Kernel Method by (N)

Groupe 10
Abdou NIANG
Annan METIKI
Gilda BANSIMBA
Mouhamadou Bamba DIOP

Plan

The presentation is organized as follows:

- I-Introduction
- II-Support Vector Machines intuition
- III-Kernel Tricks
- IV-Multiclass Classification using Support Vector Machine
- conclusion

I-Introduction

Support Vector Machines (SVMs in short) are machine learning algorithms that are used for classification and regression purposes. SVMs are one of the powerful machine learning algorithms for classification, regression and outlier detection purposes. An SVM classifier builds a model that assigns new data points to one of the given categories.

I-Introduction

SVMs can be used for linear classification purposes. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using the kernel trick. It enable us to implicitly map the inputs into high dimensional feature spaces.

Now, we should be familiar with some SVM terminology.

Hyperplane

A hyperplane is a decision boundary which separates between given set of data points having different class labels. The SVM classifier separates data points using a hyperplane with the maximum amount of margin. This hyperplane is known as the maximum margin hyperplane and the linear classifier it defines is known as the maximum margin classifier.

Support Vectors

Support vectors are the sample data points, which are closest to the hyperplane. These data points will define the separating line or hyperplane better by calculating margins.

Margin

A margin is a separation gap between the two lines on the closest data points. It is calculated as the perpendicular distance from the line to support vectors or closest data points. In SVMs, we try to maximize this separation gap so that we get maximum margin. The following diagram illustrates these concepts visually.

Margin in SVM

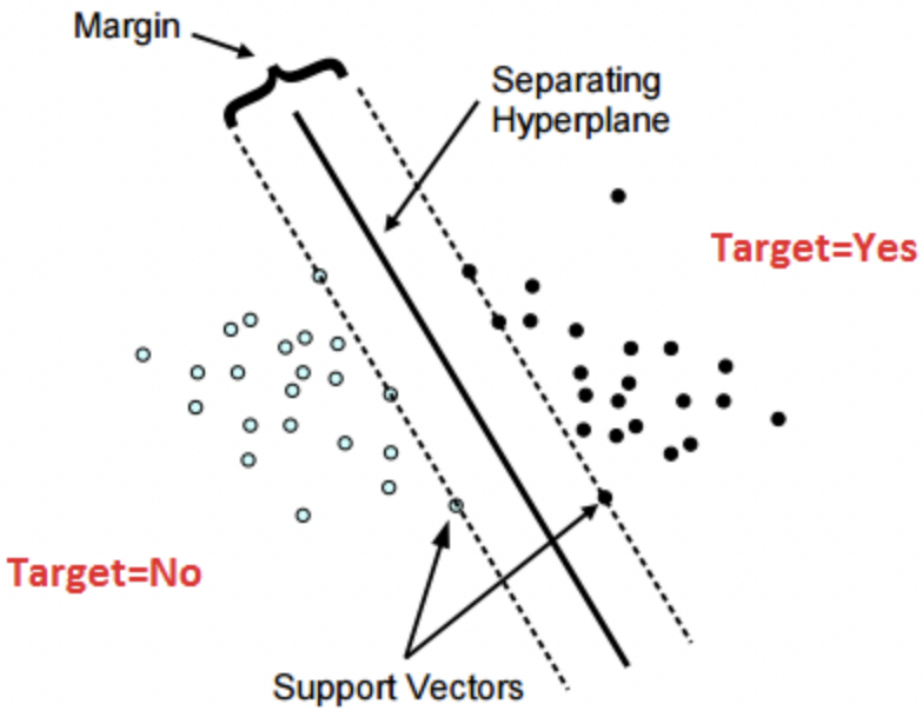

Figure: Fully connected neural network

SVM Under the hood

In SVMs, our main objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum margin hyperplane in the following 2 step process

Generate hyperplane which segregates the classes in the best possible way. There are many hyperplane that might classify the data. We should look for the best hyperplane that represents the largest separation, or margin, between the two classes.

So, we choose the hyperplane so that distance from it to the support vectors on each side is maximized. If such a hyperplane exists, it is known as the maximum margin hyperplane and the linear classifier it defines is known as a maximum margin classifier.

