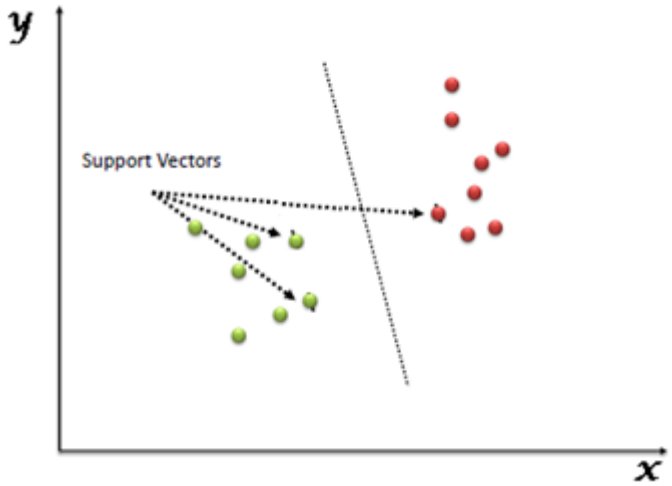
Support Vector Machine (SVM) in Python and R

What are Support Vector Machines?

Enable fullscreen

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in an n-dimensional space (where n is the number of features you have). The value of each feature here is the value of a particular coordinate. Then, we perform classification by finding the plane that differentiates the two classes very well (look at the below snapshot).



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

You can look at support vector machines and a few examples of its working here.

Why do we use SVM and how is it better?

Think of machine learning algorithms as an armoury packed with axes, sword, blades, bow, dagger, etc. You have various tools, but you ought to learn to use them at the right time. As an analogy, think of ‘Regression’ as a sword capable of slicing and dicing data efficiently, but incapable of dealing with highly complex data. On the contrary, the ‘Support Vector Machine’ is like a sharp knife – it works on smaller datasets, but on the complex ones, it can be much stronger and powerful in building machine learning models.

The main advantages of using the SVM algorithm are:

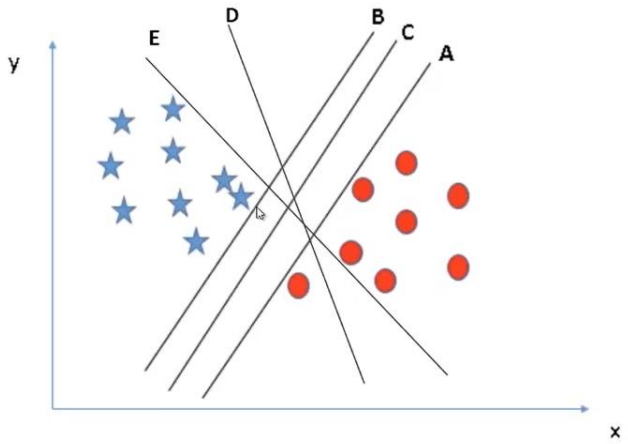
- It works really well with a clear margin of separation
- It is effective in high dimensional spaces.
- It is effective in cases where the number of dimensions is greater than the number of samples.
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Support Vector Machines

- Supervised Learning

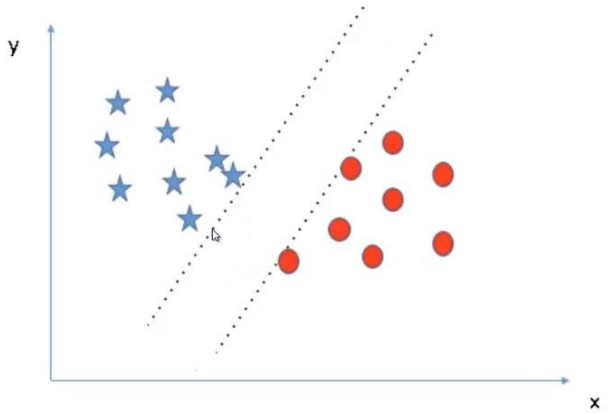
- Handles both Classification (primarily) and Regression

