Upgrad Assignment – Support Vector Machines

Assignment – Part 2

submitted by : Sweta Singh

Question 1

How is Soft Margin Classifier different from Maximum Margin Classifier?

Answer

Before we explain the difference between Soft Margin Classifier , lets first understand what is a Maximum Margin Classifier is:

The Maximal-Margin Classifier is a hypothetical classifier that best explains how SVM works in practice.

The numeric input variables (x) in your data (the columns) form an n-dimensional space. For example, if you had two input variables, this would form a two-dimensional space.

A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1. In two-dimensions you can visualize this as a line and let's assume that all of our input points can be completely separated by this line. For example:

$$c + (a * x1) + (b * x2) = 0$$

Where the coefficients (a and b) that determine the slope of the line and the intercept (c) are found by the learning algorithm, and x1 and x2 are the two input variables.

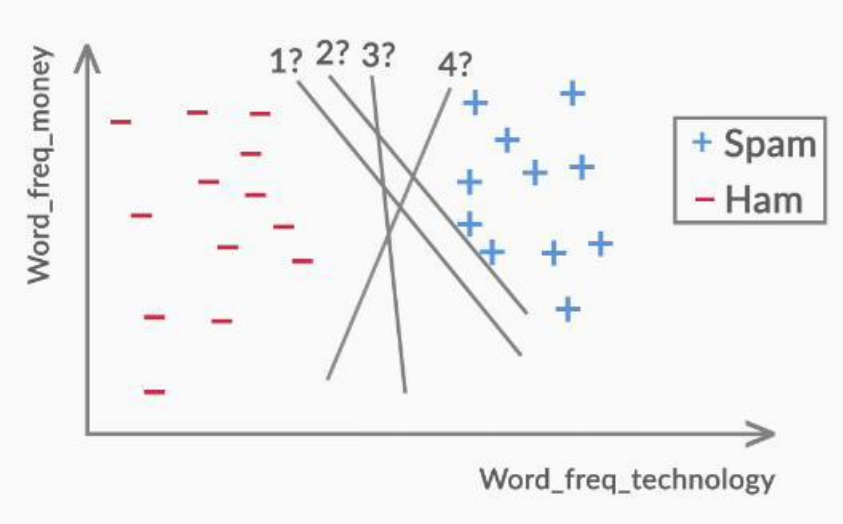
You can make classifications using this line. By plugging in input values into the line equation, you can calculate whether a new point is above or below the line.

- Above the line, the equation returns a value greater than 0 and the point belongs to the first class (class 0).
- Below the line, the equation returns a value less than 0 and the point belongs to the second class (class 1).
- A value close to the line returns a value close to zero and the point may be difficult to classify.

- If the magnitude of the value is large, the model may have more confidence in the prediction.
  The distance between the line and the closest data points is referred to as the margin. The best or optimal line that can separate the two classes is the line that as the largest margin. This is called the Maximal-Margin hyperplane.

  The margin is calculated as the perpendicular distance from the line to only the closest points. Only these points are relevant in defining the line and in the construction of the classifier. These points are called the support vectors. They support or define the hyperplane.

  The hyperplane is learned from training data using an optimization procedure that maximizes the margin.



  3rd line(Hyperplane) should be considered as a maximal margin classifier in the above figure.

  Now let’s understand Soft Margin Classifier:

  The maximal margin line (hyperplane), although it separates the two classes perfectly, is very sensitive to the training data. This means that the Maximal Margin Classifier will perform perfectly on the training data set. But on the unseen data, it may perform poorly. Also, there are cases where the classes cannot be perfectly separated.

  Thus, the soft margin classifier helps in solving this problem.

The Support Vector Classifier essentially allows certain points to be deliberately misclassified. By doing this, it is able to classify most of the points correctly in the unseen data and is also more robust.

The Support Vector Classifier is also called the Soft Margin Classifier because instead of searching for the margin that exactly classifies each and every data point to the correct class, the Soft Margin Classifier allows some observations to fall on the wrong side. The points which are close to the hyperplane are only considered for constructing the hyperplane and those points are called support vectors.

Support vector classifier works well when the data is partially intermingled (i.e. the data can be classified by minimal misclassifications). But what if the distribution looks completely intermingled and follows some pattern, something like the circular distribution of labels (+ and -), as shown in figure below.