Maximum margin hyperplane

The following diagram illustrates the concept of maximum margin and maximum margin hyperplane in a clear manner.

Maximum margin hyperplane

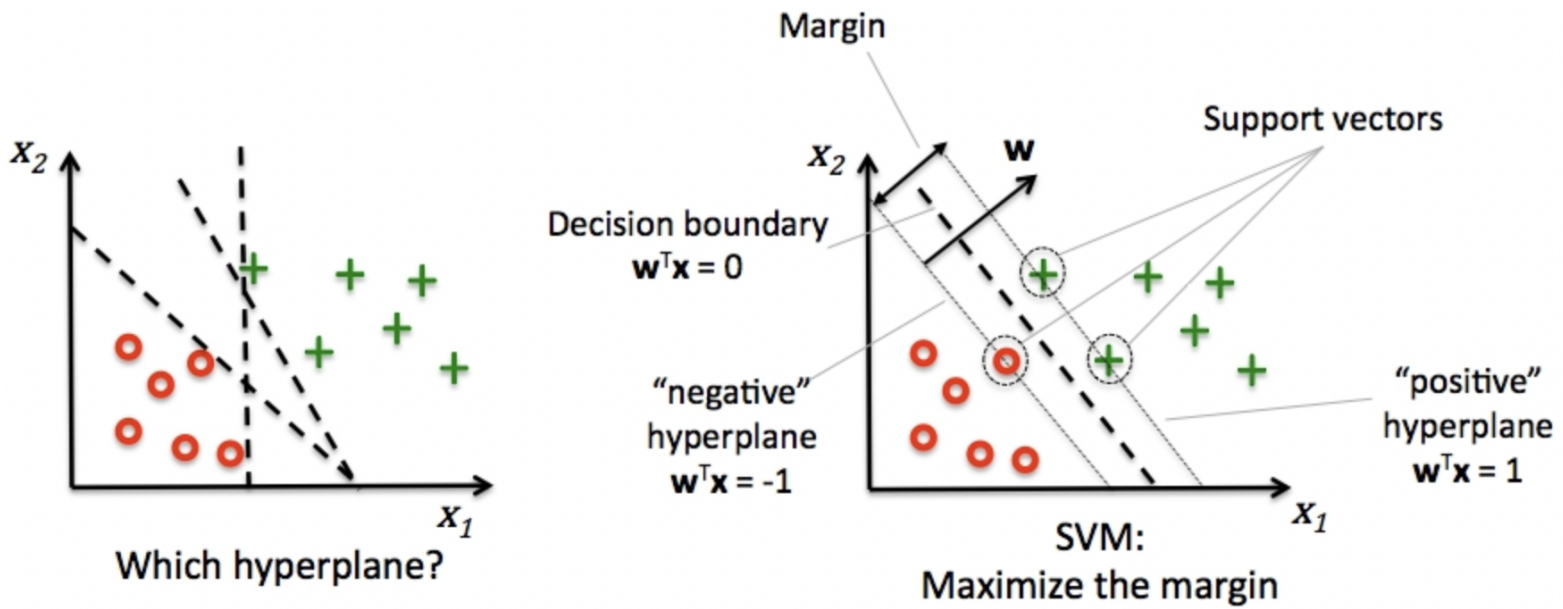

Figure: 1

Problem with dispersed datasets

Sometimes, the sample data points are so dispersed that it is not possible to separate them using a linear hyperplane. In such a situation, SVMs uses a kernel trick to transform the input space to a higher dimensional space as shown in the diagram below. It uses a mapping function to transform the 2-D input space into the 3-D input space. Now, we can easily segregate the data points using linear separation.

