All About ML

All About ML — Part 7: Support Vector Machines

Dharani J Follow
May 16, 2020 · 9 min read

Given a problem statement to categorize a set of data into classes, we can resort to algorithms like logistic regression, decision trees, boosting techniques etc., There is one more interesting and intuitive concept that helps in classification which is support vector machines. To understand SVM we must have clear idea on hyperplane, margin, kernel. Here is my attempt to help you understand these terms :)

Hyperplane:

Assume you have a 2D space with some data points as shown, and a line (ax+ by +c=0) is able to group this space into two parts.

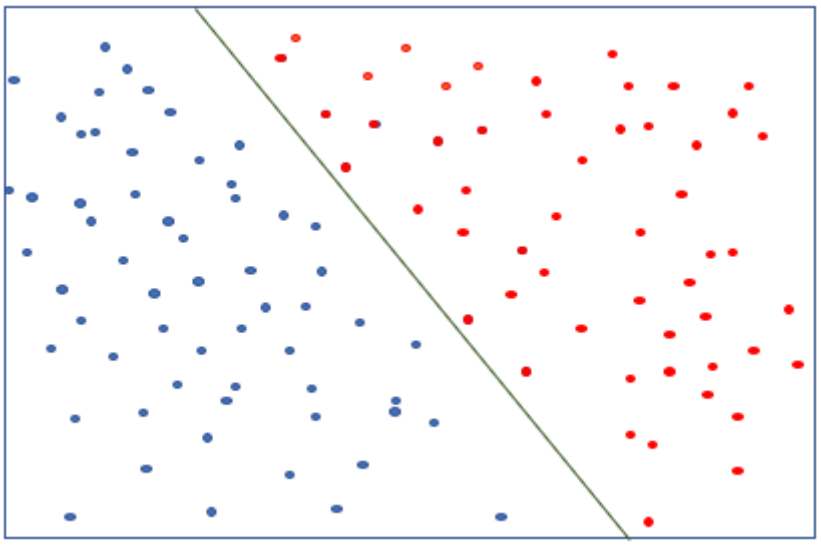


fig-1

Similarly for data in 3D space, a 2D plane can group the data into parts and for higher dimensions this works as well but it is hard to visualize. This flat

line(1D)/plane(2D)/sub space of dimension p-1 which categorizes the given data space of p dimensions(features) is called a **hyperplane**.

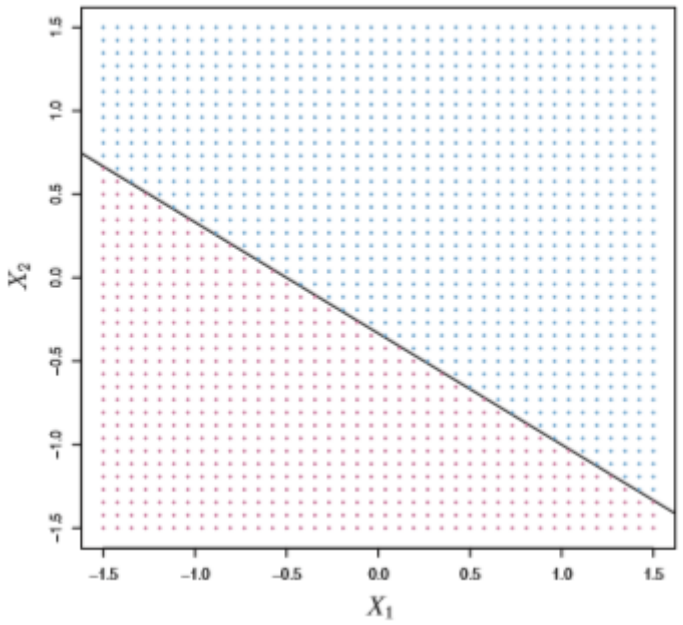
Suppose we are solving a classification problem and the data is spread as shown in fig-1. If the algorithm is able to find the hyperplane then we are sorted because it classified all the points correctly. Let us understand this approach in depth.

