What is the kernel trick? Why is it important?

Grace Zhang

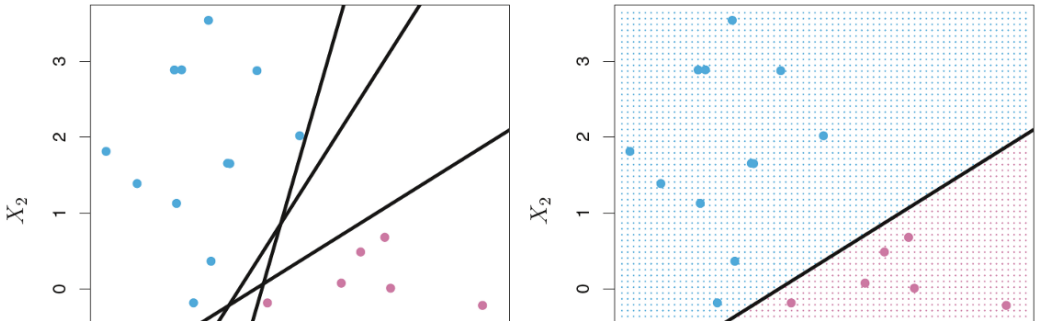
Nov 11, 2018 · 5 min read

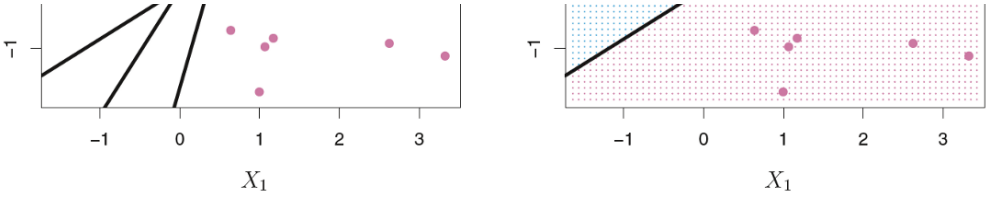
When talking about kernels in machine learning, most likely the first thing that comes into your mind is the support vector machines (SVM) model because the kernel trick is widely used in the SVM model to bridge linearity and non-linearity.

Support Vector Machines Basics

To help you understand what a kernel is and why it is important, I'm going to introduce the basics of the SVM model first.

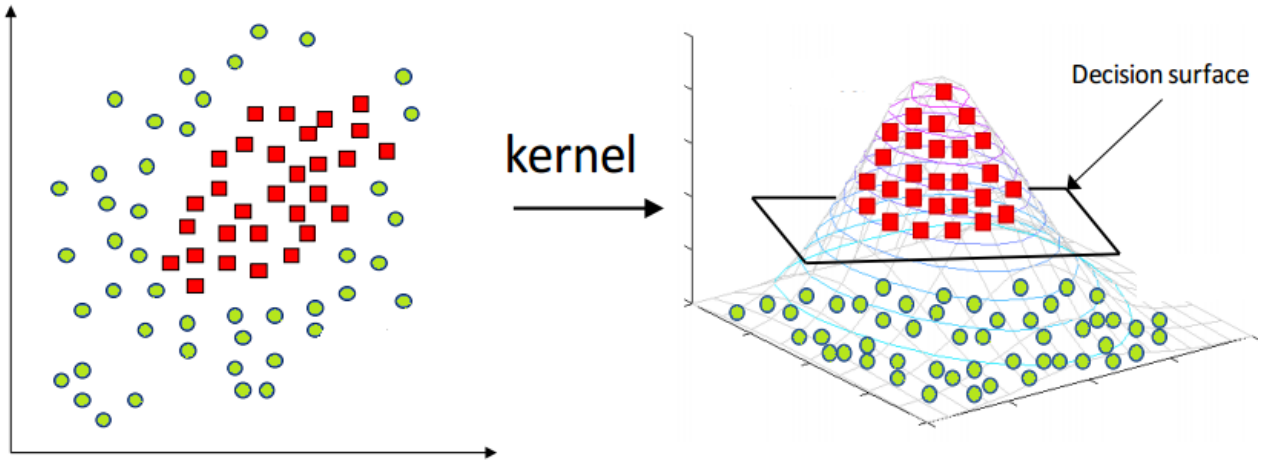
The SVM model is a supervised machine learning model that is mainly used for classifications (but it could also be used for regression!). It learns how to separate different groups by forming decision boundaries.