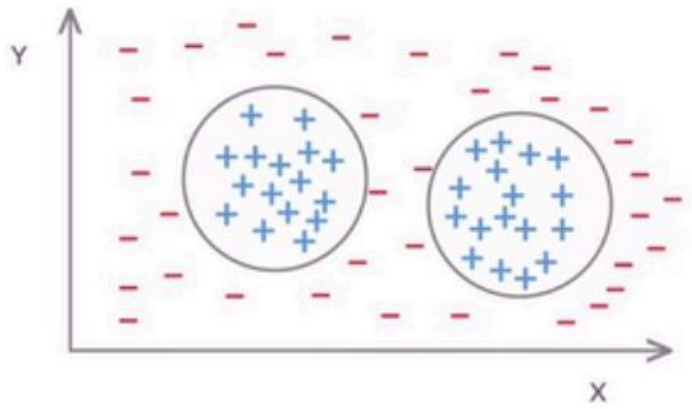
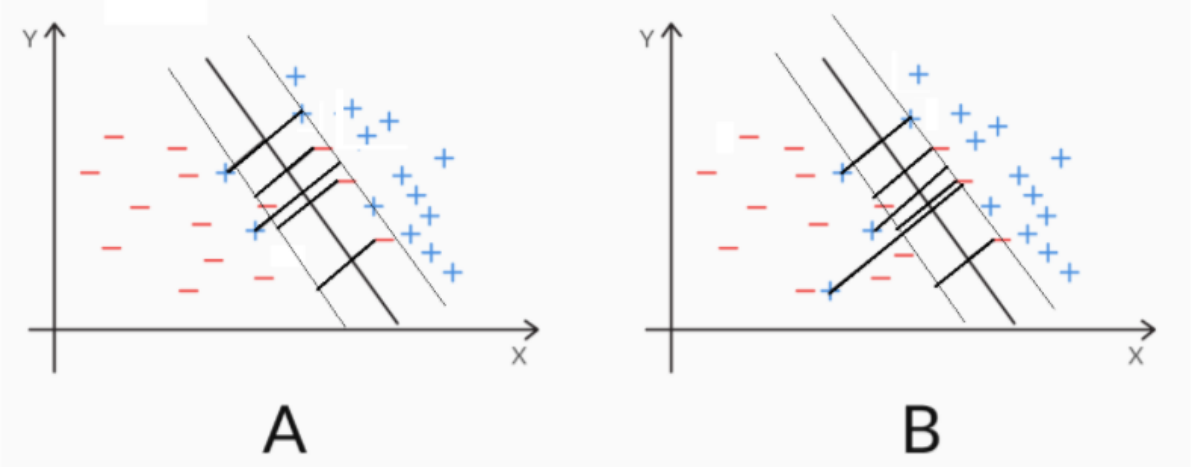


Figure 6: Intermingled Data

Obviously, the Support Vector Classifier can't classify the data above correctly, because it divides the data set into two halves, which misclassifies a lot of data points. But it doesn't mean that this problem cannot be solved.

Like the Maximal Margin Classifier, the Support Vector Classifier also maximises the margin; but the margin, here, will allow some points to be misclassified, as shown in figure below.



Question 2

What does the slack variable Epsilon (ε) represent?

Answer

A slack variable is used to control misclassifications. It tells you where an observation is located relative to the margin and hyperplane.

There are three different conditions applied if any new data point comes into play. Suppose you draw a Support Vector Classifier in such a way that it doesn't allow any misclassification, i.e. Epsilon(ϵ) = 0, then each observation is on the correct side of the margin as shown in figure below.