For a p-dimension hyperplane, if a point X = (X1,X2,…,Xp) in p-dimensional space (i.e. a vector of length p) satisfies eq-1, then X lies on the hyperplane.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

eq-1

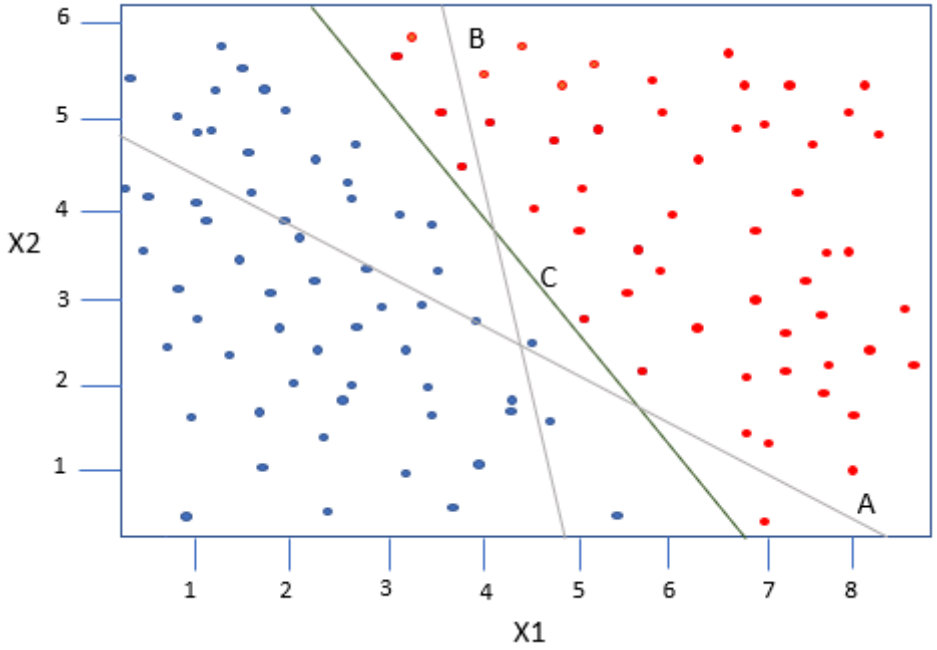
If X does not satisfy the equation and instead if β0 + β1X1 + β2X2 + …+ βpXp > 0 then it lies on one side of the hyper plane and if β0 + β1X1 + β2X2 + …+ βpXp <0 then X lies on the other side of the hyper plane. With this equation we can divide the whole data space into 2 parts and can easily classify the new data point based on the above conditions. For example, a hyperplane of 1+2X1+3X2=0 will look like:



Any point that satisfies 1+2×1+3×2<0 lies in red and 1+2×1+3×2>0 lie in blue region.

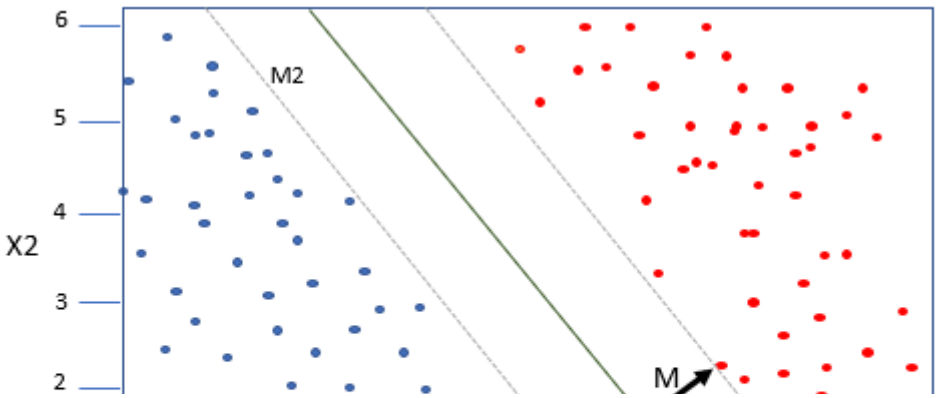
There is a data set which looks as shown below. We can bring up many hyper planes, let us understand the A,B,C hyper planes as shown below. C is able to