III-Kernel trick - transformation of input space to higher dimensional space

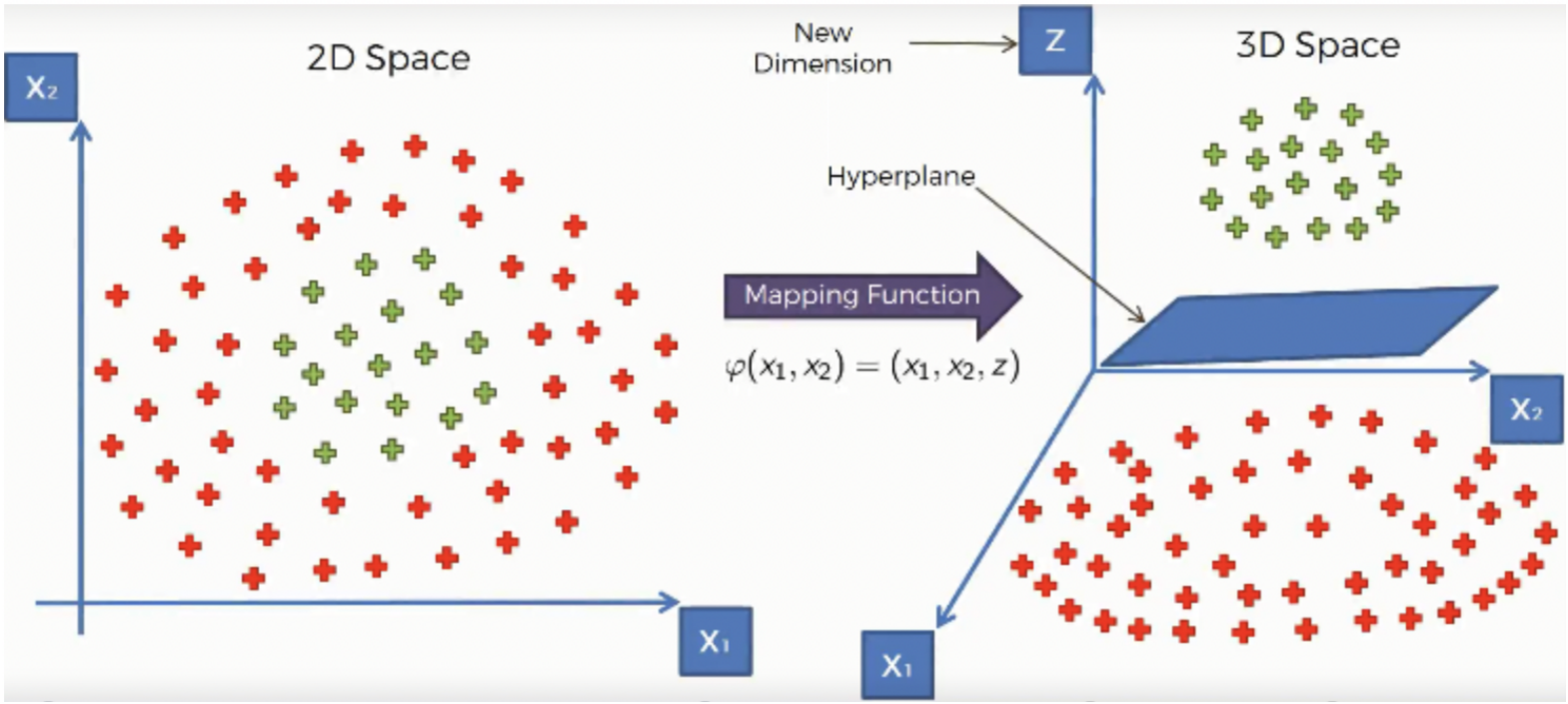

Figure: 2

Kernel trick

In practice, SVM algorithm is implemented using a kernel. It uses a technique called the kernel trick. In simple words, a kernel is just a function that maps the data to a higher dimension where data is separable. A kernel transforms a low-dimensional input data space into a higher dimensional space. So, it converts non-linear separable problems to linear separable problems by adding more dimensions to it. Thus, the kernel trick helps us to build a more accurate classifier. Hence, it is useful in non-linear separation problems.

kernel function

Kernel function is a function that takes as its input vector in the original space and returns the dot product of the vectors in the feature space is called a kernel function.

kernel function

More formally, if we have data $x, z \in X$ and a map $\phi : X \to \mathrm{R}^N$ then

$$k(x, z) = < \phi(x), \phi(z) >$$

is a kernel function

In linear kernel, the kernel function takes the form of a linear function as follows

$$K(x_i, x_j) = x_i.Tx_j$$

Linear kernel is used when the data is linearly separable. It means that data can be separated using a single line. It is one of the most common kernels to be used. It is mostly used when there are large number of features in a dataset. Linear kernel is often used for text classification purposes.