Slack Variable

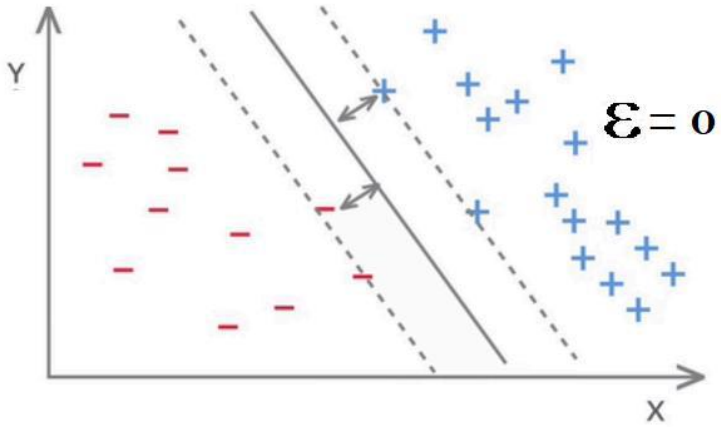


Figure 8: Slack Variable

But if you draw a Support Vector Classifier in such a way that it only violates the margin, i.e. 0< Epsilon(ϵ) < 1, the observations classify correctly as shown in figure below.

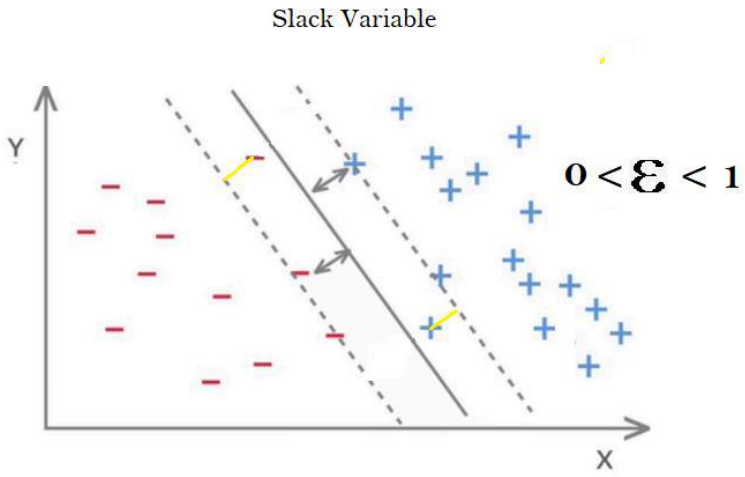


Figure 9: Slack Variables

But if the data points violate the hyperplane, i.e. Epsilon(ϵ) > 1, then the observation is on the wrong side of the hyperplane, as shown in figure below.

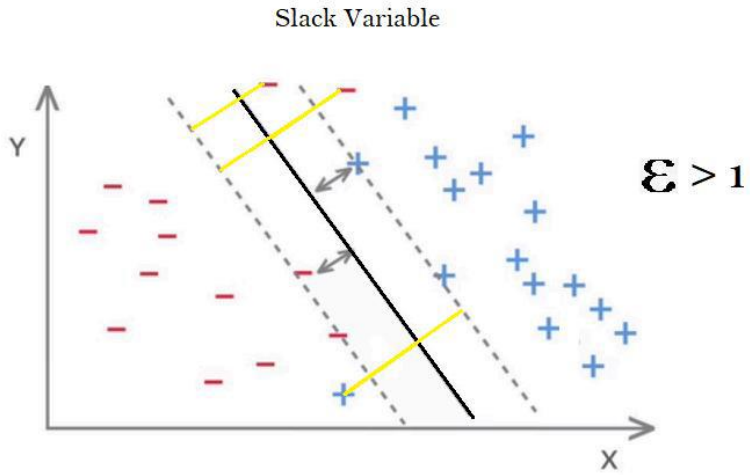


Figure 10: Slack Variable

So you can see that:

- Each data point has a slack value associated to it, according to where the point is located.
- The value of slack lies between 0 and +infinity.

Lower values of slack are better than higher values (slack = 0 implies a correct classification, but slack > 1 implies an incorrect classification, whereas slack within 0 and 1 classifies correctly but violates the margin)

Question 3

How do you measure the cost function in SVM? What does the value of C signify?

Answer

The parameter cost of misclassification (C) represents the cost of violations to the margin and the hyperplane. Thus, the parameter C is called the tuning parameter — one of the hyperparameters in SVM.

Cost of misclassification is greater than or equal to the summation of all the epsilons of each data point, and is denoted by cost or 'C'.