Support Vector Machines



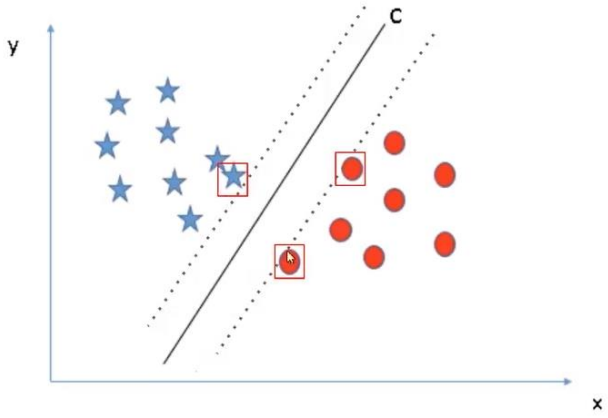
Support Vector Machines

- 100% classification
- Best Separation
- Maximum Margin



Support Vector Machines

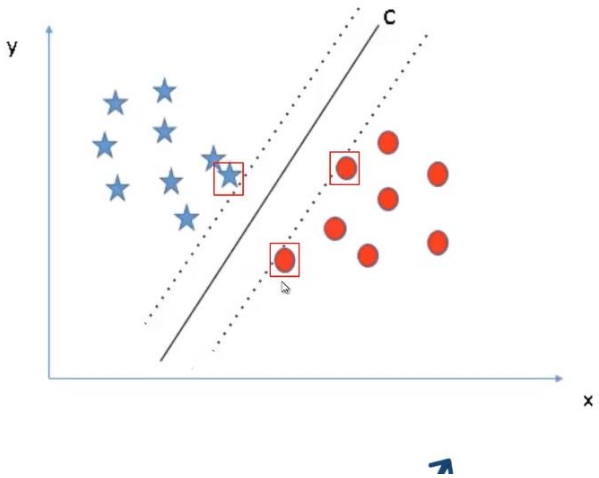
- 100% classification
- Best Separation
- Maximum Margin



Support Vector Machines

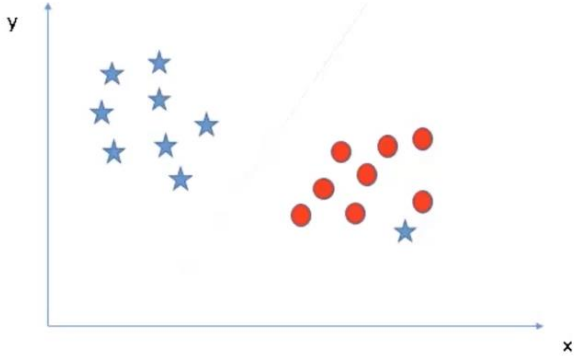
- 100% classification
- Best Separation
- Maximum Margin

