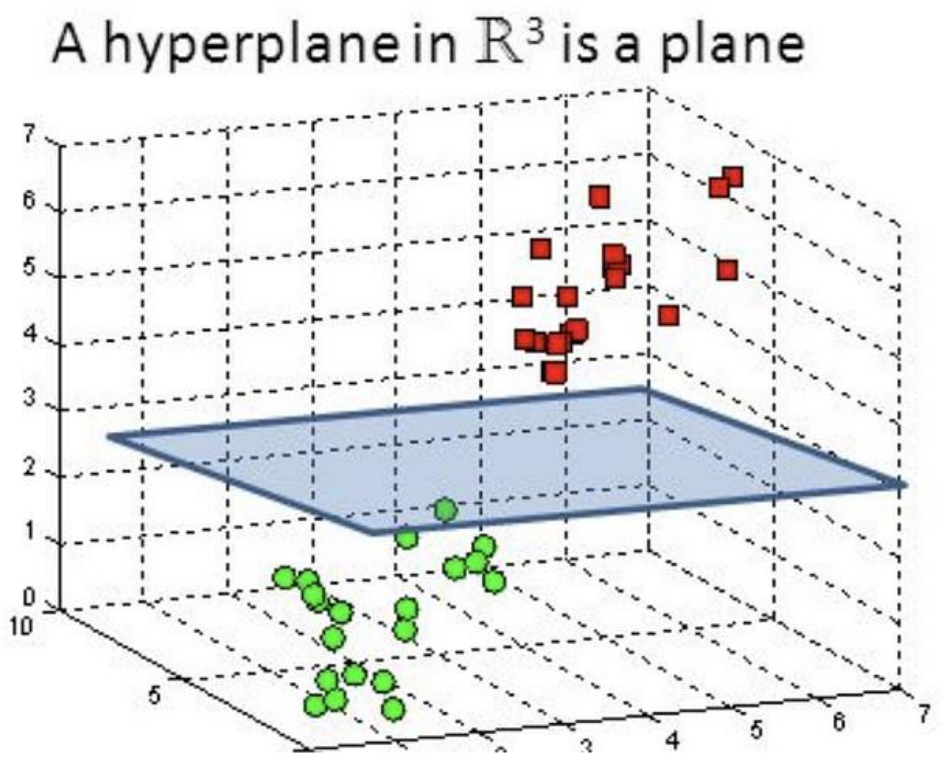
The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N - the number of features) that distinctly classifies the data points.

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.









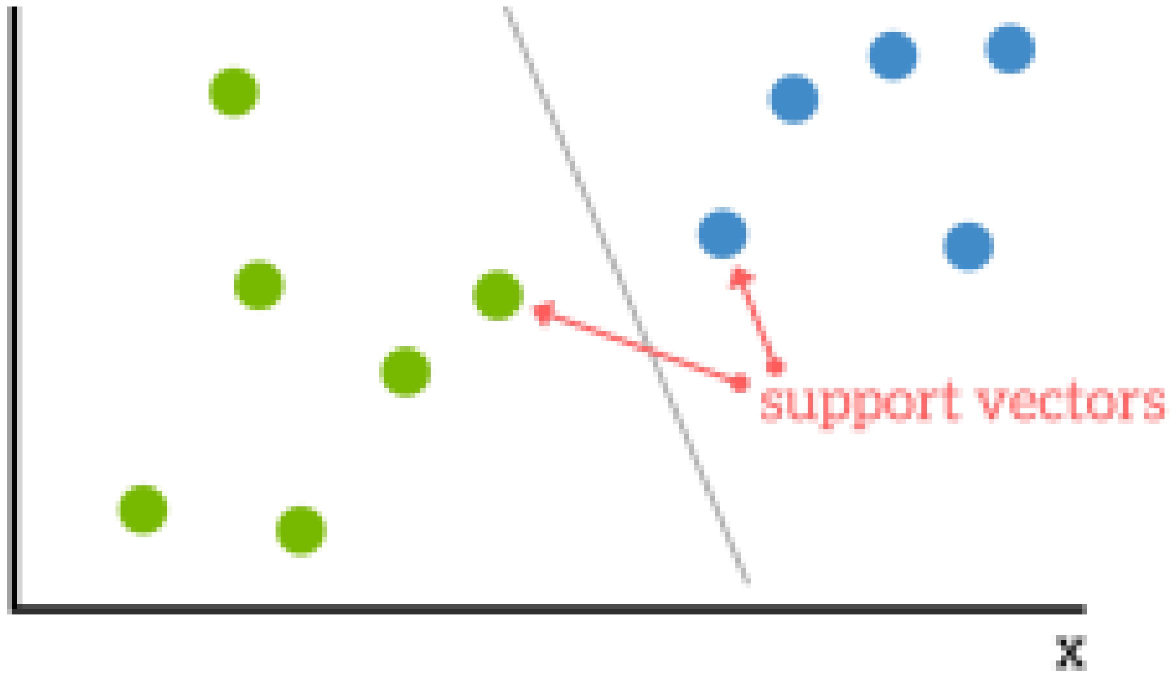
HYPERPLANES

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes.