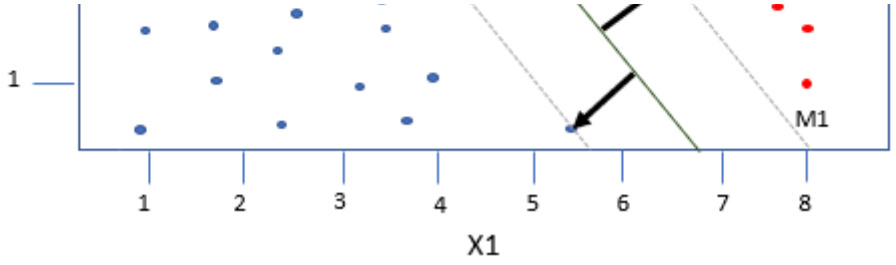
correctly classify and B had few errors and A has many errors in classifying the data. But how does the algorithm pick if A,B,C as best hyperplane to classify data?



Margin:

Let us take the same data space with only one hyperplane, the lines to the closest data points from the hyperplane are represented as M1 and M2 (these should be perpendicular to the hyperplane selected) and the distance from the hyperplane to either M1 or M2 is called **marign** and these nearest points where arrows are put are called **support vectors.** Support vectors lie along the margins and indicates the length of margin thus supporting the hyperplane because if these points are moved either near or away from hyperplane, then margin is decreased or increased respectively. If we observe closely, the hyperplane and margin is dependent only on support vectors and not on any other data points.





For different hyper planes we have different lengths of margins, and the hyperplane which has the maximum margin is called the **Maximal margin hyperplane** or the **optimal separating hyperplane.** If we are categorizing a data point into a class based on the position it lies with respect to the maximal margin hyperplane (left or right of hyperplane) then it is called **maximal margin classifier.**

For this kind of data set as shown in the figure above, the hyperplane is able to correctly categorize the points but its not the same always, many real time data sets might have hyper planes having misclassified data points as well. This an important point to make while learning SVM. This helps in adding robustness to the model when using on test data.

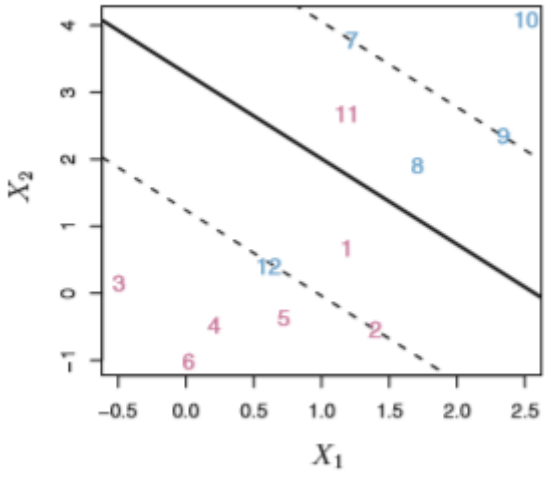


fig-2

For example in fig 2, point 1,8 are to the wrong side of their respective margins but to the correct side of hyperplane. Points 11 and 12 are located on the wrong side of hyperplane and margin as well.

What happens with different lengths of M? If margin is large i.e., if the support vectors are far from hyperplane, then it is obvious that the data points are well distanced enough and a new point coming in can be easily classified with very

less error. If M is small, then the support vectors are really close to hyperplane and any new point coming in has a very high probability of misclassification. So optimization would be to maximize M(width of margin). Mathematically,

$$\begin{aligned}
&\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M \\
&\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \\
&y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \\
&\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C,
\end{aligned}$$

