Support Vector Machines

Classifies it correctly

Support Vector Machine

- Introduction

Introduction

Support Vector Machines

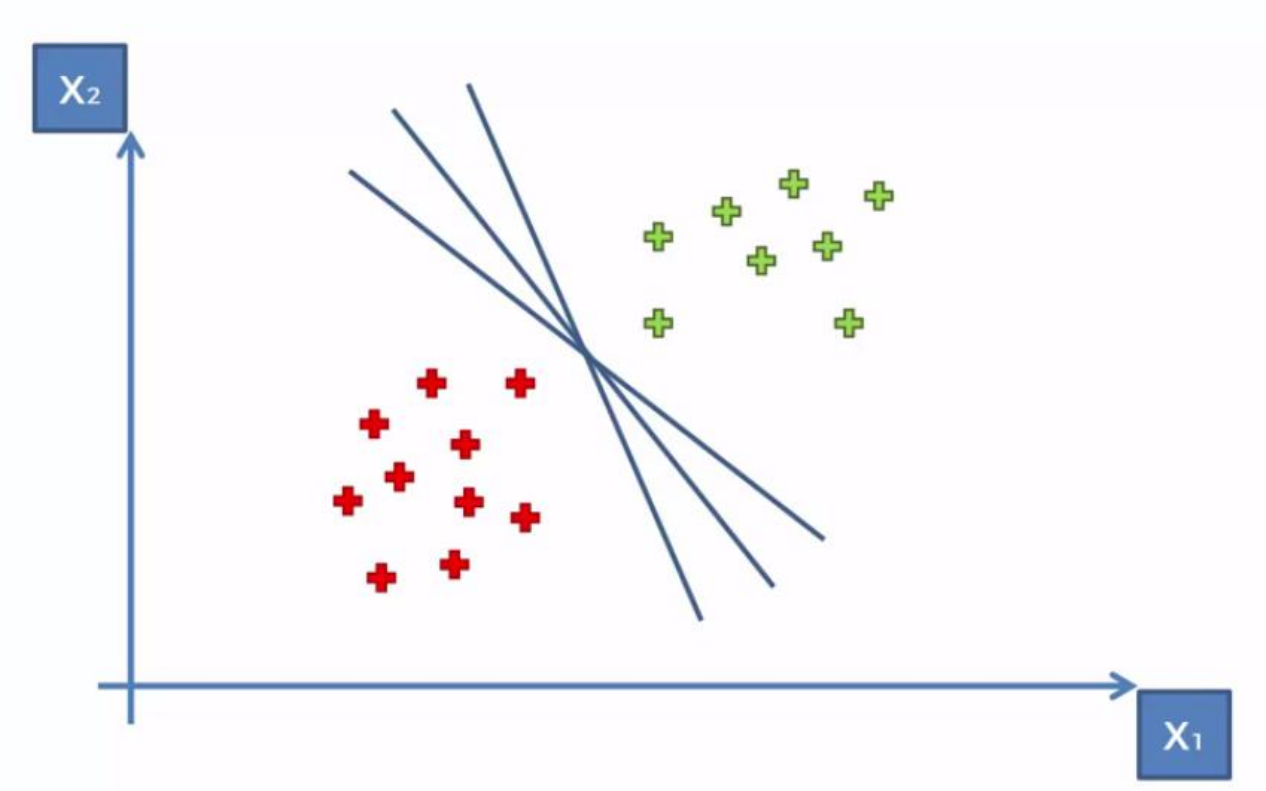
SVM were initially developed in 1960s and refined again in 1990s and now are getting popular in Machine Learning because they are very powerful and somewhat different to other machine learning algorithms.

It is a classifier as well as a regressor.

SVMs (linear or otherwise) inherently do binary classification. However, there are various procedures for extending them to multiclass problems.