Training with a linear kernel is usually faster, because we only need to optimize the C regularization parameter. When training with other kernels, we also need to optimize the parameter. So, performing a grid search will usually take more time. Linear kernel can be visualized with the following figure.

Linear Kernel

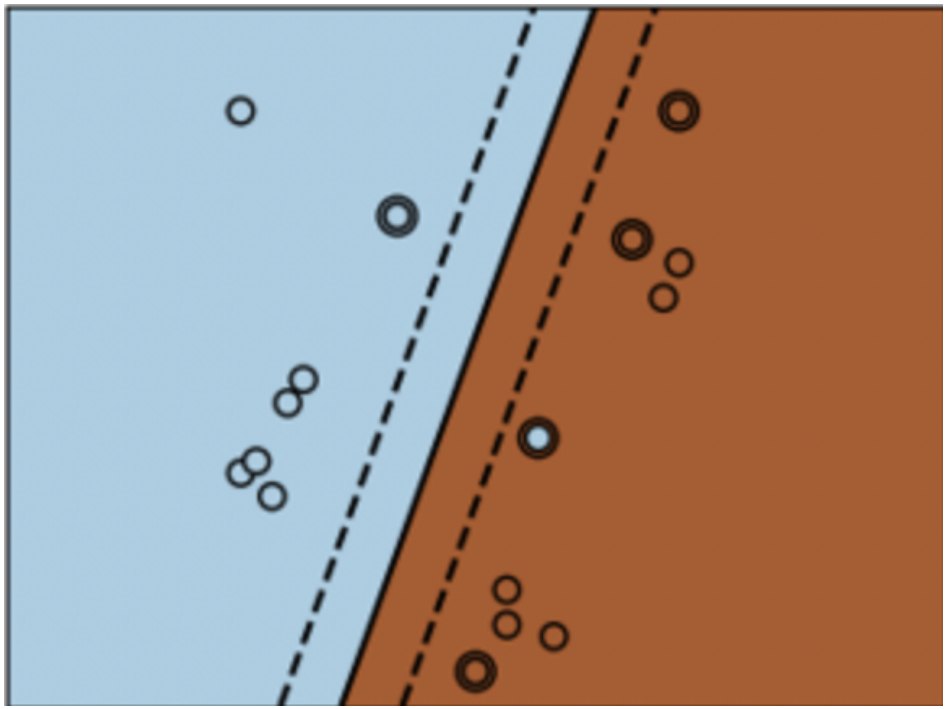

Figure: 3

Polynomial Kernel

Polynomial kernel represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables. The polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of the input samples.

Polynomial Kernel

For degree-d polynomials, the polynomial kernel is defined as follows:

$$K(x_i, x_j) = (\gamma x_i . T x_j + r)^d, \gamma > 0$$

Polynomial Kernel

Polynomial kernel is very popular in Natural Language Processing. The most common degree is d $= 2$ (quadratic), since larger degrees tend to overfit on NLP problems. It can be visualized with the following diagram.