eq-2

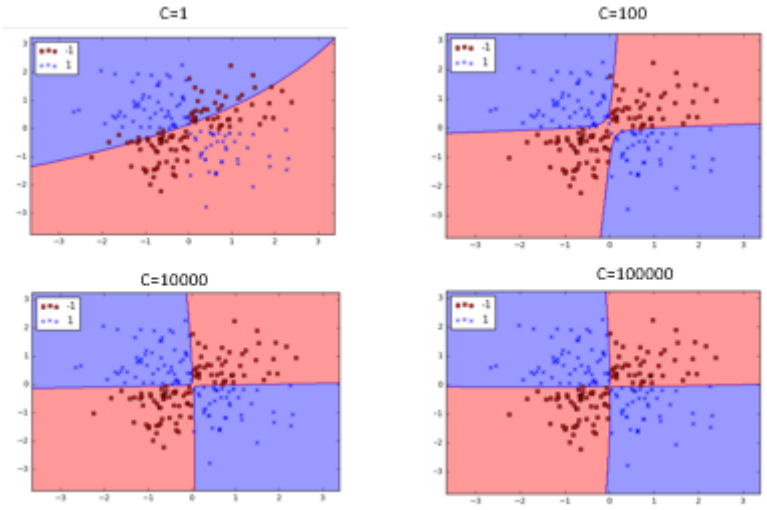
here, yi is the classification variable where it takes values of 1 and -1 only for the two classes present, C is a non negative tuning parameter. If β0,β1,…,βp are the coefficients of the maximal margin hyperplane, then a new test data point x* is classified based on the **sign** of f(x*) = β0 + β1x1* + β2x2* + …+ βpxp*. epsilon-1,…,n are slack variables that allow individual observations to be on the wrong side of the margin or the hyperplane. If epsilon-i = 0 then the ith observation is on the correct side of the margin. If epsilon-i > 0 then the ith observation is on the wrong side of the margin, and we say that the ith observation has violated the margin. If epsilon-i > 1 then it is on the wrong side of the hyperplane as explained in fig-2.

The role of the tuning parameter C — It bounds the sum of the epsilon-i's, so it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate. C is the assignment of amount that margin can be violated by n observations.

— If C = 0 then there is no budget for violations to the margin, and it must be the case that epsilon-1 = …= epsilon-n = 0, in which simply amounts to the maximum margin hyperplane (in the equation above).

— C is picked using cross-validation. If C is small, the margins are closer and the model has high fit on train data with low bias but high variance. If C is larger then margins have more width and a new data point has a high

probability of violating the hyperplane separation, thus leading to more bias but lower variance. Tuning all these parameters will result in a good classifier.



For various values of C source: Chris Albon

For a random data with blue and red points as different classes, as the value of C increases, the model is clearly intolerant of misclassified data and tries to classify all of them correctly by adjusting the hyperplane.

Kernels:

A question that might already have arised looking at fig-1 is that what if the data points are spread randomly in the space? what if there is no linear boundary separating the classes- then how does SVM work?—The trick is with kernels.

To deal with this problem of non- linear boundary between classes, we can transform the feature space to a large size using polynomial function of predictors by making them squared or cubed or to any higher degree. But doing so would lead to computational inability because of huge predictor space. Here is where kernels have an upper hand in ruling out computational roadblocks. So let us try to understand it.

Looking at eq-2 above, it is comprised of a basic mathematical computation — inner product of the observations. Inner product of any two observations xi,xi` is

$$\langle x_i, x_{i'} \rangle = \sum^{p} x_{ij} x_{i'j}$$

$$\underline{\qquad}_{j=1}$$

For a linear boundary, this is one representation, there will be different representations for non-linear boundaries which we will see soon. So when ever we refer to a function (like inner product here) we can write a generalized form K (xi,xi`). K is any function and it is referred as Kernel here which measures the similarity of two observations.

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

An example for K can be this inner product and called a linear kernel

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

Polynomial Kernel

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

Radial Kernel

So, the support vector classifier with any kernel K is represented as

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

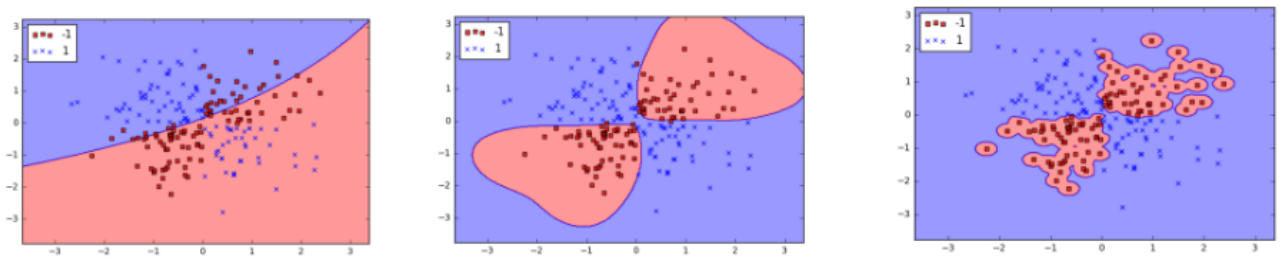
eq-3

where there are n parameters αi,i=1,…,n, one per training observation. However, it turns out that αi is nonzero only for the support vectors in the