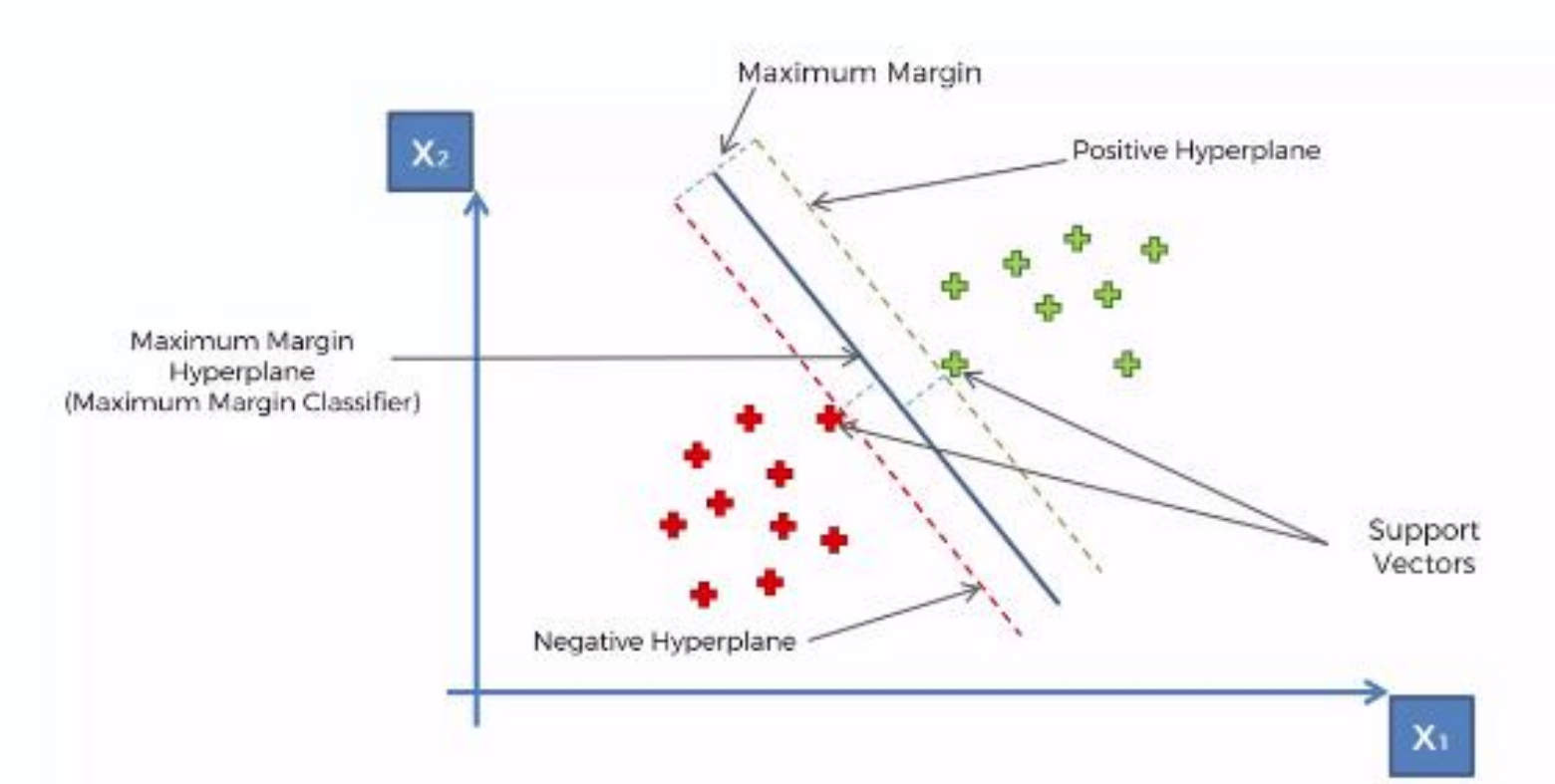
How it works ?

Lets say, you have these two different set of data.
There are many ways to separate these.



How it works ?

How SVM tries to separate these data points is that it tries to find the Maximum Margin.



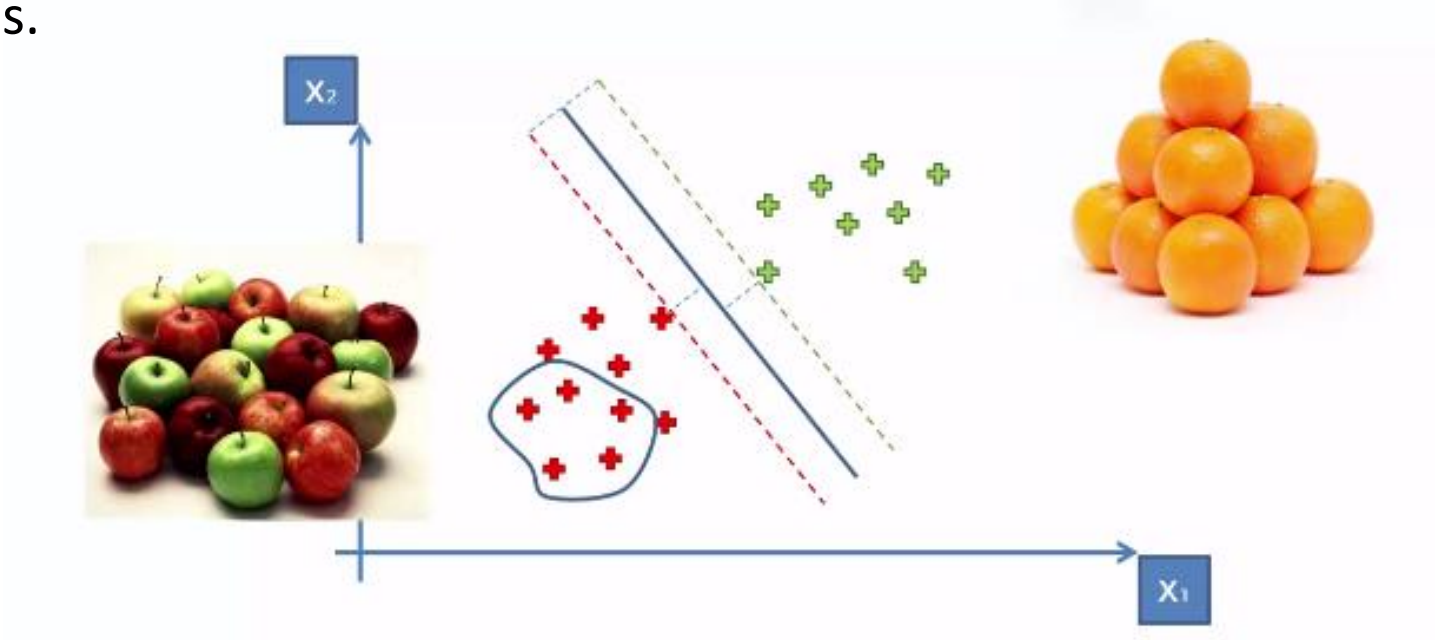
What is special about it ?