Hard Margin SVM



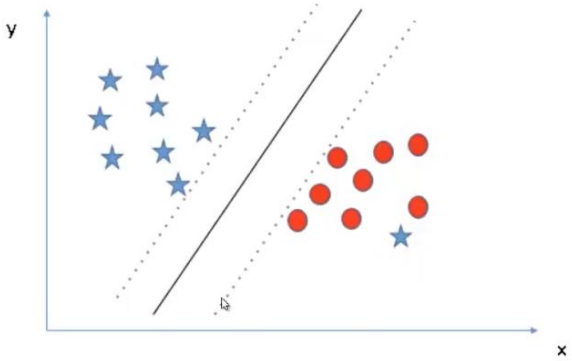
Support Vector Machines

- Maximum Classification
- Best Separation
- Maximum Margin



Support Vector Machines

- Maximum Classification
- Best Separation
- Maximum Margin



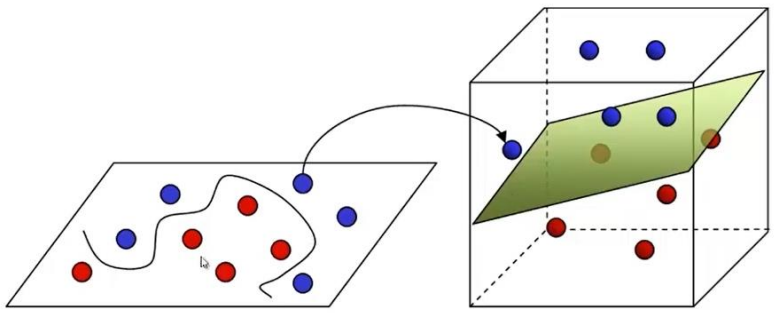
Soft Margin SVM

Analytics Vidhya

SVM is robust to outliers as it only considers cases nearer to margin

Non-linear Separation and Margins

Support Vector Machines



Input Space Feature Space

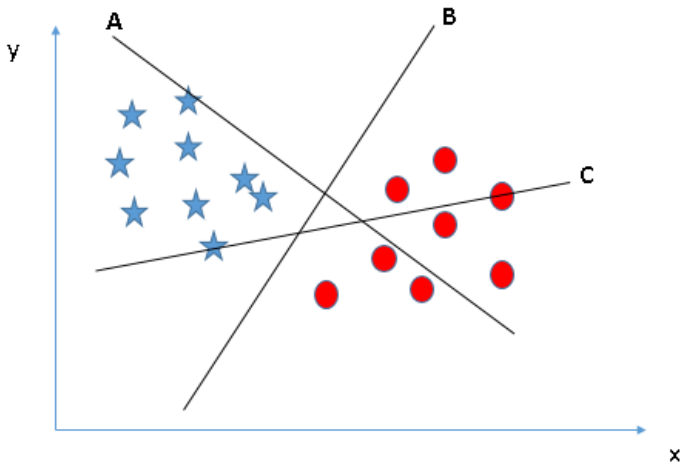
Analytics Vidhya

Hyperplanes in SVM

A hyperplane is a plane that differentiates the two classes in SVM. Now the burning question is “How can we identify the right hyper-plane?” Don’t worry, it’s not as hard as you think!

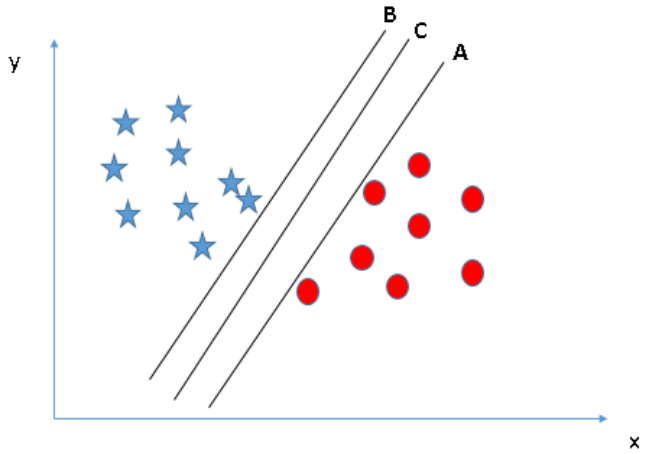
Let’s understand:

Identify the right hyper-plane(Scenario 1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

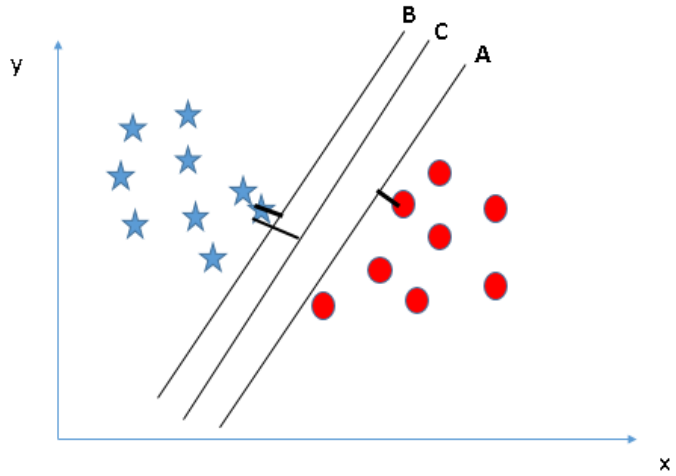


You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

Identify the right hyper-plane (Scenario 2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?



