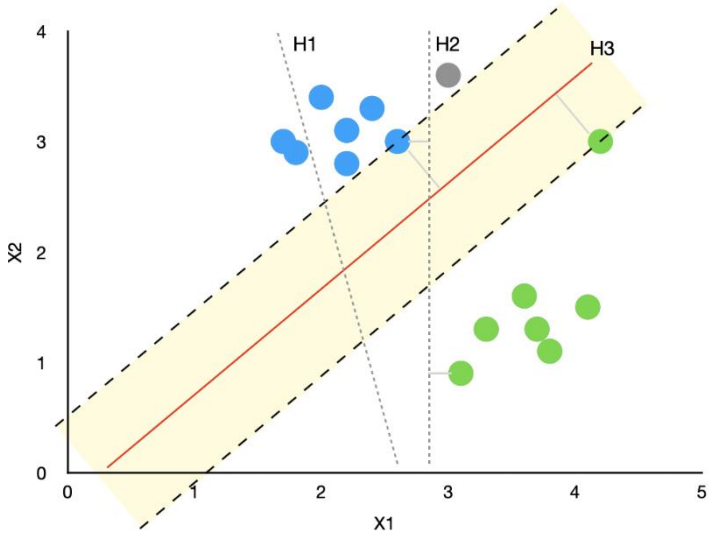
Kernel Functions-Introduction to SVM Kernel

SVM algorithm

Attempts to find a hyperplane that separates these two classes with the highest possible margin. If classes are fully linearly separable, a hard-margin can be used. Otherwise, it requires a soft-margin.

Note, the points that end up on the margins are known as support vectors.

- Hyperplane called "**H1**" cannot accurately separate the two classes; hence, it is not a viable solution to our problem.
- The "**H2**" hyperplane separates classes correctly. However, the margin between the hyperplane and the nearest blue and green points is tiny. Hence, there is a high chance of incorrectly classifying any future new points. E.g., the new grey point (x1=3, x2=3.6) would be assigned to the green class by the algorithm when it is obvious that it should belong to the blue class instead.

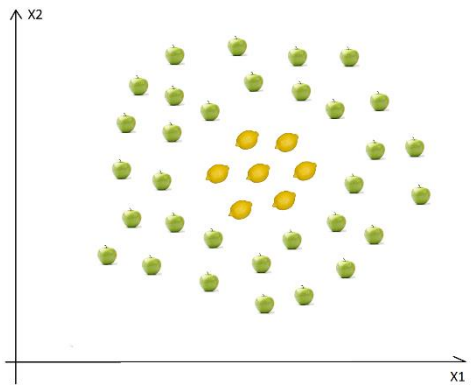


- Finally, the "**H3**" hyperplane separates the two classes correctly and with the highest possible margin (yellow shaded area). Solution found!

Kernel trick

The above explanation of SVM covered examples where blue and green classes are linearly separable. However, what if we wanted to apply SVMs to non-linear problems? How would we do that?

- This is where the kernel trick comes in. A **kernel is a function** that takes the original non-linear problem and transforms it into a linear one within the higher-dimensional space



SVM for Non-Linear Data Sets

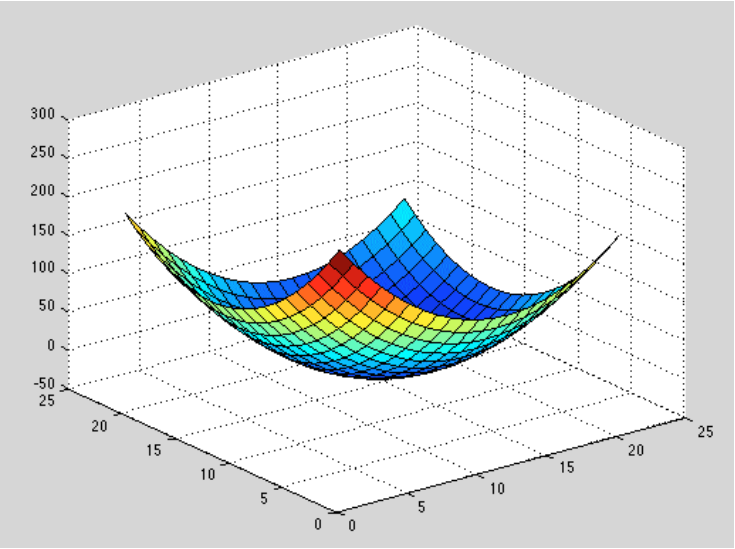
An example of non-linear data is:

In this case we cannot find a straight line to separate apples from lemons. So how can we solve this problem. We will use the Kernel Trick!