Imagine you are trying to teach a machine how to classify apples and oranges

So you pass the data to the machine and train it and then it will learn to recognize the apples and orange.
This is generally what happens in other classification algorithms.

What is special about it ?

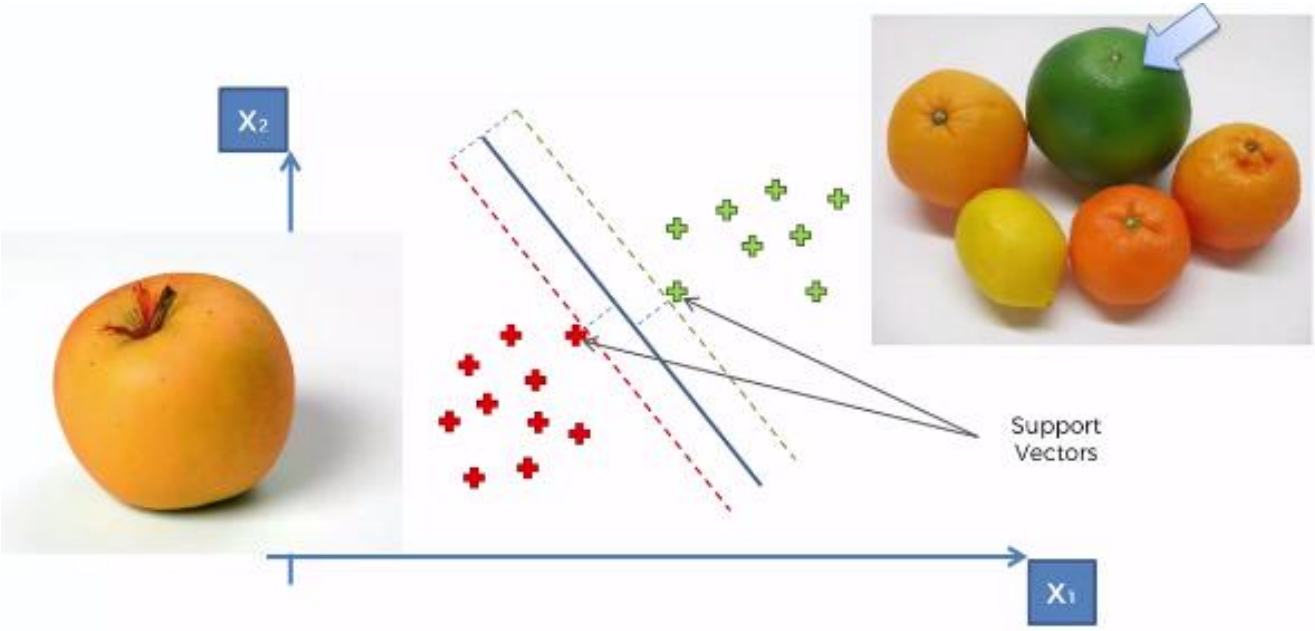
Other Machine Learning algorithms will look at the most standard common types of apples and oranges and then classify accordingly.
So basically they will have more inwards approach. The more closer the data point to the cluster, the more apple or orange it is.



What is special about it ?

However, in case of SVM, it is a bit different. It will look for apples that are not very standard looking apple and same for oranges.

Those different looking data points are what we call as **support vectors**.