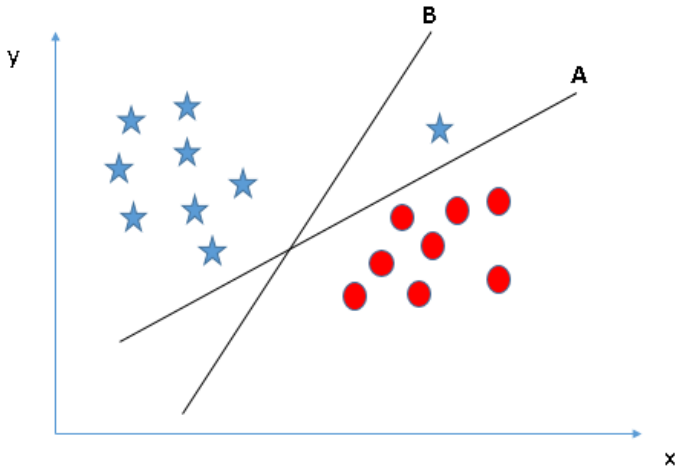
Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Let's look at the below snapshot:



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is a high chance of miss-classification.

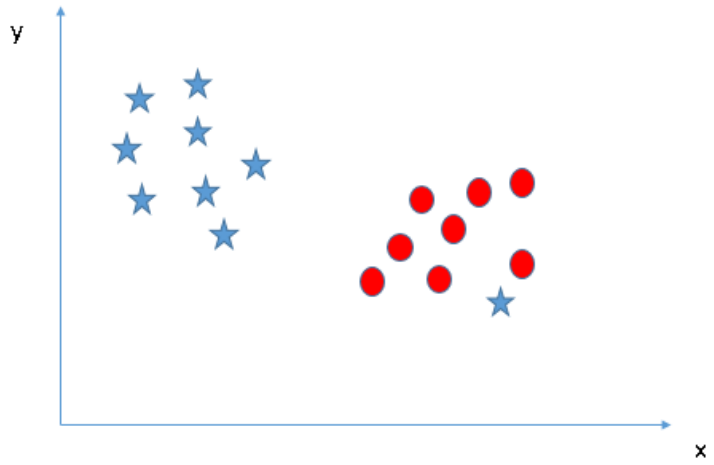
Identify the right hyper-plane(Scenario 3):

Hint: Use the rules as discussed in the previous section to identify the right hyper-plane

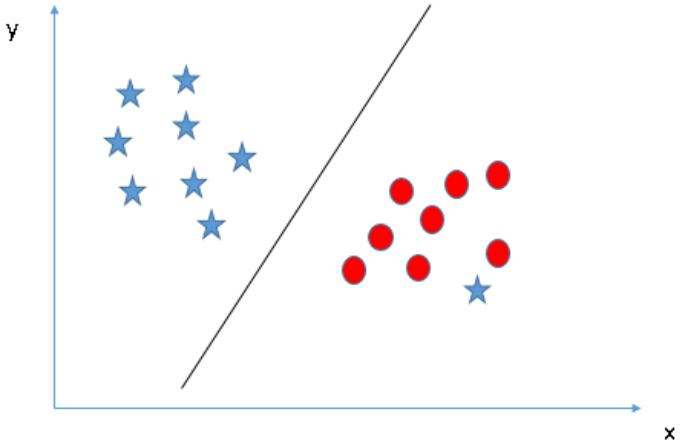


Some of you may have selected the hyper-plane **B** as it has a higher margin compared to **A.** But, here is the catch - SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

Can we classify two classes (Scenario 4)?: Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.

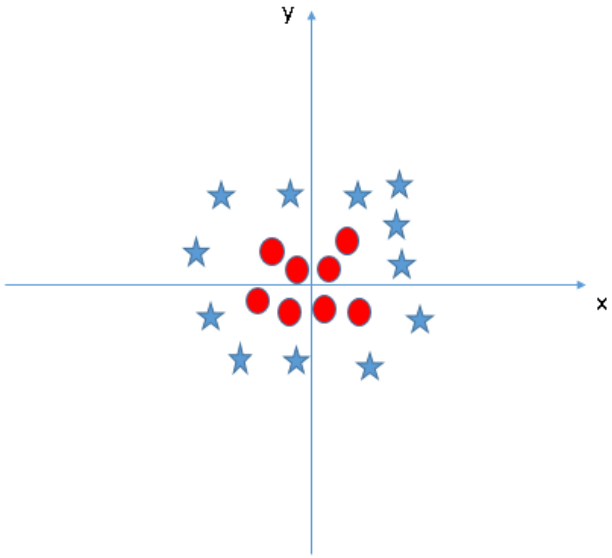


As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyperplane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.



