**Ace your Machine Learning Interview — Part 5**

**Dive into Kernel Support Vector Machines using Python**

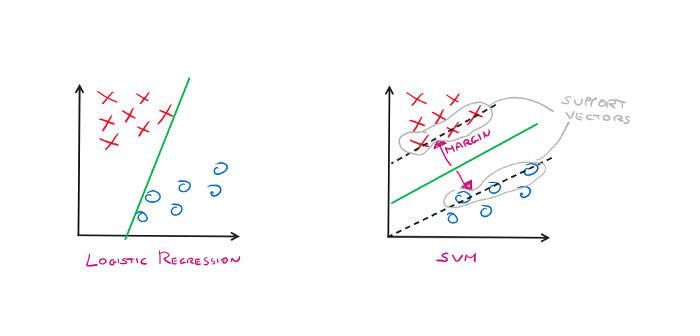
As I mentioned in the previous article in this series, I will now go on to talk about Support Vector Machines, and how they can be used to classify nonlinearly separable datasets using the **kernel trick**.

If you missed the previous articles in this *Ace your Machine Learning Interview series*, I leave the links below:

1. [*Ace your Machine Learning Interview — Part 1*](https://towardsdatascience.com/ace-your-machine-learning-interview-part-1-e6a5897e6844)*: Dive into Linear, Lasso and Ridge Regression and their assumptions*
2. [*Ace your Machine Learning Interview — Part 2*](https://towardsdatascience.com/ace-your-machine-learning-interview-part-2-c58526b5faba)*: Dive into Logistic Regression for classification problems using Python*
3. [*Ace your Machine Learning Interview — Part 3*](https://towardsdatascience.com/ace-your-machine-learning-interview-part-3-af432f922aa7)*: Dive into Naive Bayes Classifier using Python*
4. [*Ace your Machine Learning Interview — Part 4*](https://towardsdatascience.com/ace-your-machine-learning-interview-part-4-e30b695ce63)*: Dive into Support Vector Machines using Python*

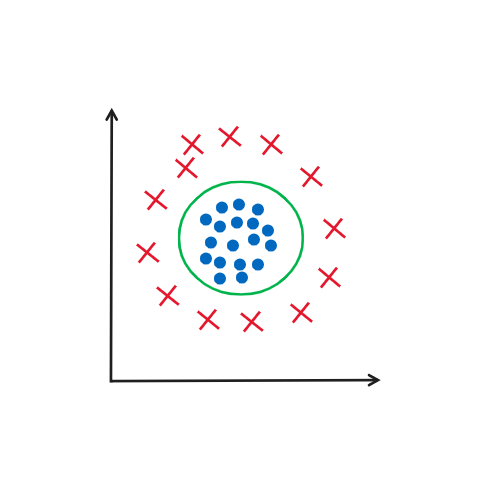
**Introduction**

We saw in the previous article how to use SVMs for classification problems by obtaining good generalizations by maximizing the margin.



Logistic Regression VS SVM (Image By Author)

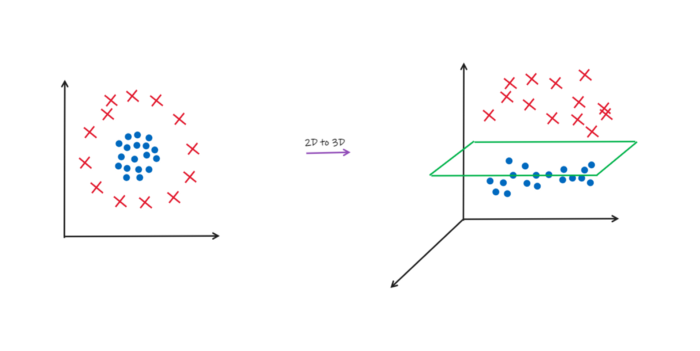
What do we do though if we have a dataset that cannot be classified by a line (or a hyperplane in n dimensions)?



Dataset not linearly separable (Image By Author)

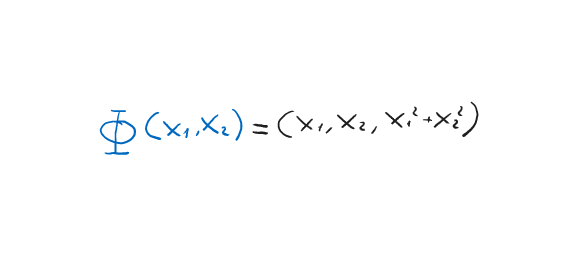
Notice that in the dataset shown in the figure above we have two classes, red and blue. But there is no way to separate these two classes using a straight line. The only way is by using a “circular” kind of function, the one depicted in green.

This is where the magic of SVM kernels happens. They allow us to **project our dataset into a higher-dimensional space**. **In this new space, it will then be easy to find a hyperplane that will divide the two classes correctly.**



Transformation from 2D to 3D (Image By Author)

So the basic idea of kernel methods for handling nonlinearly separable data is to create nonlinear combinations of the original features in order to project the dataset into a new space using a Φ function. Let us see an example of a **Φ function that brings us from a 2D space into a 3D space**.