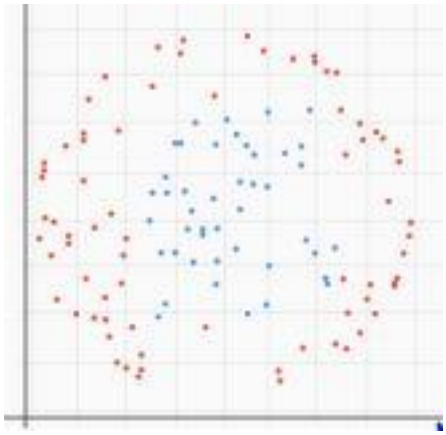
$$\sum \epsilon_i \leq C.$$

Once you understand the notion of the slack variable, you can easily compare the two Support Vector Classifiers. You can measure the summation of all the epsilons(ϵ) of both the hyperplanes and choose the best one that gives you the least sum of epsilons(ϵ). The summation of all the epsilons of each data point is denoted by cost or 'C', i.e.

When C is large, the slack variables can be large, i.e. you allow a larger number of data points to be misclassified or to violate the margin. So you get a hyperplane where the margin is wide and misclassifications are allowed. In this case, the model is flexible, more generalizable, and less likely to overfit. In other words, it has a high bias.

On the other hand, when C is small, you force the individual slack variables to be small, i.e. you do not allow many data points to fall on the wrong side of the margin or the hyperplane. So, the margin is narrow and there are few misclassifications. In this case, the model is less flexible, less generalizable, and more likely to overfit. In other words, it has a high variance.

Question 4



Given the above dataset where red and blue points represent the two classes, how will you use SVM to classify the data?

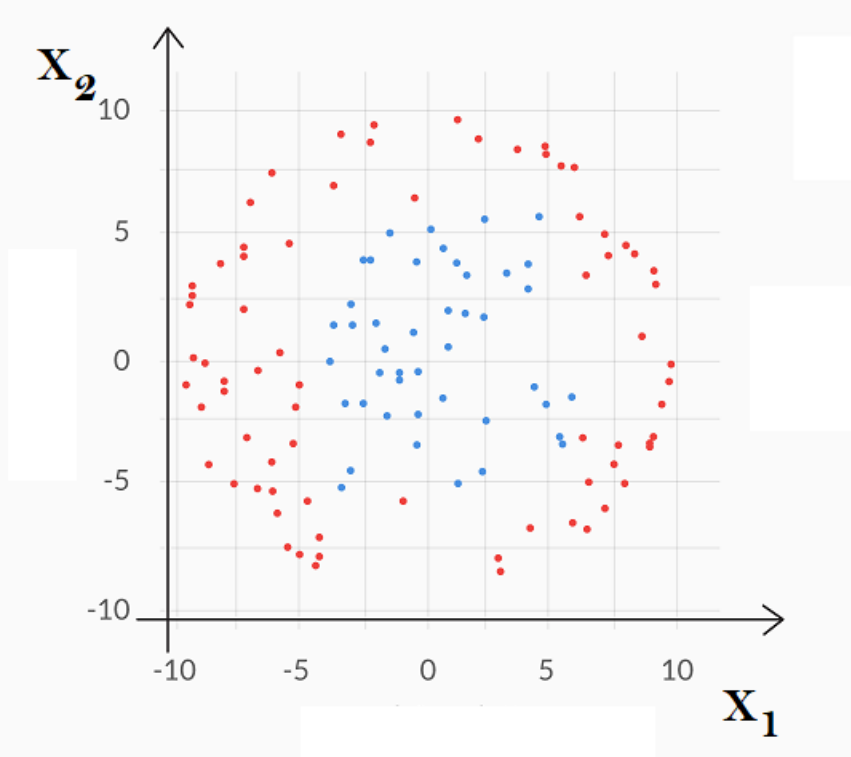
Answer

Support Vector Machine (SVM) is an advanced machine learning technique which has a unique way of solving complex problems such as image recognition, face detection, voice detection etc.

So far, we have learnt about hyperplanes, the Maximal Margin Classifier, and the Support Vector Classifier. All of these are linear models (since they use linear hyperplanes to separate the classes). However, many real-world data sets are not separable by linear boundaries.

For instance, if you have a data as shown in the figure below, SVMs can handle it easily and that's how SVM distinguishes from logistic regression.

Figure



You'll agree that it is not possible to imagine a linear hyperplane (a line in 2D) that separates the red and blue points reasonably well. Thus, you need to tweak the linear SVM model and enable it to incorporate nonlinearity in some way. Kernels serve this purpose — they enable the linear SVM model to separate nonlinearly separable data points.