Find the hyper-plane to segregate to classes (Scenario-5): In the scenario below, we can't have linear hyperplane between the two classes, so how does SVM classify these two classes? Till now,

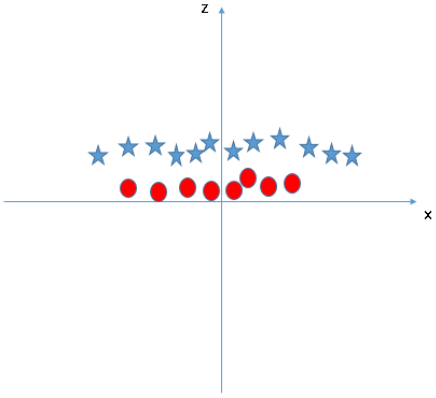
we have only looked at the linear hyperplane.



SVM can solve this problem. Easily! It solves this problem by introducing an additional feature. Here, we will add a new feature:

$$z = x^2+y^2.$$

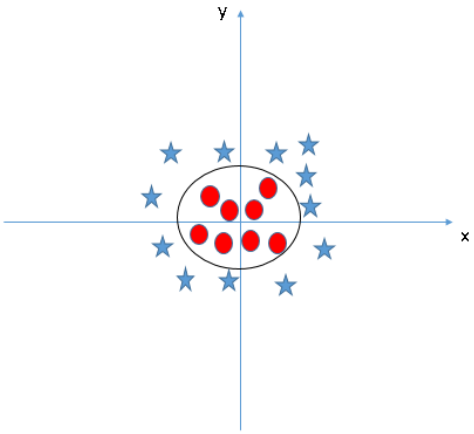
Now, let's plot the data points on axis x and z:



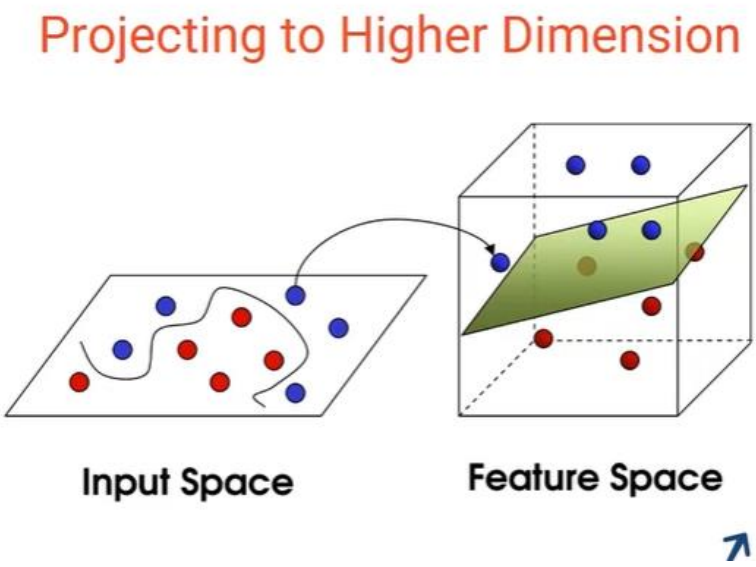
In the above plot, the points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to a lower value of z. The star relatively away from the origin results to higher value of z.

