SVM

* It is highly preferred by many as it produces significant accuracy with less computation power.
* It is used for both regression and classification tasks. But it is widely used for classification of objectives.
* The objective of the SVM algorithm is to find a hyper plane in an N-dimensional space that distinctly classifies the data points. Here N is number of features.
* It follows a technique called the **“Kernel Trick”** to transform the data and based on these transformations, it finds an optimal boundary between the possible outputs.

# Hyperplanes and Support vectors

* To separate two classes there are many hyperplanes that could be chosen, but our goal is to find a plane that has the maximum margin (maximum distance b/w the data points of both classes).
* Hyperplane for which the margin is maximum is called, **“Optimal hyperplane”.**
* If number of input features are **1 then a single point** represents the hyperplane.
* If number of input features **2 then hyperplane is a line.**
* f number of input features are **3 then hyperplane is a 2D plane.**
* **Support vectors** are data points that are close to hyperplane and influence the position and orientation of the hyperplane. By using these we will maximize the margin of the classifier. Deletion of supports vectors can change the position of the hyperplane.

# Kernel, Regularization parameter(C), gamma

* SVM uses function called **“Kernel trick”.** Where the kernel function transforms the data into the higher dimensional feature space so that a linear separation is possible.
* SVM with a Linear Kernel behaves very much like to logistic regression, it is implemented in LinearSVC where you can specify you desired loss.
* SVM with more complicated kernels are implemented in SVC where you must use the "hinge" loss but you can specify the kernel.
* Kernel – it specifies the kernel type to be used in the algorithm. It can be ‘linear’,’ploy’,’rbf’,’sigmoid’. The default value is ‘rbf’
* In real word application, finding perfect class for millions of training data set takes lot of time, this is called **“Regularization parameter”.**
* **For large values of C**, the optimization will choose a **smaller-margin hyperplane** if that hyperplane does a better job of getting all the training points **classified correctly.**
* With **low gamma, points far away** from plausible separation line are considered in calculation for the separation line. Whereas **high gamma means the points close** to plausible line are considered in calculation.

# Logistic Regression vs SVM

* In logistic Regression we take the output of the linear function and squash the value within the range of [0, 1] using sigmoid function. If the squashed value is greater than the **threshold value 0.5**, we assign it as label 1, else we assign it a label 0.
* In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify as another class. Since the threshold values are changed to **1 and -1 in SVM**.

# Coding Part

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

plt.rcParams["figure.figsize"]=(10,10)

from sklearn.svm import SVC

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

data=load\_iris()

X=data.data

Y=data.target

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,random\_state=0,test\_size=0.4)

model = SVC(kernel = 'linear')

model.fit(X\_train,Y\_train)

Y\_pred = model.predict(X\_test)

print(“Accuracy: ”,accuracy\_score(Y\_test,Y\_pred)\*100)

plt.scatter(X\_train[:,1],Y\_train,label="Training type")

plt.scatter(X\_test[:,1],Y\_pred,color='red', marker='+',label="Predicted type")

plt.xlabel("Features")

plt.ylabel("Target")

plt.title("SVM")

plt.show()

output:

Accuracy: 96.66666666666667