Polynomial Kernel

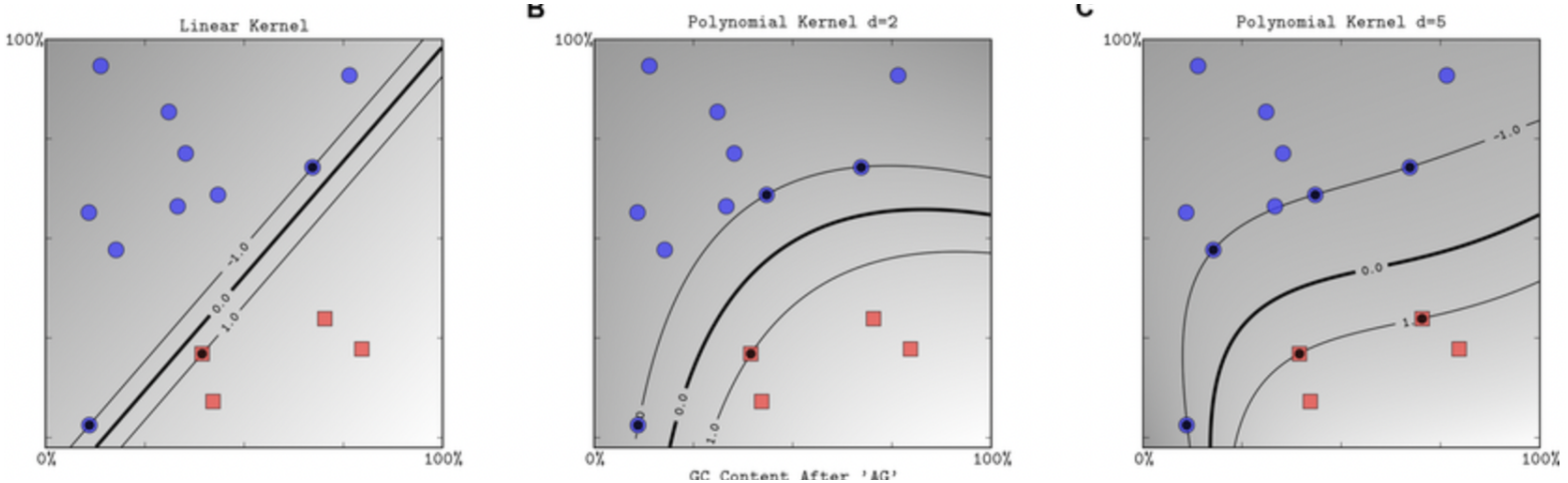

Figure: 4

Radial basis function kernel

Radial basis function kernel is a general purpose kernel. It is used when we have no prior knowledge about the data. The RBF kernel on two samples x_i and x_j is defined by the following equation

Radial Basis Function kernel

$$K(x_i, x_j) = \exp\{-\frac{||x_i - x_j||^2}{2\sigma^2}\}$$

The following diagram demonstrates the SVM classification with rbf kernel

SVM Classification with rbf kernel

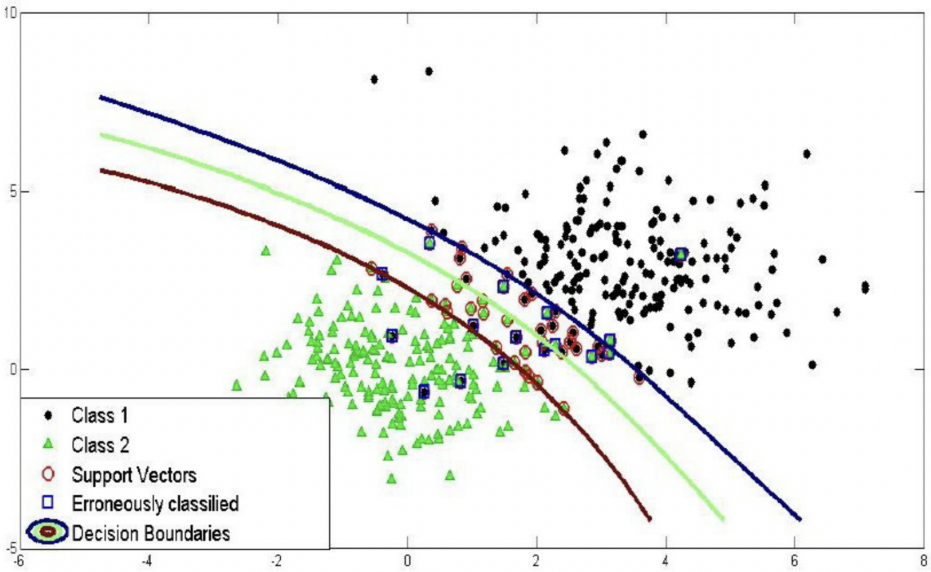

Figure: 5

Sigmoid kernel

Sigmoid kernel has its origin in neural networks. We can use it as the proxy for neural networks. Sigmoid kernel is given by the following equation.

$$K(x_i, x_j) = \tanh(\alpha x_i . T x_j + c)$$

Sigmoid kernel

Sigmoid kernel can be visualized with the following diagram:

Sigmoid kernel

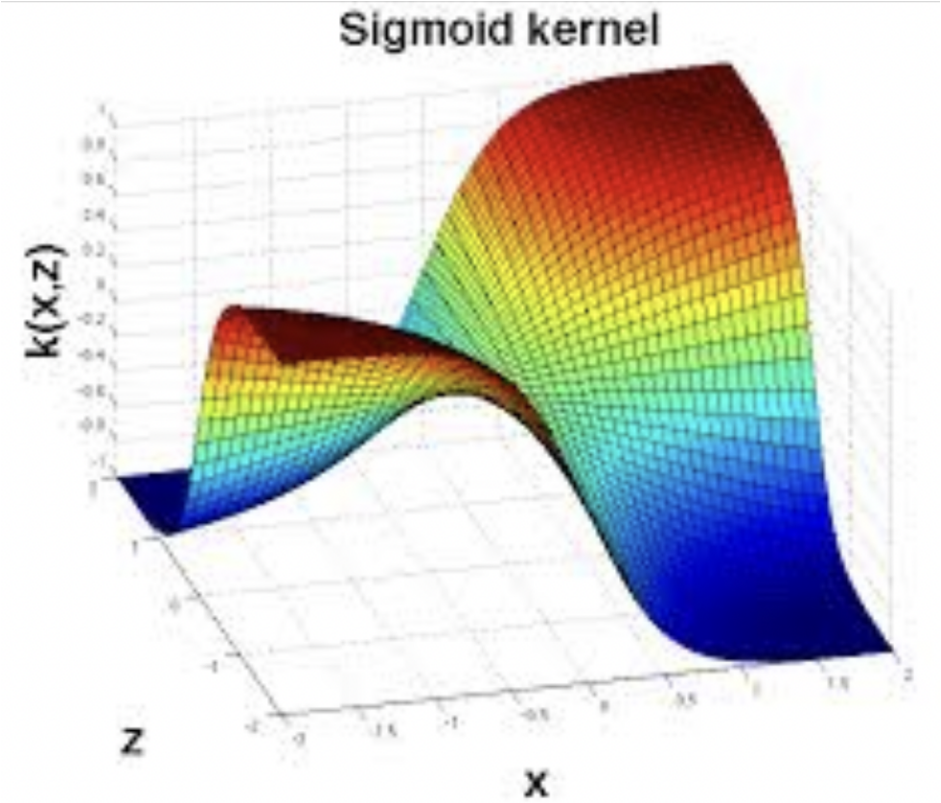

Figure: 6

Non-linear transformations

In 1-dimension, this data is not linearly separable, but after applying the transformation $\phi(x) = x^2$ and adding this second dimension to our feature space, the classes become linearly separable.

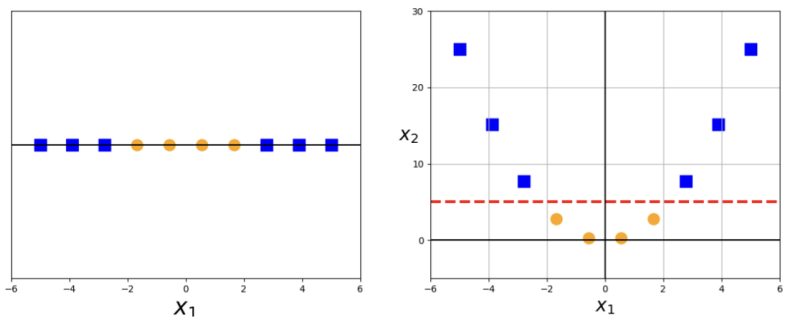

Figure: 7

Non-linear transformations

Now let's look at an example where our original data is not linearly separable in two dimensions. Here is our original data, which cannot be linearly separated.