When we look at the hyper-plane in the original input space, it looks like a circle:

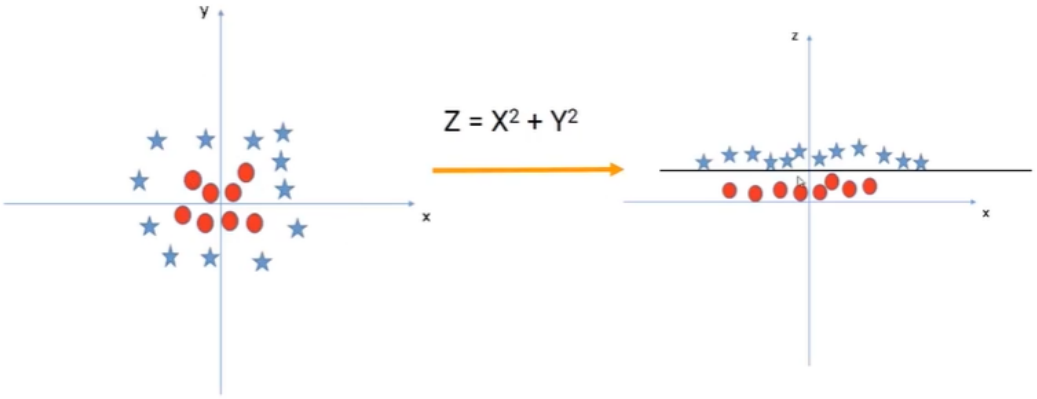


Types of Kernels used in SVM



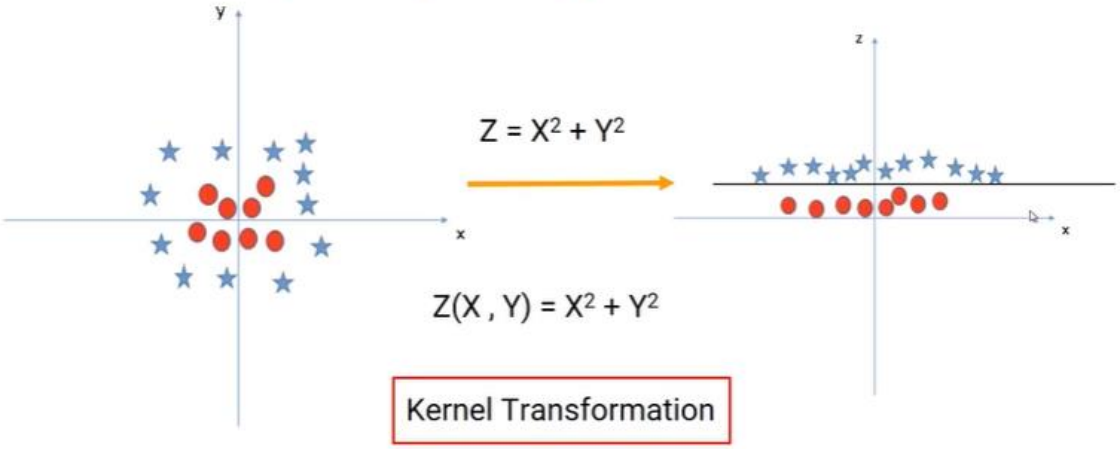
https://en.wikipedia.org/wiki/Radial_basis_function_kernel

Projecting to Higher Dimension



Analytics Vidhya
Learn everything about analytics

Projecting to Higher Dimension



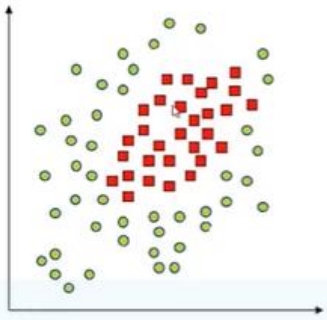
$Z(X , Y) = X^2 + Y^2$

Kernel Transformation

Projecting to Higher Dimension

$$Z(X, Y) = \exp\left(-\frac{\| X - Y \|^2}{2\sigma^2}\right)$$

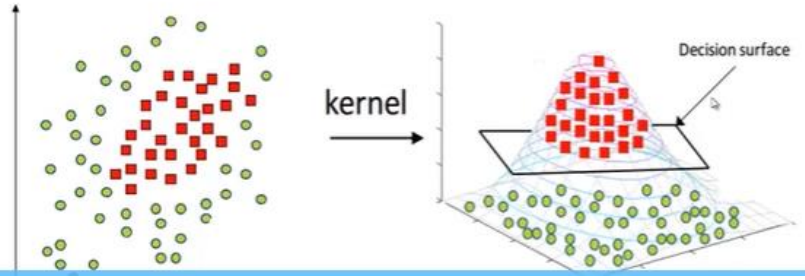
'Rbf' radial basis function/ Gaussian Kernel

