**Kernel Support Vector Machine**

**Intuition**

* In the Support Vector Machine algorithm, we had a set of observations which belonged to different classes and the point was to find this decision boundary between them so that it would be easy to identify which class the future points would belong to.
* In that case, there was a decision boundary, and the support vectors tells us how to find that boundary.
* But what happens when we cannot find a boundary? What happens in a situation shown below?

Chart, scatter chart

Description automatically generated

* We cannot simply separate the class with a simple horizontal, vertical, or diagonal line as the data is not linearly separable.
* What Support Vector Machine Algorithm does is that it helps us find that decision boundary, or correctly place that decision boundary. But it does have an assumption that the data is linearly separable.
* Whereas in non-linearly separable data, we cannot draw a single decision boundary.
* So, the Support Vector Machine Algorithm won’t work as per the definition.

**Higher-Dimensional Space:**

Chart, line chart

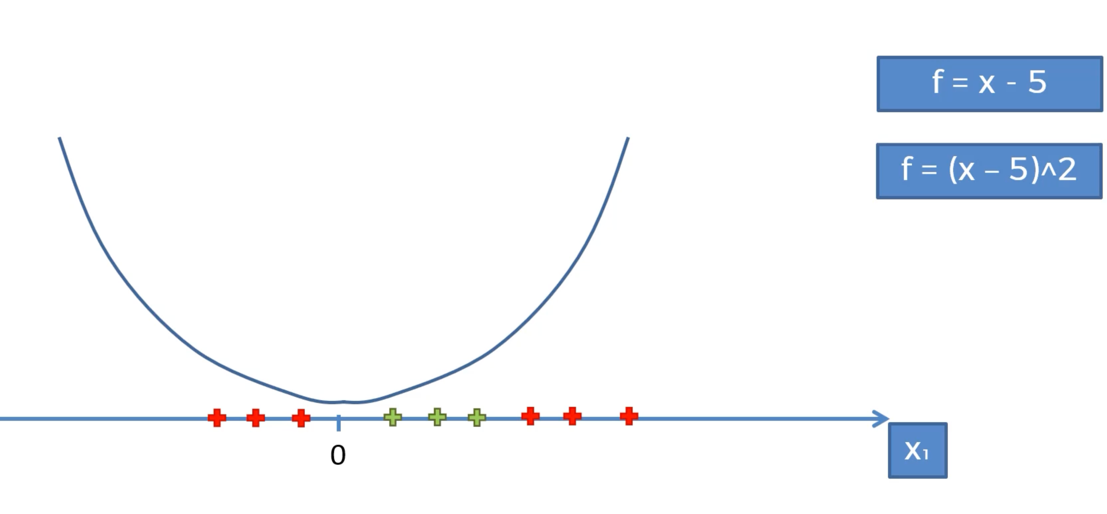
Description automatically generated

* Let’s say we have non-linearly separable points plotted on a graph.
* The first step that we will do is
* Where x = the point itself, and n = the point between the third red point and the first green point.
* In our case, the equation would be: (where we assume n to be 5).
* And the output of that would be something like this.

Chart, line chart

Description automatically generated

* The next step would be square the previous function.



* Therefore, the function would be:

Chart, line chart

Description automatically generated

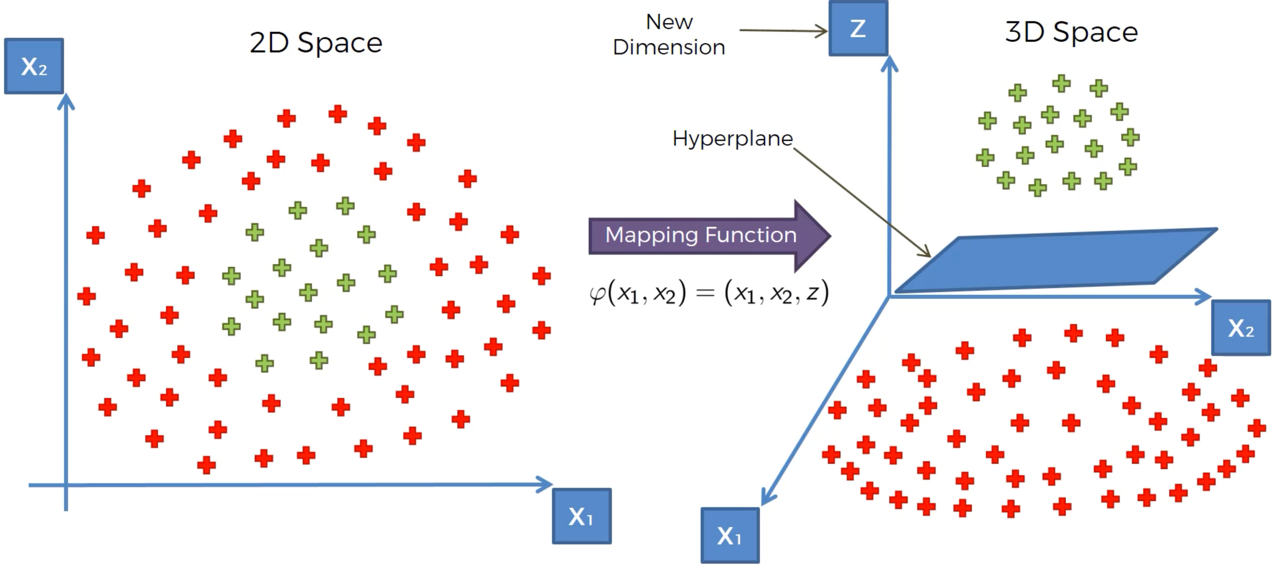
* In the next step, all the points on the x-axis would be projected on the curve.
* And finally, we pass a linear separator through the projected points.

Chart, line chart

Description automatically generated

* And as you can see, the data become linearly separable.
* Then what we would do next is project back the points on to the original space, and we would know how to functionally separate green from the red.
* Now we will try to implement the same principle on a 2-D space.
* In a 2-D space, we would apply the same principle that we applied on the 1-D space, and simply map the 2-D space into a 3-D space and somehow it would become a linearly separable dataset in that space.A picture containing timeline

  Description automatically generated
* And in a 3-D space, the linear separator is no longer a line, it is a hyperplane.



* And the hyperplane separates the dataset into two separate classes in the way we want.
* And then once we’ve got the result we want, we project it back into its original 2-D space.

Diagram

Description automatically generated

* And in the original projection, we have also got a non-linear separator.
* But there is a problem with this algorithm – mapping to a Higher Dimension Space can be highly compute-intensive. So, it might require a lot of computation and lot of processing power. The larger the dataset, the more computation resources it would require.
* Therefore, we are going to use a different approach which is called, in mathematics, the ‘kernel trick’. That approach is going to help us get similar results without having to go to higher dimensional space.

**The Kernel Trick:**

* Here we have the Gaussian RBF (Radial Basis Function) Kernel, and those are two interchangeable terms.

A picture containing text

Description automatically generated

* Where:
* K = kernel and it is the function applied to the x and l vectors
* x = a point on our dataset
* l = landmark
* i = number of landmarks
* e = exponent to the power of an equation
* ||x – l ||2 = distance between x and the landmark squared
* And the above equation is divided by two sigma squared.

**Types of Kernel Function:**

Graphical user interface, text

Description automatically generated

**Non-Linear Support Vector (Advanced) –**

Chart, scatter chart

Description automatically generated

* Let’s say we have a plot like this, and we want to fit a support vector model on it.
* We can try multiple linear models as shown below.

Chart, scatter chart

Description automatically generatedShape

Description automatically generatedA picture containing chart

Description automatically generated

* But as we can see, no matter what kind of a line we draw, it’s doesn’t look like a Linear Model fit here.
* It looks like something more complex is going on here.
* So, how do we build a support vector regression that will fit our data here?
* To solve this issue, we will be adding a new dimension to this.

Chart, scatter chart

Description automatically generated

* And for visualization purposes and to keep track of things, just because we are moving to a higher dimension, we will be adding that blue box, the diagonals, as well as a point in the middle. By adding that we are not changing the algorithm, or the dataset, or the axis. It is just for visual aid so that we can understand the algorithm better.

Chart, surface chart

Description automatically generated

* Basically, what we did was add a higher dimension to our plot, a 3rd dimension, and then ran an RBF on it which resulted into a model as shown on the right.
* In the next step, we added a support vector regression line around the points plotted on the 3rd dimension (The center green plan)
* And then, we added positive and negative support border lines.
* Finally, we projected our regression back on the original 2-D plan.