The dimension of the hyperplane is determined by the number of features in the data. For instance, with two features, the hyperplane is a line, while with three features, it becomes a two-dimensional plane. Visualizing hyperplanes becomes increasingly challenging as the number of features exceeds three.

Support Vectors

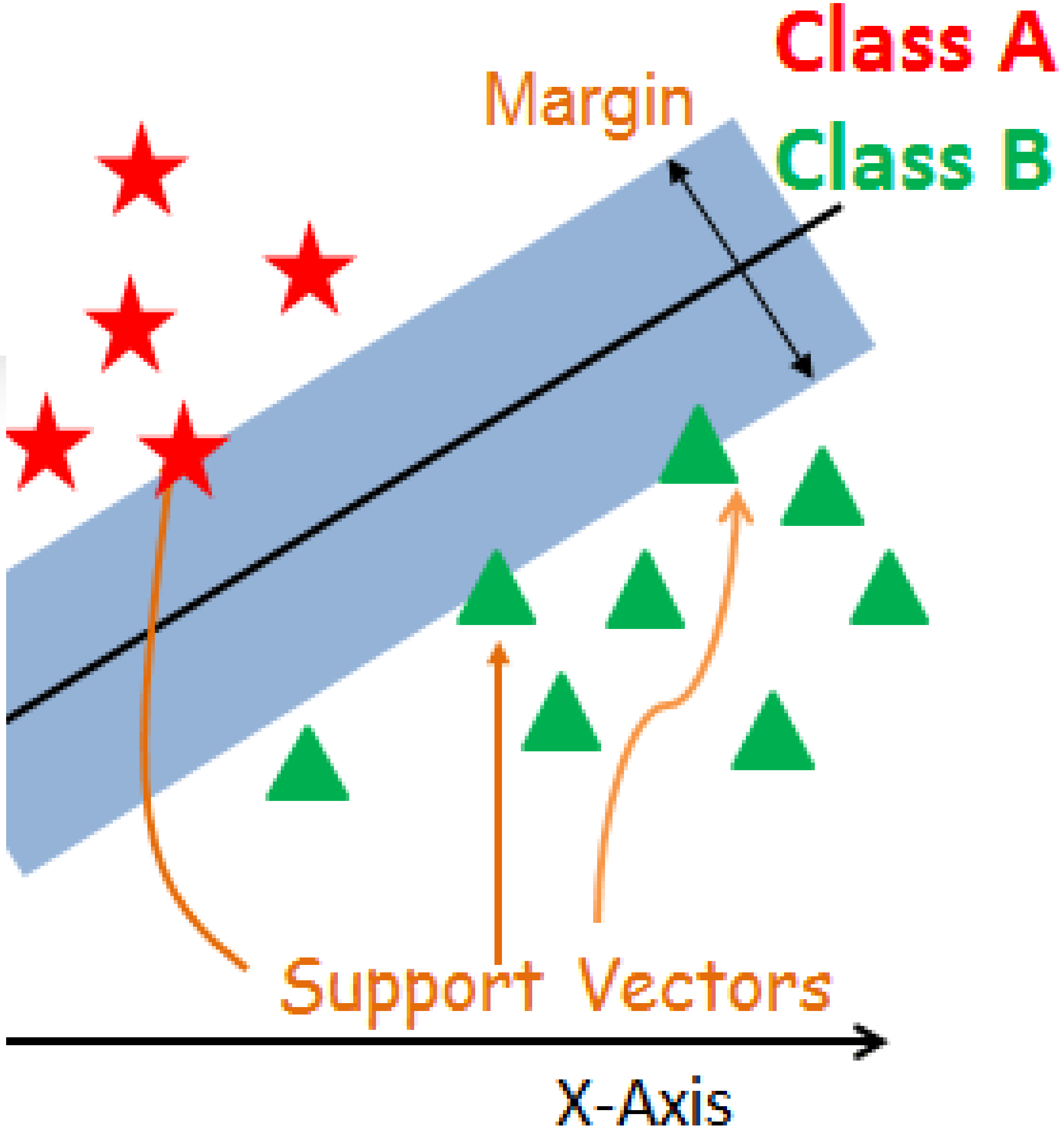
Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane, deleting the support vectors will change the position of the hyperplane. Using these support vectors, we maximize the margin of the classifier



Classification

We apply the sigmoid function to an output of a linear function, which If the transformed value is greater than a predefined threshold (usually 0.5) then it transforms the value into a range between 0 and 1.