The basic idea is that when a data set is inseparable in the current dimensions, add another dimension, maybe that way the data will be separable. Just think about it, the example above is in 2D and it is inseparable, but maybe in 3D there is a gap between the apples and the lemons, maybe there is a level difference, so lemons are on level one and lemons are on level two. In this case we can easily draw a separating hyperplane (in 3D a hyperplane is a plane) between level 1 and 2.

Mapping to Higher Dimensions

To solve this problem we shouldn't just blindly add another dimension, we should transform the space so we generate this level difference intentionally.

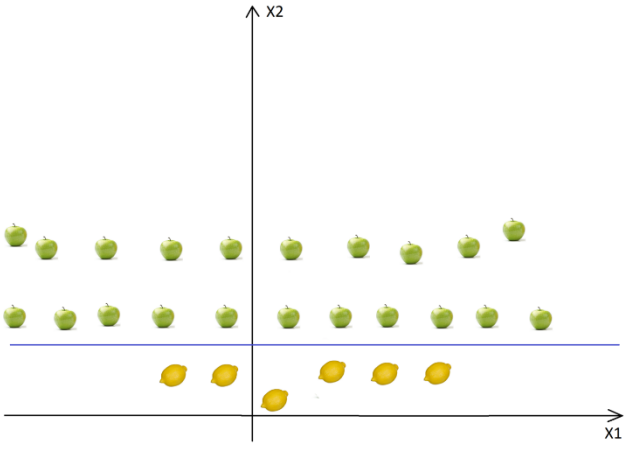


Mapping from 2D to 3D

Let's assume that we add another dimension called X3. Another important transformation is that in the new dimension the points are organized using this formula $x1^2 + x2^2$.

If we plot the plane defined by the $x^2 + y^2$ formula, we will get something like this:

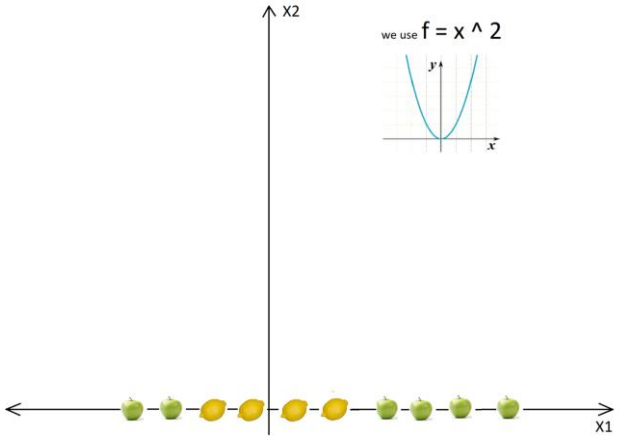
Now we have to map the apples and lemons (which are just simple points) to this new space. Think about it carefully, what did we do? We just used a transformation in which we added levels based on distance. If you are in the origin, then the points will be on the lowest level. As we move away from the origin, it means that we are climbing the hill (moving from the center of the plane towards the margins) so the level of the points will be higher. Now if we consider that the origin is the lemon from the center, we will have something like this:



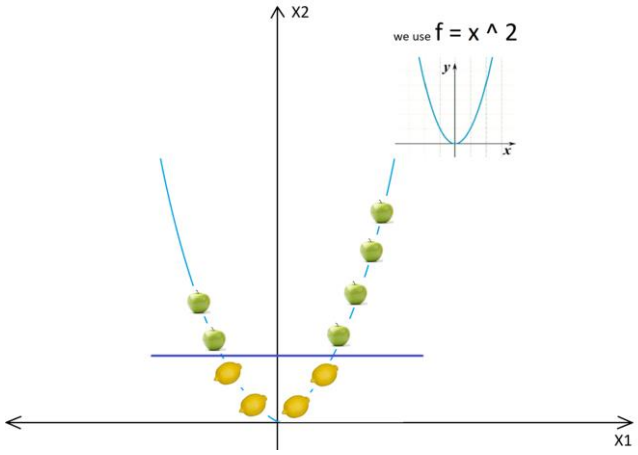
Now we can easily separate the two classes. These transformations are called kernels. Popular kernels are: Polynomial Kernel, Gaussian Kernel, Radial Basis Function (RBF), Laplace RBF Kernel, Sigmoid Kernel, Anove RBF Kernel, etc (see Kernel Functions or a more detailed description Machine Learning Kernels).