Φ Mapping (Image By Author)

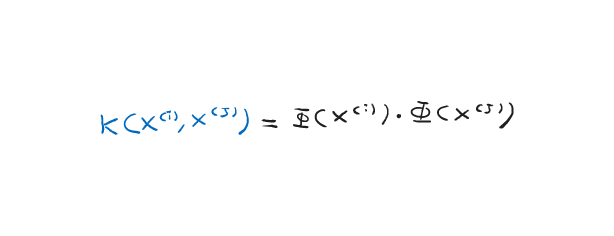
The problem with this approach is that building these new features is computationally very expensive. And this is where the so-called kernel trick comes to our aid. First, we would have to figure out how to solve the optimization problem for training SVMs seen in the previous chapter, but the math is very complex and would take up a lot of time.

To summarize we need only understand that we need to **calculate the dot product for each vector pair *x\_i \* x\_j***. Then in our case, **we should first project each vector into the new space using the function Φ and then calculate *Φ(x\_i) \* Φ(x\_j)***.

So there are two things to do:

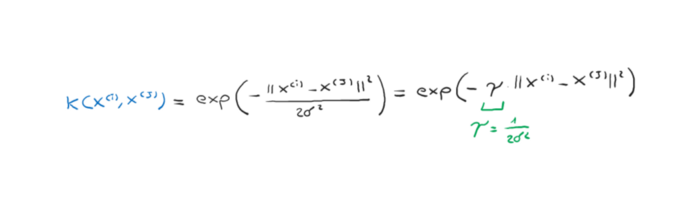
* project each vector into the new space
* calculate the dot product between pairs of vectors in the new space

The **kernel trick is a function that allows us to have directly the dot product result between the vectors in the new space without having to project each individual vector into the new space**. This will save us a lot of time and computation. More formally:



Kernel Trick (Image By Author)

But what does this kernel trick function look like? Don’t worry, you are not the one who has to find this function but they are well-known functions that you can find online. Let’s look at one of the most famous the **Gaussian Kernel or also called Radial Basis Function (RBF)**.



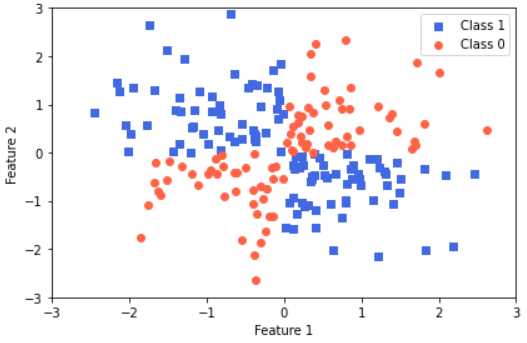
Gaussian Kernel (Image By Author)

Where *y* is the free parameter to be optimized.

The term kernel can be interpreted as a similarity function between a pair of examples. The minus sign inverts the distance measure into a similarity score, and, due to the exponential term, the resulting similarity score will fall into a range between 1 (for exactly similar examples) and 0 (for very dissimilar examples).

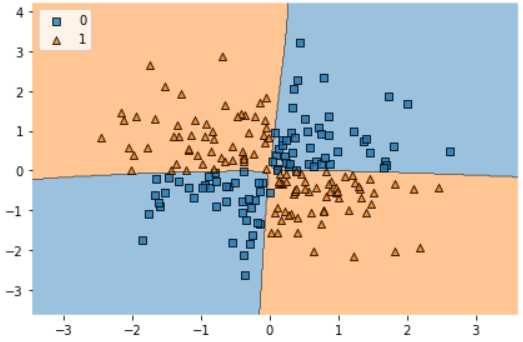
**Let’s code!**

First, let’s create our linearly inseparable dataset using the [xor](https://en.wikipedia.org/wiki/XOR_gate) function.



Linearly inseparable dataset (Image By Author)

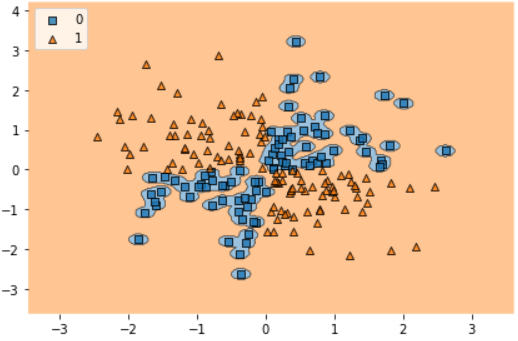
Now we can train an SVM using the parameter **kernel = ‘rbf’** where rbf stands for Radial Basis Function. And we see the decision boundaries that our classifier creates.



Decision Boundaries (Image By Author)

You can see that SVM was able to separate the two classes properly using the rbf kernel!

If we increase the value of gamma, we will increase the effect that the training data will have on decision boundaries. Therefore **excessively high gamma will create decision boundaries that will form as a kind of contour to our training data thus losing generalization ability**.



Overfitting (Image By Author)

**Final Thoughts**

In this article, we saw the theory behind Kernel SVMs and how to implement them using sklearn. We saw how the output is affected as the gamma parameter changes, which gives more weight to the training set data.

SVM kernels have long been the state of art of Machine Learning in many areas including Computer Vision. They are still widely used today so it is important to know about them! I hope you have found this article useful. If so, follow me to read the next articles of this *“Ace you Machine Learning Interview”* series!😁