In the graph above, we notice that there are two classes of observations: the blue points and the purple points. There are tons of ways to separate these two classes as shown in the graph on the left. However, we want to find the “best” hyperplane that could maximize the margin between these two classes, which means that the distance between the hyperplane and the nearest data points on each side is the largest. Depending on which side of the hyperplane a new data point locates, we could assign a class to the new observation.

It sounds simple in the example above. However, not all data are linearly separable. In fact, in the real world, almost all the data are randomly distributed, which makes it hard to separate different classes linearly.



Why is it important to use the kernel trick?

As you can see in the above picture, if we find a way to map the data from 2-dimensional space to 3-dimensional space, we will be able to find a decision surface that clearly divides between different classes. My first thought of this data transformation process is to map all the data point to a higher dimension (in this case, 3 dimension), find the boundary, and make the classification.

That sounds alright. However, when there are more and more dimensions, computations within that space become more and more expensive. This is when the kernel trick comes in. It allows us to operate in the original feature space

without computing the coordinates of the data in a higher dimensional space.

Let's look at an example:

$$\mathbf{x} = (x_1, x_2, x_3)^T$$
$$\mathbf{y} = (y_1, y_2, y_3)^T$$

Here x and y are two data points in 3 dimensions. Let's assume that we need to map x and y to 9-dimensional space. We need to do the following calculations to get the final result, which is just a scalar. The computational complexity, in this case, is $O(n^2)$.

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$
$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

$$\phi(\mathbf{x})^T\phi(\mathbf{y}) = \sum_{i,j=1}^{3} x_i x_j y_i y_j$$

However, if we use the kernel function, which is denoted as k(x, y), instead of doing the complicated computations in the 9-dimensional space, we reach the same result within the 3-dimensional space by calculating the dot product of x - transpose and y. The computational complexity, in this case, is O(n).

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y})^2$$
$$= (x_1y_1 + x_2y_2 + x_3y_3)^2$$

$$= \sum_{i,j=1}^{3} x_i x_j y_i y_j$$

In essence, what the kernel trick does for us is to offer a more efficient and less expensive way to transform data into higher dimensions. With that saying, the application of the kernel trick is not limited to the SVM algorithm. Any computations involving the dot products (x, y) can utilize the kernel trick.

Different kernel functions

There are different kernels. The most popular ones are the polynomial kernel

and the radial basis function (RBF) kernel.

“Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these” (Wikipedia), just like the example above. With n original features and d degrees of polynomial, the polynomial kernel yields n^d expanded features.

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

The format of polynomial kernel

The RBF kernel is also called the Gaussian kernel. There is an infinite number of dimensions in the feature space because it can be expanded by the Taylor Series. In the format below, The γ parameter defines how much influence a single training example has. The larger it is, the closer other examples must be to be affected (sklearn documentation).

$$k(\mathbf{x}, \mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}, \gamma > 0$$

The format of the RBF kernel

There are different options for the kernel functions in the sklearn library in Python.

You can even build a custom kernel if needed.

The End

The kernel trick sounds like a “perfect” plan. However, one critical thing to keep in mind is that when we map data to a higher dimension, there are chances that we may overfit the model. Thus choosing the right kernel function (including the right parameters) and regularization are of great importance.

If you’re curious about what regularization is, I have written an article talking about my understanding of it and you can find it here.

I hope you enjoy this article. As always, please let me know if you have any questions, comments, suggestions, etc. Thanks for reading :)

Reference:

What are kernels in machine learning and SVM and why do we need them?

Kernel method — Wikipedia

Kernel Functions by Tejumade Afonja

Kernel functions for Machine Learning by PERPETUAL ENIGMA

MSDS621 Machine Learning at the University of San Francisco