Non-linear transformations

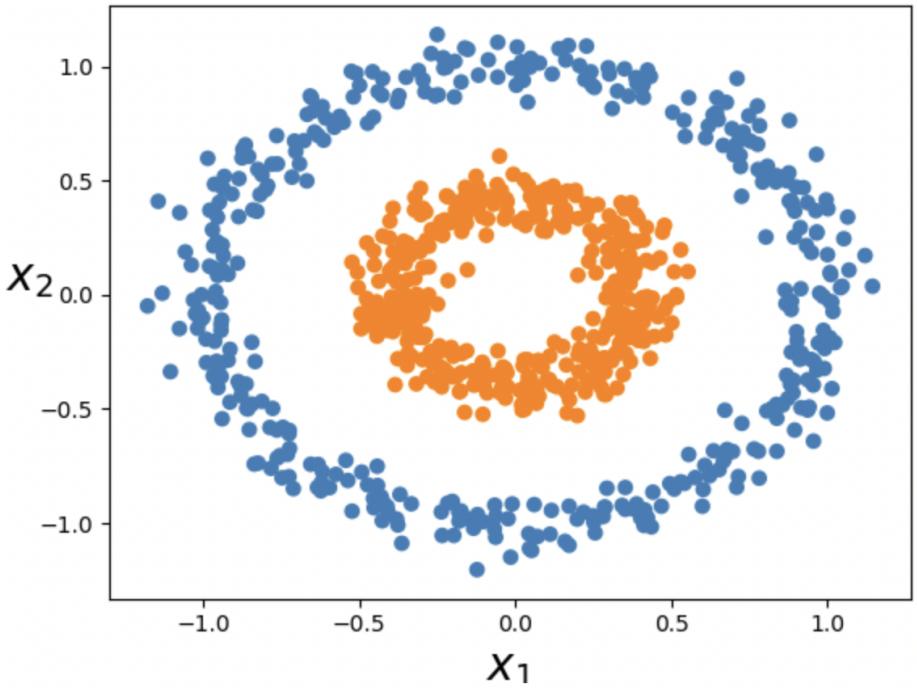

Figure: 8

After the following transformation:

Equation 5-8. Second-degree polynomial mapping

$$\phi(\mathbf{x}) = \phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} {x_1}^2 \\ \sqrt{2}x_1x_2 \\ {x_2}^2 \end{pmatrix}$$

Our data becomes linearly separable (by a 2-d plane) in 3-dimensions.

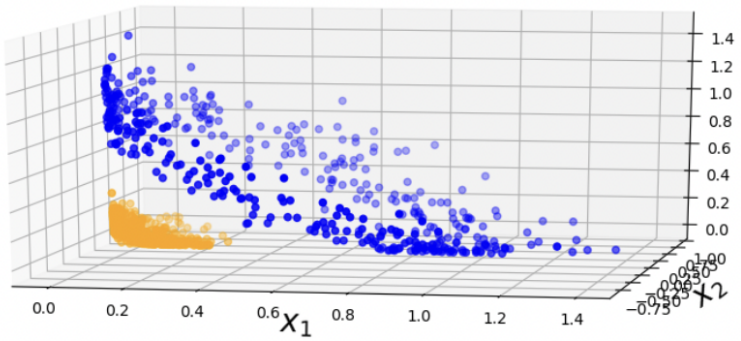

Figure: 9

The Kernel Trick

The ultimate benefit of the kernel trick is that the objective function we are optimizing to fit the higher dimensional decision boundary only includes the dot product of the transformed feature vectors. Therefore, we can just substitute these dot product terms with the kernel function, and we don't even use $\phi(x)$.

The Kernel Trick

- Solution of the dual problem gives us:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$$

- The decision boundary:

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

- The decision:

$$\hat{y} = \text{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

- Mapping to a feature space, we have the decision:

$$\hat{y} = \text{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)) + w_0 \right]$$

Remember, our data is only linearly separable as the vectors $\phi(x)$ in the higher dimensional space, and we are finding the optimal separating hyperplane in this higher dimensional space without having to calculate or in reality even know anything about $\phi(x)$.

IV-Multiclass Classification using Support Vector Machine

In its most simple type SVM are applied on binary classification, dividing data points either in 1 or 0. For multiclass classification, the same principle is utilized. The multiclass problem is broken down to multiple binary classification cases, which is also called one-vs-one.

Multiclass Classification using Support Vector Machine

The objective of using kernels in multi class svm is to find relations between pairs of data samples based on the kernel type. And the kernel matrix represents pairwise similarities in the training set. It represents information on the training set provided as input to the optimization process.

