Kernel Functions-Introduction to SVM Kernel

By: Ahmed Nabil Ibrahim Awaad Mansoura Group1

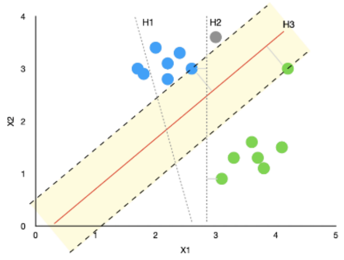
Contents:

1. History of SVM

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues (Boser et al., 1992, Guyon et al., 1993, Vapnik et al., 1997) SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximize the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently **perform a non-linear classification using what is called the kernel trick**, implicitly mapping their inputs into high-dimensional feature spaces.

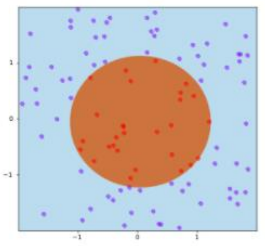
2. Motivation

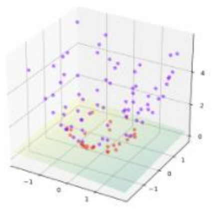


Attempts to find a hyperplane that separates these two classes with the highest possible margin. If classes are fully linearly separable, a hard margin can be used. Otherwise, it requires a soft margin.
Note, the points that end up on the margins are known as support vectors.

- 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!

3. 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

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. **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.
Now we have to map the 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.

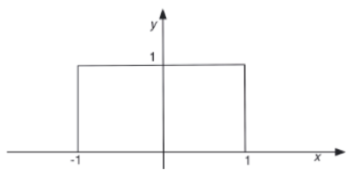
Now we can easily separate the two classes. These transformations are called kernels. Popular kernels are:
Polynomial Kernel, Gaussian Kernel, Radial Basis Function (RBF), Laplace RBF Kernel, Sigmoid Kernel, Anove RBF Kernel, etc

4. Kernel Rules

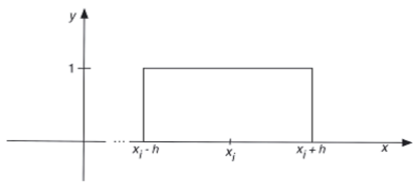
Define kernel or a window function as follows:

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

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.

5. Examples of SVM Kernels and how to choose correct Kernel.

Let us see some common kernels used with SVMs and their uses:
5.1 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.

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

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

5.3. 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:

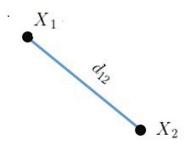
$$\gamma > 0$$

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:

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

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:

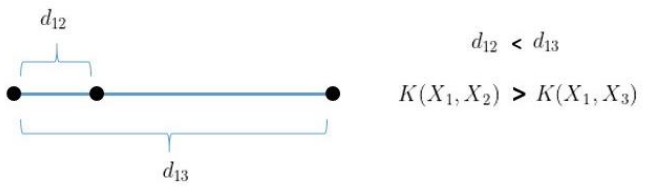


$$d_{12} = ||X_1 - X_2||^2$$

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, $X_1 = X_2$.
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.

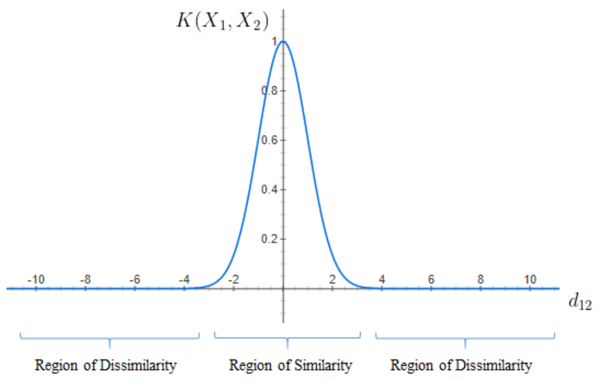


It is important to find the right value of 'σ' to decide which points should be considered similar and this can be demonstrated on a case-by-case basis.

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.

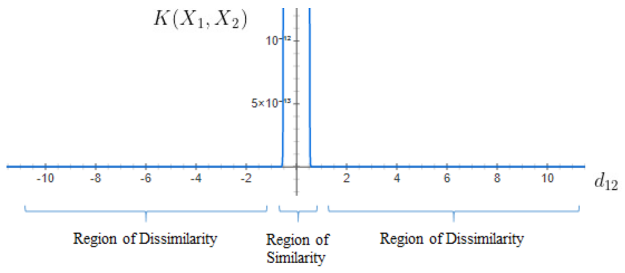
We can notice that when $d_{12} = 0$, the similarity is 1 and as d_{12} increases beyond 4 units, the similarity is 0

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] σ = 0.1

When σ = 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 σ = 0.1 and hence, only if points are extremely close, they are considered similar.