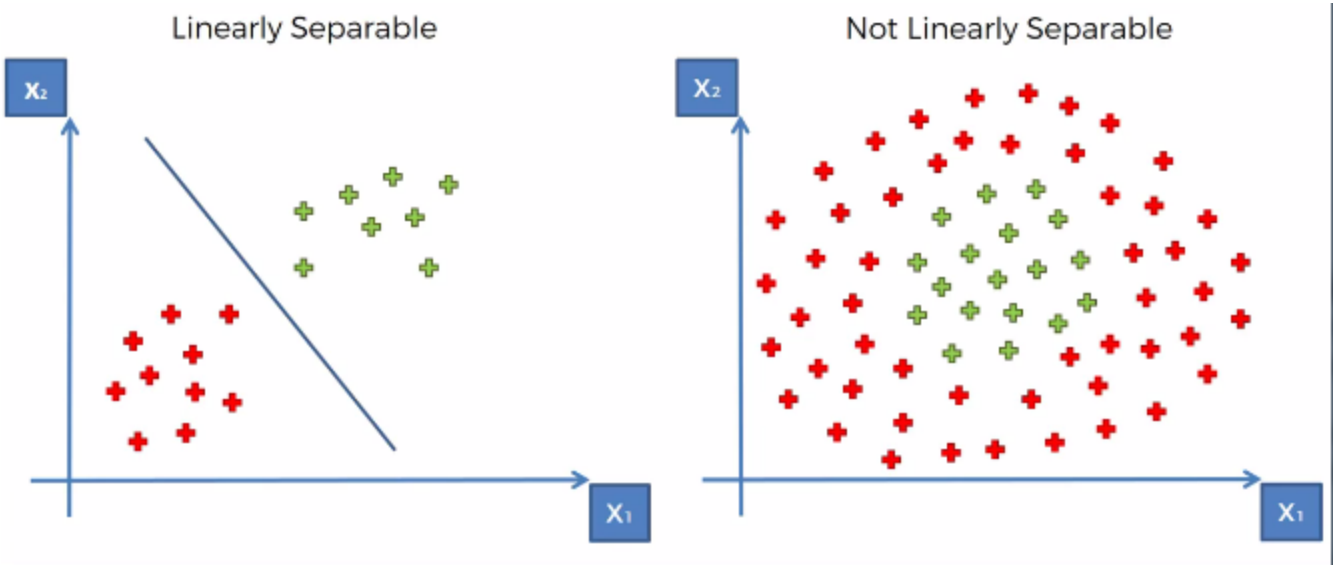


Kernel SVM

Kernel SVM

Generally, in linearly separated data, we find the boundaries using the SVM.

But what happens if we cannot find the boundaries. Like in cases when our data is not linearly separated.

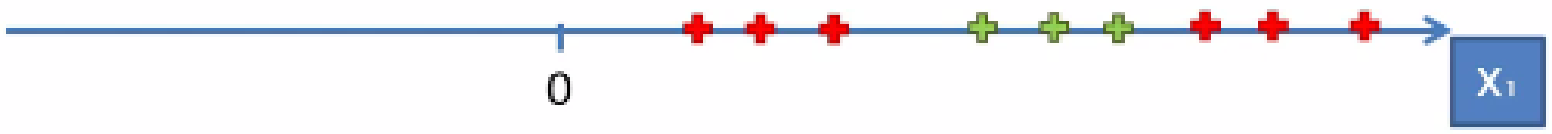


Kernel SVM – Mapping to Higher Dimension

In such cases of non linearly separation, we map our data to higher dimension.

e.g. lets start with some point in single dimension space.
Here Linear Separation is not possible as you can see.