You can transform nonlinear boundaries to linear boundaries by applying certain functions to the original attributes. The original space (X, Y) is called the attribute space, and the transformed space (X', Y') is called the feature space.

To convert this data set into a linearly separable one, a simple transformation into a new feature space (X', Y') can be made. For now, don't worry about the math behind the transformation. You may almost never need to manually transform data sets. Just assume that some appropriate transformation from (X, Y) to (X', Y') can make the data linearly separable.

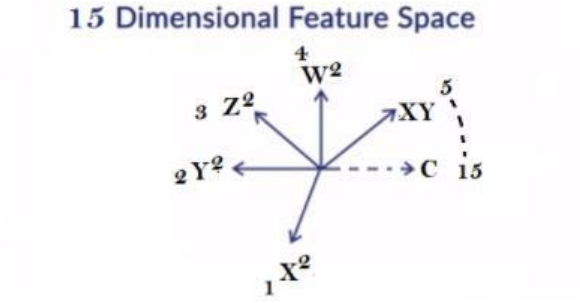
In the original attribute space, notice that the observations are distributed in a circular fashion. This gives you a hint that the transformation should convert the circular distribution to a linear distribution.

- word_freq_office(X′)=(word_freq_office(X)−a)2
- word_freq_office′(Y′)=(word_freq_office(Y)−b)2

Feature Transformation

The process of transforming the original attributes into a new feature space is called ‘feature transformation’. However as the number of attributes increases, there is an exponential increase in the number of dimensions in the transformed feature space. Suppose you have four variables in your data set, then considering only a polynomial transformation with degree 2, you end up making 15 features in the new feature space, as shown in the figure below.

$$a_1X^2 + a_2Y^2 + a_3Z^2 + a_4W^2 + a_5XY + a_6YZ + a_7ZW + a_8WX + a_9WY + a_{10}ZX + a_{11}X + a_{12}Y + a_{13}Z + a_{14}W + C = 0$$

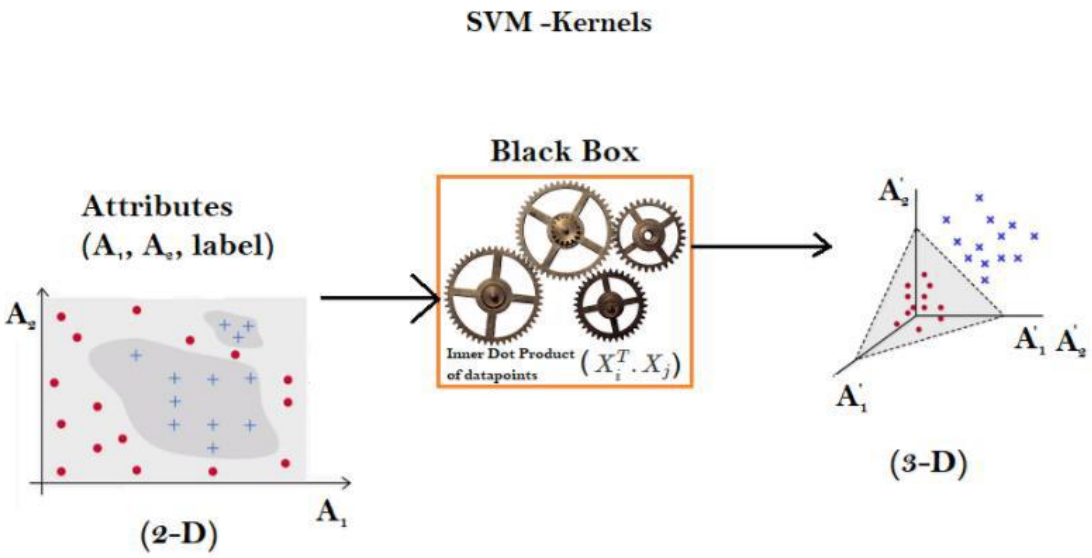


Kernel Trick

As feature transformation results in large number of features, it makes the modelling (i.e. the learning process) computationally expensive. The use of kernel resolves this issue. The key fact that makes the kernel trick possible is that to find a best fit model, the learning algorithm only needs the inner products of the observations (XTi.Xj). It never uses the individual data points X1, X2 etc. in silo.