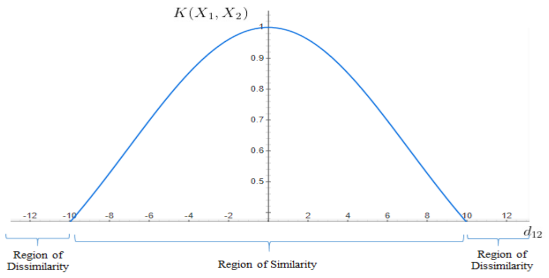
We see that the curve is extremely peaked and is 0 for distances greater than 0.2

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

c] σ = 10

When σ = 10, $\sigma^2 = 100$ 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.

5.4. Laplace RBF kernel

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

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

5.5. Hyperbolic tangent kernel

We can use it in neural networks. for some (not every) k>0 and c<0.

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

5.6. Sigmoid kernel

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

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

5.7. 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.

5.8. ANOVA radial basis kernel

We can use it in regression problems. Equation is:

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

5.9. 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:

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

6. SVC for multi-class classification

Multiclass SVM aims to assign labels to instances by using support-vector machines, where the labels are drawn from a finite set of several elements.

The dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems. Common methods for such reduction include:

- Building binary classifiers that distinguish between one of the labels and the rest (one-versus-all) or between every pair of classes (one-versus-one). Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest-output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determine the instance classification.
- Directed acyclic graph SVM (DAGSVM)
- Error-correcting output codes.
- Crammer and Singer proposed a multiclass SVM method which casts the multiclass classification problem into a single optimization problem, rather than decomposing it into multiple binary classification problems

7. The Concept of VC dimension

In Vapnik–Chervonenkis theory, the Vapnik–Chervonenkis (VC) dimension is a measure of the capacity (complexity, expressive power, richness, or flexibility) of a set of functions that can be learned by a statistical binary classification algorithm. It is defined as the cardinality of the largest set of points that the algorithm can shatter, which means the algorithm can always learn a perfect classifier for any labeling of that many data points. It was originally defined by Vladimir Vapnik and Alexey Chervonenkis.

VC dimension of a classification model
A binary classification model f with some parameter vector theta is said to shatter a set of data (x{1},x{2},... ,x{n}) if, for all assignments of labels to those points, there exists a theta such that the model f makes no errors when evaluating that set of data points. The VC dimension of a model f is the maximum number of points that can be arranged so that f shatters them. More formally, it is the maximum cardinal D such that some data point set of cardinality D can be shattered by f.

8. The Curse of Dimensionality

The Curse of Dimensionality is termed by mathematician R. Bellman in his book “Dynamic Programming” in 1957. According to him, the curse of dimensionality is the problem caused by the exponential increase in volume associated with adding extra dimensions to Euclidean space.
The curse of dimensionality basically means that the error increases with the increase in the number of features. It refers to the fact that algorithms are harder to design in high dimensions and often have a running time exponential in the dimensions. A higher number of dimensions theoretically allow more information to be stored, but practically it rarely helps due to the higher possibility of noise and redundancy in the real-world data.
In machine learning, a small increase in the dimensionality would require a large increase in the volume of the data in order to maintain a similar level of performance. Thus, the curse of dimensionality is the expression of all phenomena that appear with high-dimensional data, and that have most often unfortunate consequences on the behavior and performances of learning algorithms.

How To Combat The CoD:

- Dimensionality Reduction: is a method of converting the high dimensional variables into lower dimensional variables without changing the specific information of the variables.
- Regularization: The problem of curse of dimensionality comes from that parameter estimates which are unstable, hence, regularizing these estimates will help that the parameters to make correct estimation.
- PCA: (Principal Component Analysis) is one of the most traditional tools used for dimension reduction. It transforms the data into the most informative space, thereby allowing the use of lesser dimensions which are almost as informative as the original data.

References:

1. https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200
2. https://data-flair.training/blogs/svm-support-vector-machine-tutorial/
3. https://www.python-engineer.com/courses/mlfromscratch/07_svm/
4. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
5. https://towardsdatascience.com/support-vector-machines-svm-clearly-explained-a-python-tutorial-for-classification-problems-29c539f3ad8
6. https://analyticsindiamag.com/curse-of-dimensionality-and-what-beginners-should-do-to-overcome-it/
7. https://en.wikipedia.org/wiki/Vapnik%E2%80%93Chervonenkis_dimension
8. https://en.wikipedia.org/wiki/Support-vector_machine