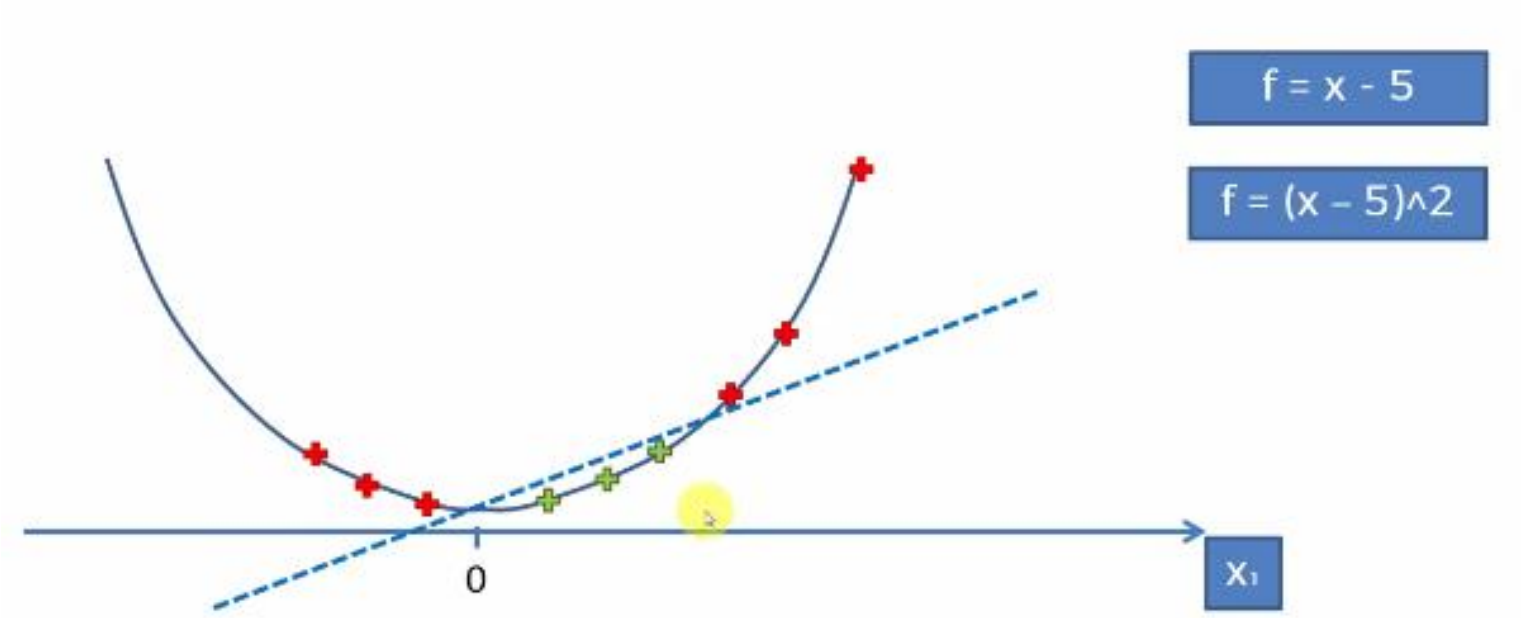
This is a non-linearly separable dataset.



Kernel SVM – Mapping to Higher Dimension

Here, we will increase the dimension using some mapping function. There can be many ways to do this.

Step 1: f = x-5
Step 2: $(x-5)^2$
Step 3: Separate the data points



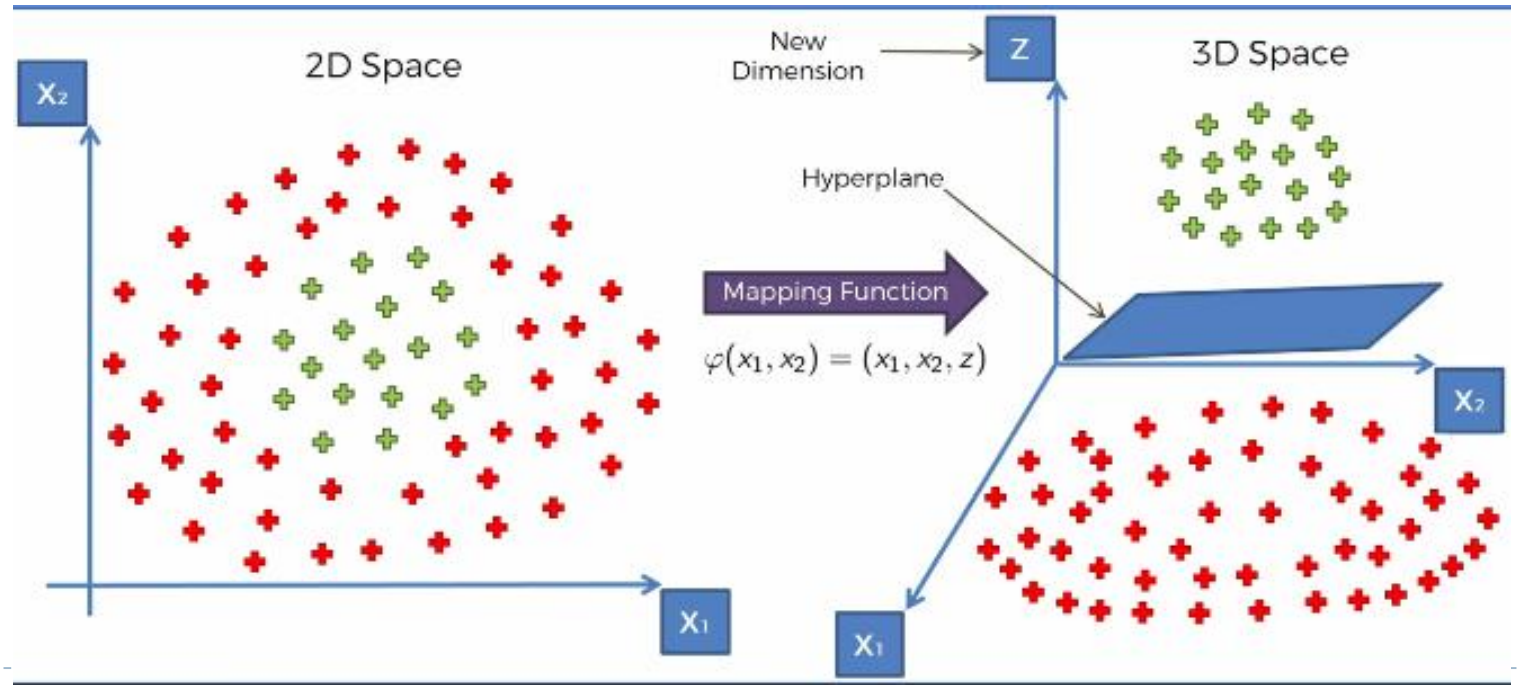
Kernel SVM – Mapping to Higher Dimension

Similarly, we can use this approach for 2-dimension space as well.

e.g. lets say we have this 2 dimensional space.