Mapping from 1D to 2D

Another, easier example in 2D would be:



After using the kernel and after all the transformations we will get:



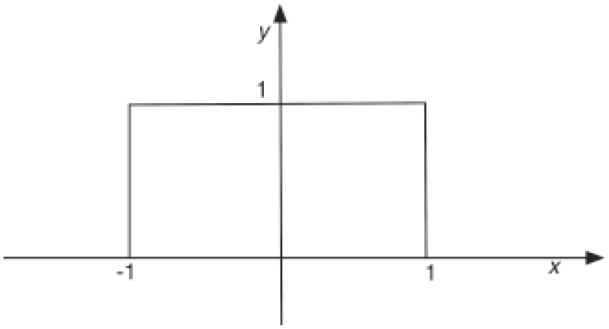
So after the transformation, we can easily delimit the two classes using just a single line.

Kernel Rules

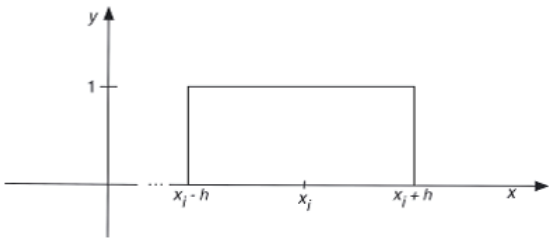
Define kernel or a window function as follows:

$$K\left(\overline{x}\right) = \begin{matrix} 1 & \text{if} \left\|\overline{x}\right\| \leq 1 \\ 0 & \text{otherwise} \end{matrix}$$

This value of this function is 1 inside the closed ball of radius 1 centered at the origin, and 0 otherwise . As shown in the figure below:



For a fixed xi, the function is K(z-xi)/h) = 1 inside the closed ball of radius h centered at xi, and 0 otherwise as shown in the figure below:



So, by choosing the argument of K(·), you have moved the window to be centered at the point xi and to be of radius h.

Examples of SVM Kernels

Let us see some common kernels used with SVMs and their uses:

1. Polynomial kernel

It is popular in image processing.

Equation is:

$$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$$

where d is the degree of the polynomial.

2. Gaussian kernel

It is a general-purpose kernel; used when there is no prior knowledge about the data. Equation is:

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$

3. Gaussian radial basis function (RBF) very important

It is a general-purpose kernel; used when there is no prior knowledge about the data.

Equation is:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

, for:

$$\gamma > 0$$

Sometimes parametrized using:

$$\gamma = 1/2\sigma^2$$

RBF kernels are the most generalized form of kernelization and is one of the most widely used kernels due to its similarity to the Gaussian distribution. The RBF kernel function for two points X_1 and X_2 computes the similarity or how close they are to each other. This kernel can be mathematically represented as follows:

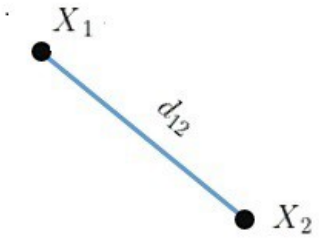
$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2\sigma^2})$$

where,

1. ‘σ’ is the variance and our hyperparameter

2. $||X_1 - X_2||$ is the Euclidean (L_2-norm) Distance between two points X_1 and X_2

Let d_{12} be the distance between the two points X_1 and X_2, we can now represent d_{12} as follows:



$$d_{12} = ||X_1 - X_2||^2$$

The kernel equation can be re-written as follows:

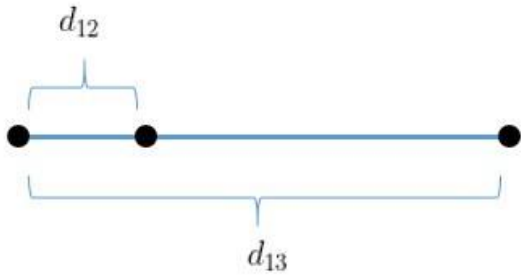
$$K(X_1, X_2) = exp(-\frac{d_{12}}{2\sigma^2})$$

The maximum value that the RBF kernel can be is 1 and occurs when d_{12} is 0 which is when the points are the same, i.e. $X_1 = X_2$.