In SVM, instead of using a threshold of 0.5, we use a threshold of 1 and -1. If the output is greater than 1, the data point is associated with one class, and if the output is less than -1, it's associated with another class.



Loss Function

The hinge loss measures the cost associated with a specific data point (x) with respect to its true label (y) and the model's prediction (f(x)).

If the product of the true label (y) and the model's prediction (y * f(x)) is greater than or equal to 1, the hinge loss is 0. This indicates that the data point is correctly classified and sufficiently far from the decision boundary, so there's no cost associated with this point.

If the product y * f(x) is less than 1, it means the prediction is on the wrong side of the decision boundary or is too close to it. In this case, the hinge loss is defined as (1 - y * f(x)), which quantifies the error or cost associated with this misclassification. The closer the value of y * f(x) is to 1, the higher the hinge loss will be.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

A regularization parameter (often denoted as λ) is added to the cost function. This regularization parameter helps strike a balance between two important objectives: maximizing the margin to find a decision boundary that maximizes the separation between different classes of data points, and minimizing the loss to reduce the errors in prediction.

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

The term $min_w \lambda \parallel w \parallel^2$ is the regularization term. It encourages the model to keep the weight vector (w) as small as possible while still achieving accurate classification. The regularization parameter (λ) controls the trade-off between margin maximization and loss minimization.

The sum represents the sum of hinge losses for all data points. The hinge loss quantifies the errors in classification, with a larger loss for points that are farther from the correct side of the decision boundary. $(1 - y_i(x_i, w))_+$ ensures that the loss is only considered when the prediction is on the wrong side of the decision boundary (i.e., when $y_i(x_i, w)$ is less than 1).

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

$$\frac{\delta}{\delta w_k}\lambda||w||^2 = 2\lambda w_k$$

For the regularization term the gradient is a straightforward calculation. It's equal to twice the regularization parameter (2λ) multiplied by the weight (w_k) itself.

$$\frac{\delta}{\delta w_k}(1 - y_i(x_i, w))_+ = \begin{cases} 1 - yf(x), \text{if } y * f(x) \geq 1 \\ 0, \text{otherwise} \end{cases}$$

For the hinge loss term, the gradient depends on whether the data point is correctly classified or not.

- If the product y * f(x) is greater than or equal to 1, meaning the prediction is correct, the gradient is 1-y * f(x). we only have to update the gradient from the regularization parameter.

$$w = w - \alpha \cdot (2\lambda w)$$

- If the product y * f(x) is less than 1, the gradient is 0 because there is no contribution to the loss. we include the loss along with the regularization parameter to perform gradient update.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

w is the current weight vector.

α is the learning rate, which determines the step size in the weight update.

y_i is the true label of the data point.

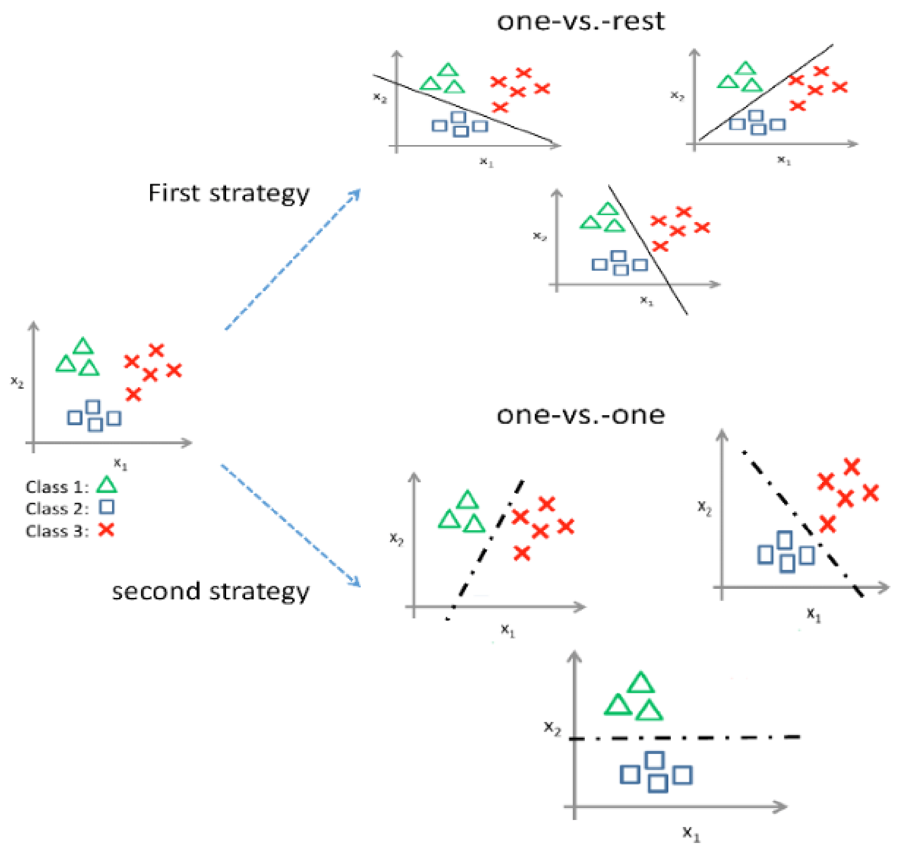
x_i is the feature vector of the data point.