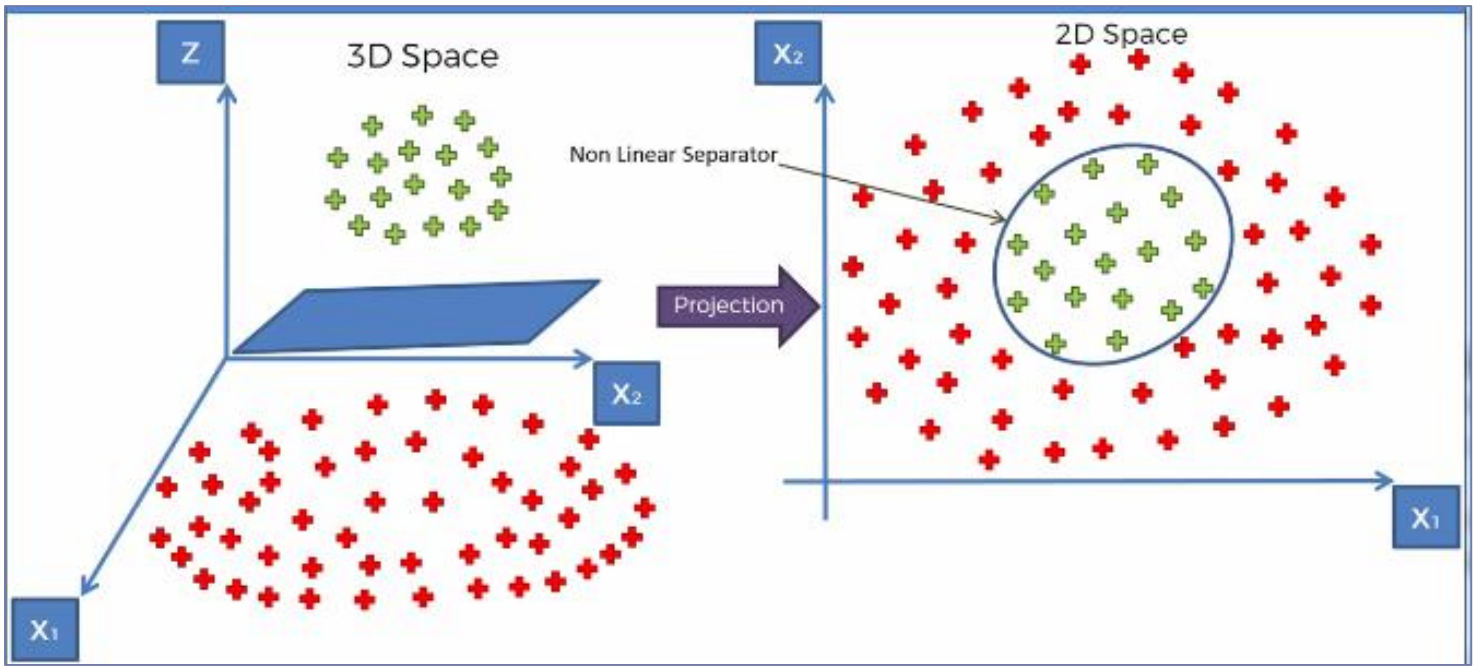
Step 1: Use some mapping function

Step 2: Use a hyperplane to separate the data points



Kernel SVM – Mapping to Higher Dimension

Step 3: Project back the data points to 2-dimension and get the decision boundary



Kernel SVM – Mapping to Higher Dimension

There is one problem with this approach though.