Given our input dataset with m samples, the kernel matrix is given by

$$\begin{pmatrix} k(x^{(1)}, x^{(1)}) & k(x^{(1)}, x^{(2)}) & \cdots & k(x^{(1)}, x^{(m)}) \\ k(x^{(2)}, x^{(1)}) & \ddots & & \vdots \\ \vdots & & & \\ k(x^{(m)}, x^{(1)}) & \cdots & & k(x^{(m)}, x^{(m)}) \end{pmatrix}$$

The number of classifiers necessary for one-vs-one multiclass classification can be retrieved with the following formula (with n being the number of classes):

$$\frac{n * (n - 1)}{2}$$

Multiclass Classification using Support Vector Machine

In the one-vs-one approach, each classifier separates points of two different classes and comprising all one-vs-one classifiers leads to a multiclass classifier.

For a better understanding of how the separation of different hyperplane can look like, the different kinds of kernel functions are visualized in the graphic below.

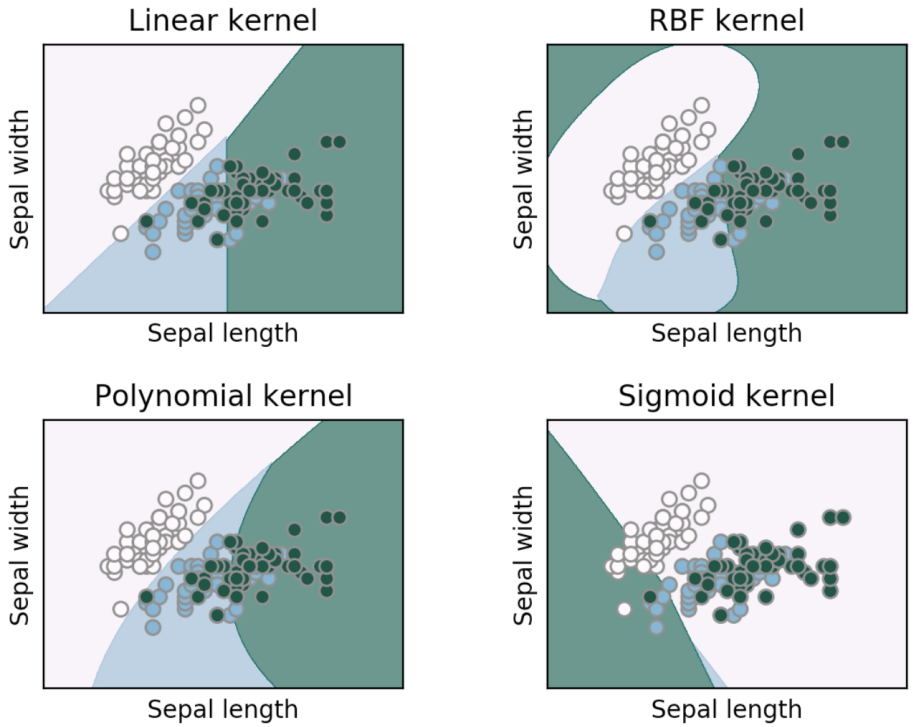

Figure: 10

Similarities between SVMs and Neural Networks: Parametric

SVM and neural networks are both parametric but for different reasons.
-For SVM the typical parameters are; soft-margin parameter (C), parameter of the kernel function (gamma).
-Neural networks also have parameters but it is a lot more than SVM. Some NN parameters are the number of layers and their size, number of training epochs, and the learning rate.

Embedding Non-Linearity:

Both the methods can embedded non-linear functions. SVM does this through the usage of kernel method. Neural Networks embed non-linearity using non-linear activation functions.

Comparable Accuracy:

-If both SVM and Neural Networks are trained in the same dataset, given the same training time, and the same computation power they have comparable accuracy.
-If neural networks are given as much computation power and training time as possible then it outperforms SVMs.

Conclusion

Practically, there are situations where the data is linearly separable and situations where it is not. In the latter, the use of kernels comes as an alternative by taking the data to a higher dimensional space in which one can find boundaries to separate the data with respect to their classes.

References

- https://www.mlstack.cafe/blog/support-vector-machine-interview-questions
- https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

Thank you for your kind Attention