When the points are the same, there is no distance between them and therefore they are extremely similar

When the points are separated by a large distance, then the kernel value is less than 1 and close to 0 which would mean that the points are dissimilar

Distance can be thought of as an equivalent to dissimilarity because we can notice that when distance between the points increases, they are less similar.



$$d_{12} < d_{13}$$

$$K(X_1, X_2) > K(X_1, X_3)$$

Fig 3: Similarity decreases as distance increases [Image by Author]

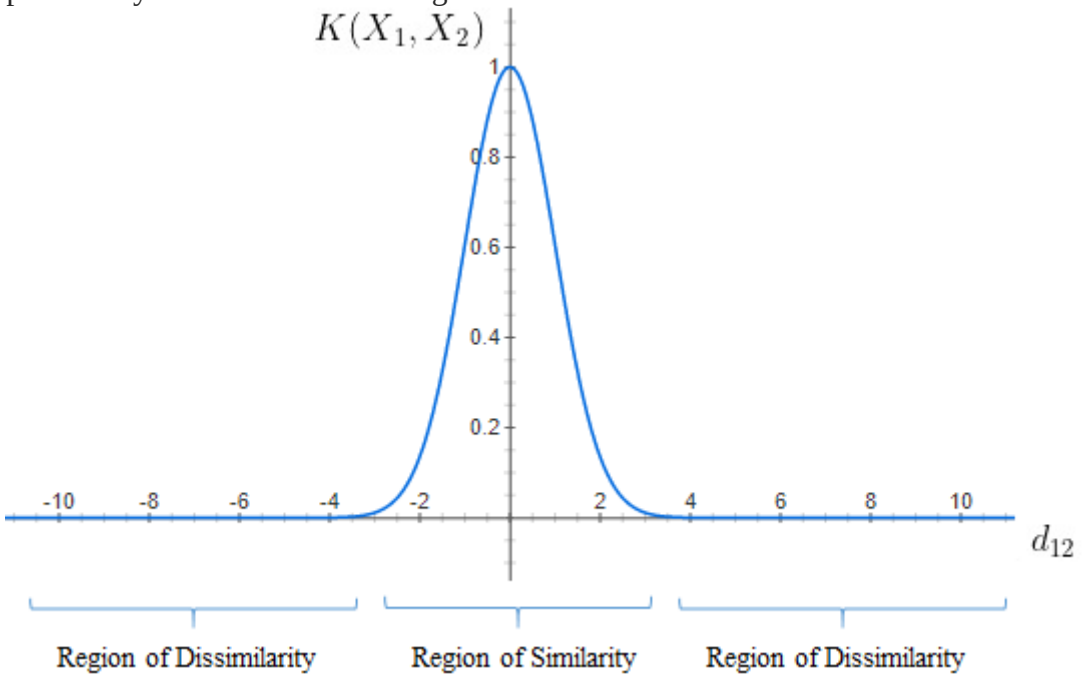
It is important to find the right value of ‘σ’ to decide which points should be considered similar and this can be demonstrated on a case by case basis.

a] σ = 1

When σ = 1, $\sigma^2 = 1$ and the RBF kernel’s mathematical equation will be as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2})$$

The curve for this equation is given below and we can notice that as the distance increases, the RBF Kernel decreases exponentially and is 0 for distances greater than 4.



We can notice that when $d_{12} = 0$, the similarity is 1 and as d_{12} increases beyond 4 units, the similarity is 0

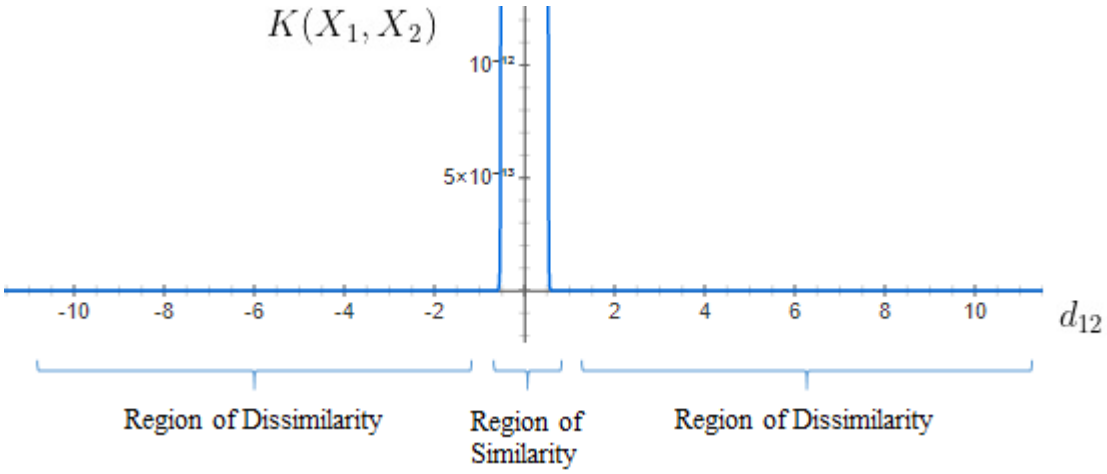
From the graph, we see that if the distance is below 4, the points can be considered similar and if the distance is greater than 4 then the points are dissimilar

b] $\sigma = 0.1$

When $\sigma = 0.1$, $\sigma^2 = 0.01$ and the RBF kernel's mathematical equation will be as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{0.01})$$

The width of the Region of Similarity is minimal for $\sigma = 0.1$ and hence, only if points are extremely close they are considered similar.