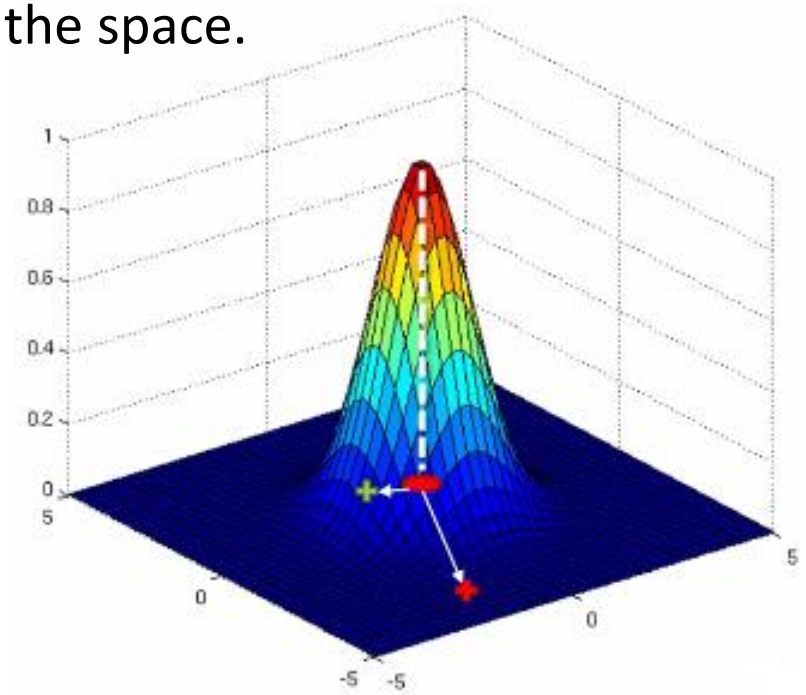
Mapping to higher dimensional space can be highly compute-intensive.

Therefore this mapping approach is not the best one; and hence we will use the **Kernel trick** to solve these problems.

Kernel SVM – Kernel Trick

So in case of kernel trick, we use the kernel formula (usually rbf) to project our points accordingly.

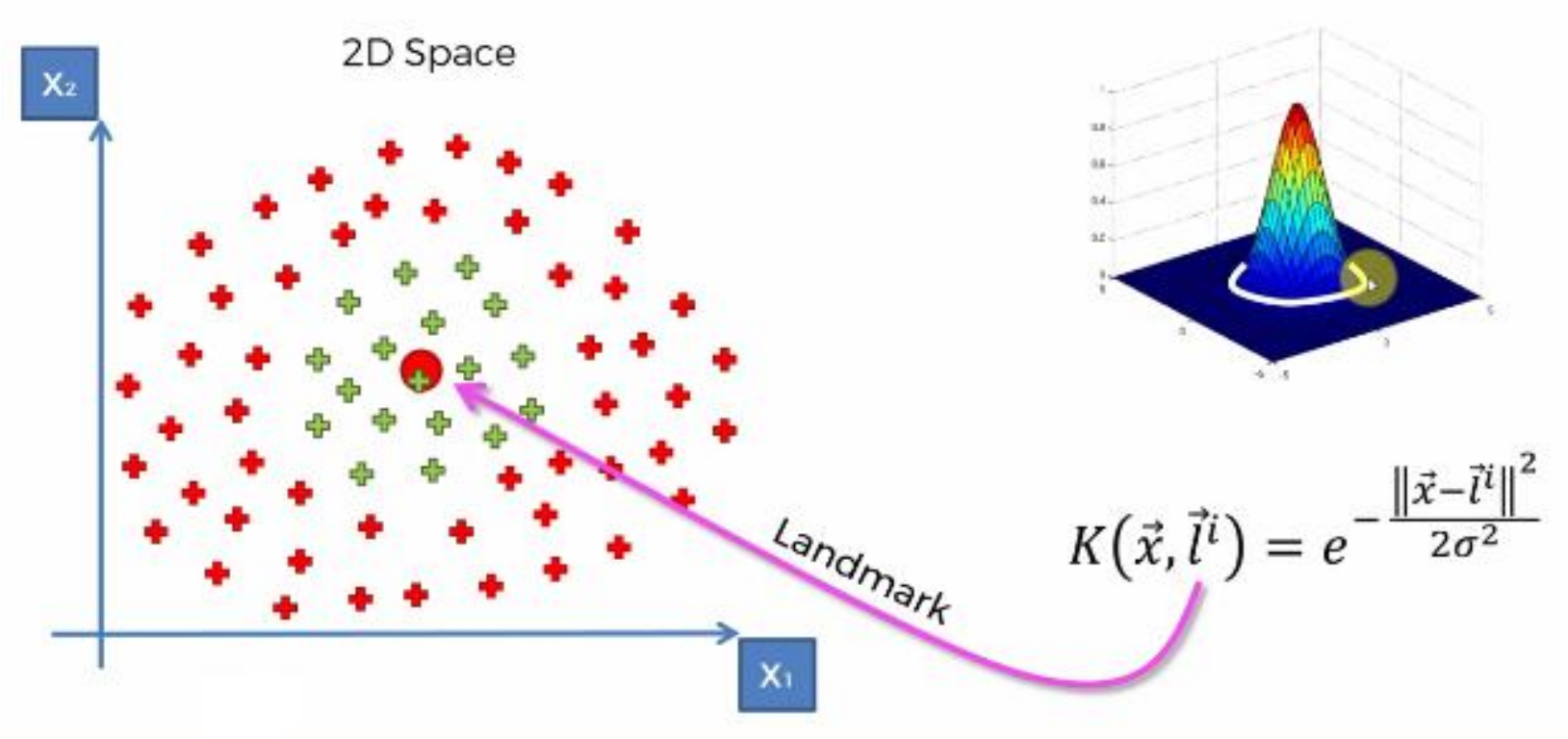
The father the point from landmark, the lower it will be in the space, the closer the point to the landmark, the higher it will be in the space.