Think of a kernel as a black box, as shown in the figure below. The attributes are passed on the black box, and it returns the linear boundaries for the classification of the nonlinear data implicitly.

Figure 14: Black Box



Kernel functions use this fact to bypass the explicit transformation process from the attribute space to the feature space, and rather do it implicitly. The benefit of implicit transformation is that now you do not need to:

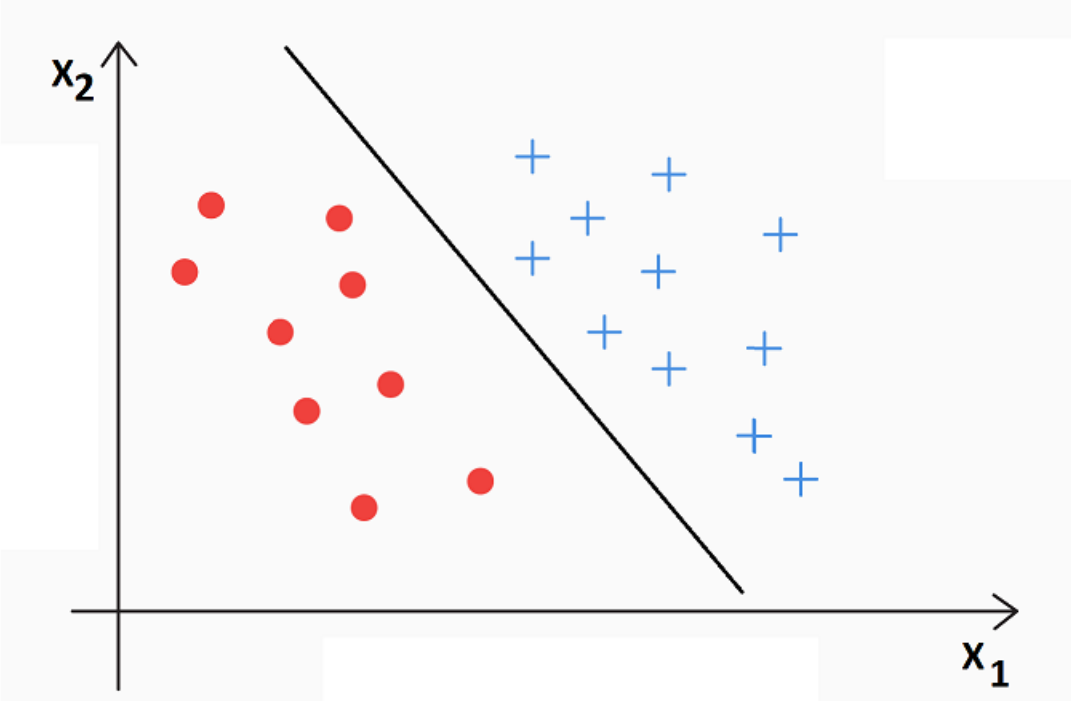
- Manually find the mathematical transformation needed to convert a nonlinear to a linear feature space
- Perform computationally heavy transformations

In practice, you only need to know that kernels are functions which help you transform non-linear datasets. Given a dataset, you can try various kernels, and choose the one that produces the best model. The three most popular types of kernel functions are:

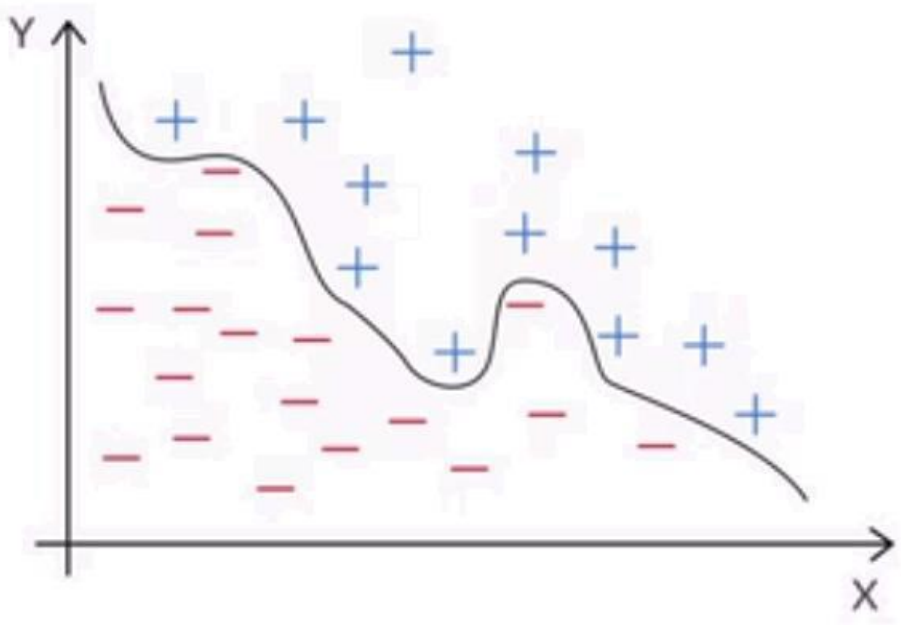
- The linear kernel: This is the same as the support vector classifier, or the hyperplane, without any transformation at all
- The polynomial kernel: It is capable of creating nonlinear, polynomial decision boundaries

- The radial basis function (RBF) kernel: This is the most complex one, which is capable of transforming highly nonlinear feature spaces to linear ones. It is even capable of creating elliptical (i.e. enclosed) decision boundaries

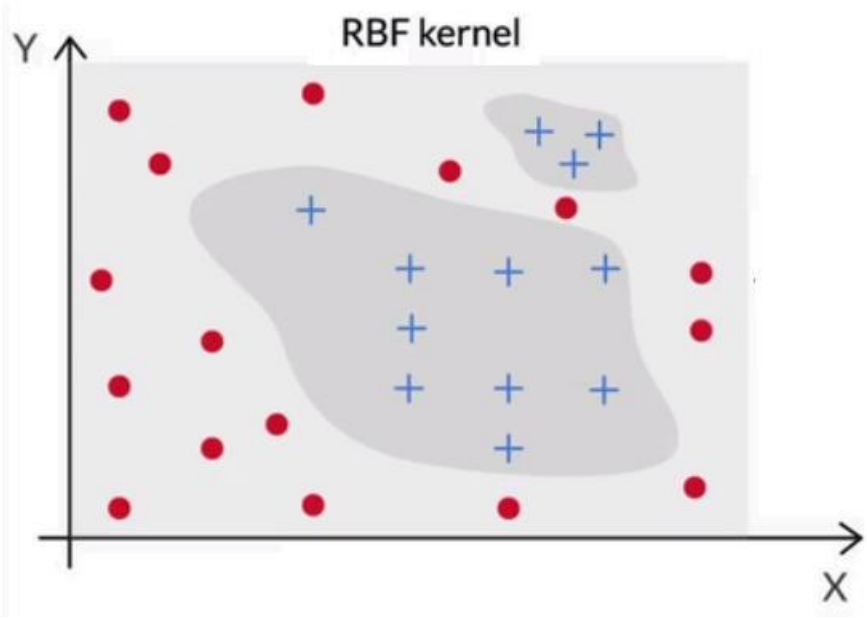
The three types of kernel functions shown in the figures below represent the typical decision boundaries they are capable of creating. Notice that the linear kernel is same as the vanilla hyperplane, the polynomial kernel can produce polynomial shaped nonlinear boundaries, and the RBF kernel can produce highly nonlinear, ellipsoid shaped boundaries.



Fig(1) Linear Kernel



Fig(2) Ploynomial Kernel



Fig(3) RBF Kernel

Question 5

What do you mean by feature transformation?

Answer

Feature Transformation

The process of transforming the original attributes into a new feature space is called ‘feature transformation’. However as the number of attributes increases, there is an exponential increase in the number of dimensions in the transformed feature space. Suppose you have four variables in your data set, then considering only a polynomial transformation with degree 2, you end up making 15 features in the new feature space, as shown in the figure below.

$$a_1X^2 + a_2Y^2 + a_3Z^2 + a_4W^2 + a_5XY + a_6YZ + a_7ZW + a_8WX + a_9WY + a_{10}ZX + a_{11}X + a_{12}Y + a_{13}Z + a_{14}W + C = 0$$