We see that the curve is extremely peaked and is 0 for distances greater than 0.2

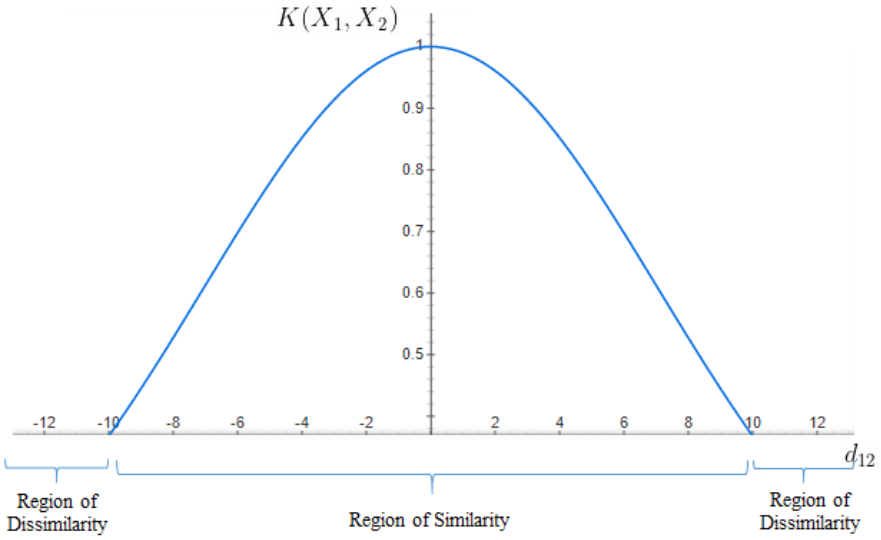
The points are considered similar only if the distance is less than or equal to 0.2

b] $\sigma = 10$

When σ = 10, σ² = 100 and the RBF kernel's mathematical equation will be as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{100})$$

The width of the Region of Similarity is large for σ = 100 because of which the points that are farther away can be considered to be similar.



The width of the curve is large

The points are considered similar for distances up to 10 units and beyond 10 units they are dissimilar

It is evident from the above cases that the width of the Region of Similarity changes as σ changes.

Finding the right σ for a given dataset is important and can be done by using hyperparameter tuning techniques like Grid Search Cross Validation and Random Search Cross Validation.

RBF Kernel is popular because of its similarity to K-Nearest Neighborhood Algorithm. It has the advantages of K-NN and overcomes the space complexity problem as RBF Kernel Support Vector Machines just needs to store the support vectors during training and not the entire dataset.

4. Laplace RBF kernel

It is general-purpose kernel; used when there is no prior knowledge about the data.

Equation is:

$$k(x, y) = \exp\left(-\frac{||x - y||}{\sigma}\right)$$

5. Hyperbolic tangent kernel

We can use it in neural networks.

Equation is:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c)$$

, for some (not every) k>0 and c<0.

6. Sigmoid kernel

We can use it as the proxy for neural networks. Equation is

$$k(x, y) = \tanh(\alpha x^T y + c)$$

Sigmoid kernel equation

7. Bessel function of the first kind Kernel

We can use it to remove the cross term in mathematical functions. Equation is :

$$k(x, y) = \frac{J_{v+1}(\sigma \|x - y\|)}{\|x - y\|^{-n(v+1)}}$$

where j is the Bessel function of first kind.

8. ANOVA radial basis kernel

We can use it in regression problems. Equation is:

$$k(x, y) = \sum_{k=1}^{n} \exp(-\sigma(x^k - y^k)^2)^d$$

9. Linear splines kernel in one-dimension

It is useful when dealing with large sparse data vectors. It is often used in text categorization. The splines kernel also performs well in regression problems. Equation is:

$$k(x, y) = 1 + xy + xy\ min(x, y) - \frac{x + y}{2}\ min(x, y)^2 + \frac{1}{3}\min(x, y)^3$$

References:

https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200
https://data-flair.training/blogs/svm-support-vector-machine-tutorial/
https://www.python-engineer.com/courses/mlfromscratch/07_svm/
https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
https://towardsdatascience.com/support-vector-machines-svm-clearly-explained-a-python-tutorial-for-classification-problems-29c539f3ad8