$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^{2}}{2\sigma^{2}}}$$

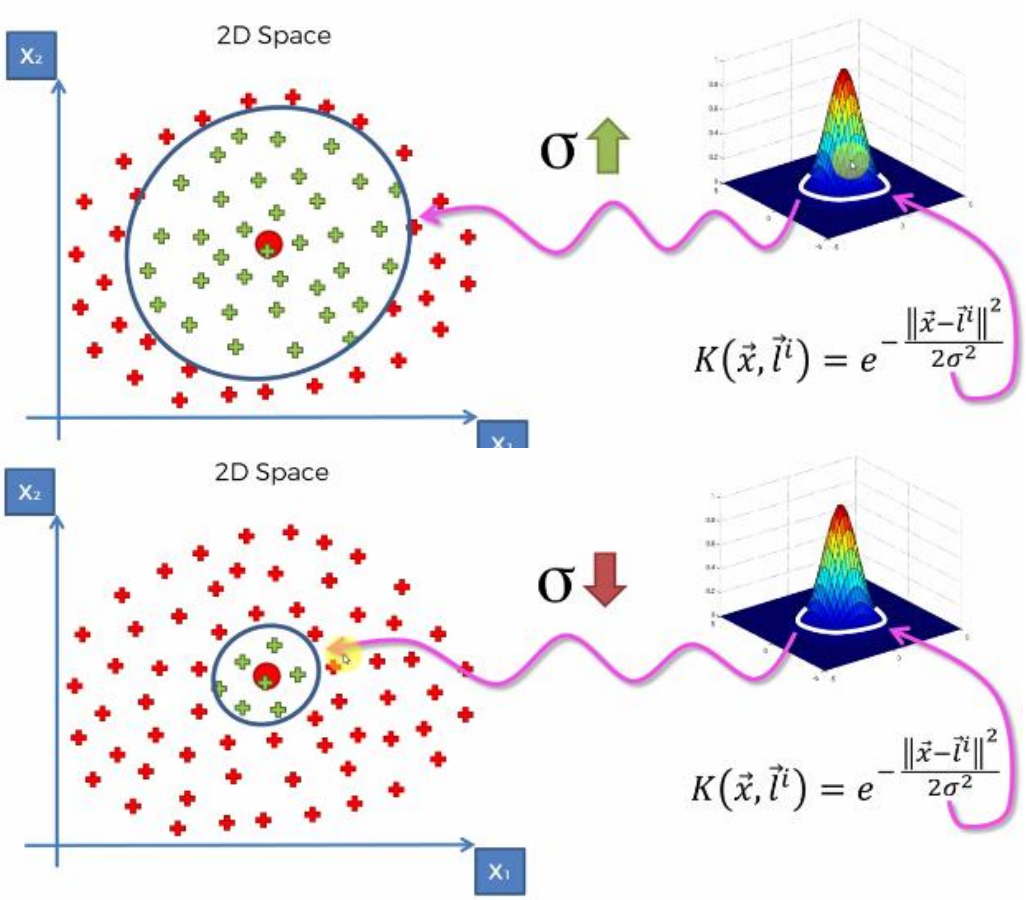
Kernel SVM – Kernel Trick

Sigma -> maintains the circumference.



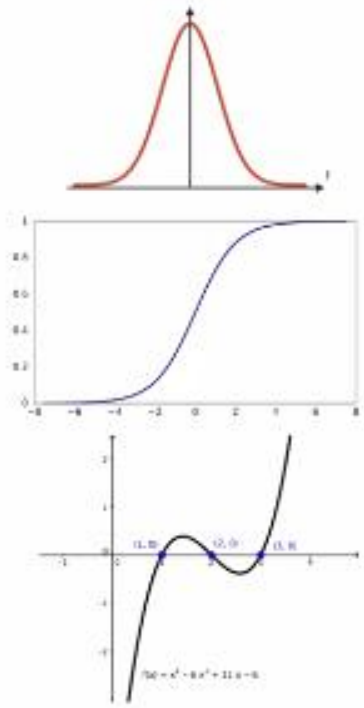
Kernel SVM – Kernel Trick

If sigma increases, the boundary increases and vice versa



Kernel SVM – Types of kernels

We have different types of Kernels in sklearn library.
Some of the most popular choice of kernels are:



Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$

Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

Rbf: Radial basis function