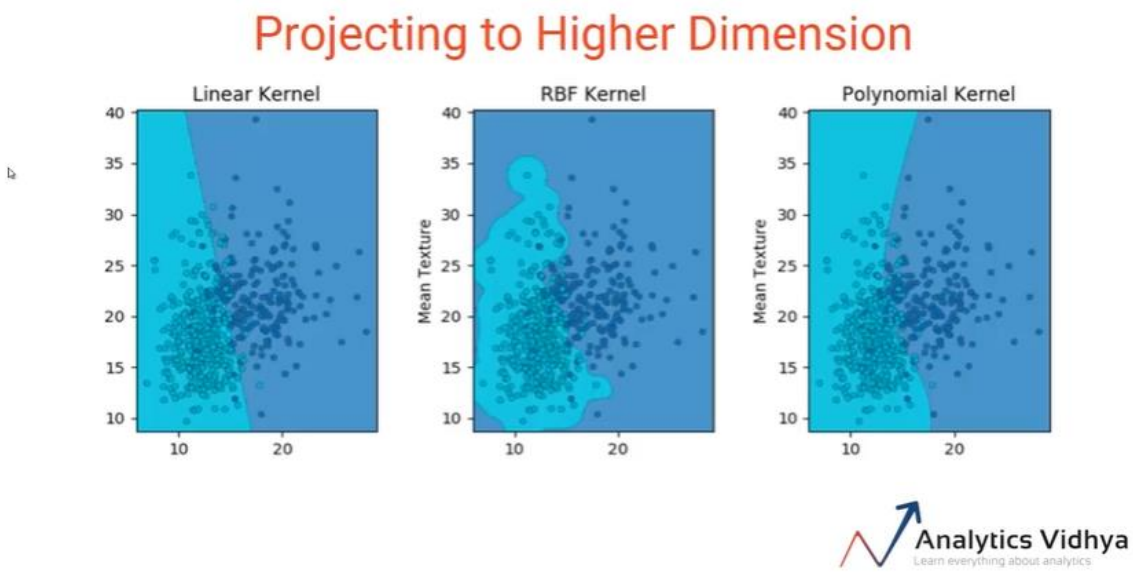


Projecting to Higher Dimension

$$Z(X, Y) = \exp\left(-\frac{\| X - Y \|^2}{2\sigma^2}\right)$$

'Rbf' radial basis function/ Gaussian Kernel





How to implement Support Vector Machine Classifier in R?

The e1071 package in R is used to create Support Vector Machines with ease. It has helper functions as well as code for the Naive Bayes Classifier. The creation of a support vector machine in R and Python follow similar approaches, let's take a look now at the following code:

```
#Import Library
require(e1071) #Contains the SVM
Train <- read.csv(file.choose())
Test <- read.csv(file.choose())
# there are various options associated with SVM training; like changing kernel, gamma and C value.

# create model
model <- svm(Target~Predictor1+Predictor2+Predictor3,data=Train,kernel='linear',gamma=0.2,cost=100)

#Predict Output
preds <- predict(model,Test)
table(preds)
```

In R, SVMs can be tuned in a similar fashion as they are in Python. Mentioned below are the respective parameters for e1071 package:

- The kernel parameter can be tuned to take "Linear","Poly","rbf" etc.
- The gamma value can be tuned by setting the "Gamma" parameter.

- The C value in Python is tuned by the “Cost” parameter in R.

Drawbacks of SVM

Enable fullscreen

Though Support Vector Machine is a popular classification algorithm, there are certain restrictions we need to keep in mind before using it:

- SVM doesn’t perform well when we have a large data set because the required training time is higher
- It also doesn’t perform very well when the data set has more noise i.e. target classes are overlapping
- As we have seen in the earlier lessons, there are quite a few hyperparameters we need to tune in SVM.Thus, we might have to run the classifier many times with different values for the hyperparameters before getting the right combination. Only then, we can get the best results out of the SVM algorithm.