λ is the regularization parameter.

SVM for more than two classes.

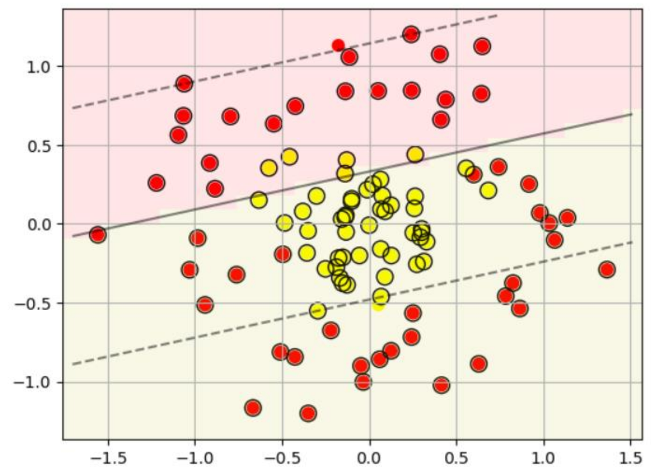
- (OvR) It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.

- (OvO) One-vs-one splits a multi-class classification dataset into binary classification problems. The one-vs-one approach splits the dataset into one dataset for each class versus every other class.

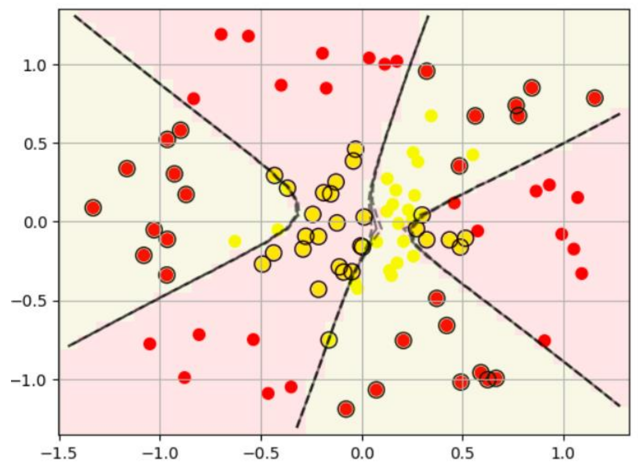


Kernel Trick.

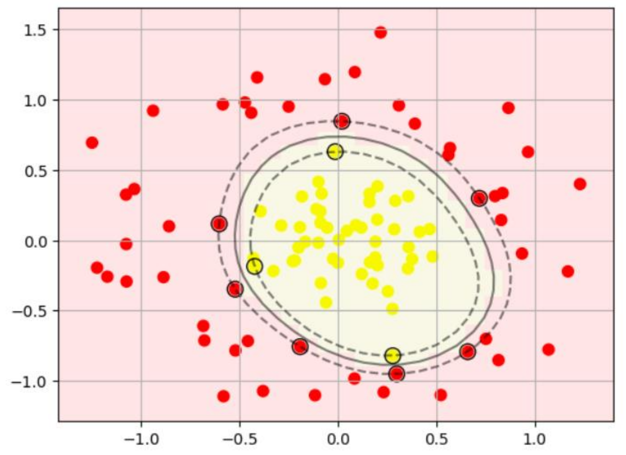
The Kernel trick refers to a technique that allows the correct functioning of SVM on those data that are not linearly separable, that is, data that cannot be easily divided with a straight line. This allows SVMs to handle more complex data and find nonlinear patterns more effectively.



SVC(kernel='linear')



SVC(kernel='sigmoid')



SVC(kernel='rbf')