solution — that is, if a training observation is not a support vector, then its αi equals zero. S is the collection of indices of these support points.

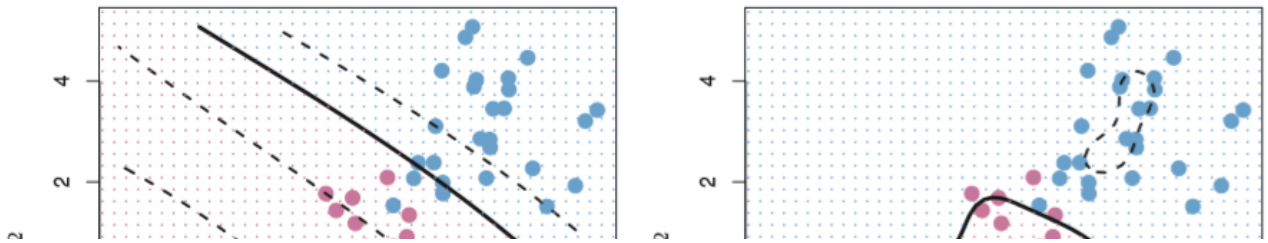
The **polynomial** kernel can be defined as above with degree **d**. Instead of using linear kernel, polynomial kernel provides a much better decision boundary for classifying the data. So, a kernel essentially amounts to fitting a support vector classifier in a higher-dimensional space involving polynomials of degree d, rather than in the original feature space.

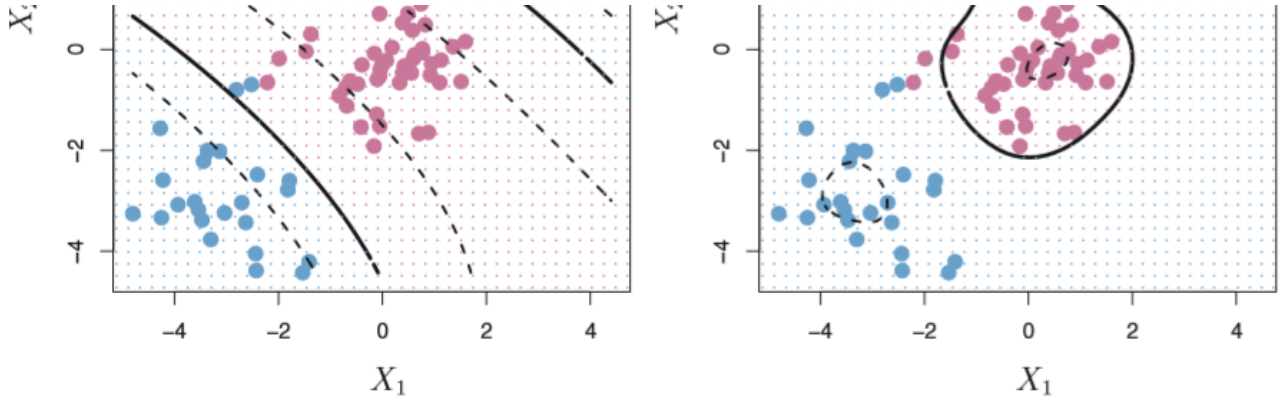
Another popular kernel used is **radial** kernel which has gamma as a positive constant that has a great impact on the model. If a given test observation x* = (x1*, …xp*) is far from a training observation xi in terms of Euclidean distance is large but the equation in radial kernel discounts it using gamma and the distance is greatly reduced, so xi will play virtually no role in f(x*)— eq-3 above. Recall that the predicted class label for the test observation x* is based on the sign of f(x*). In other words, training observations that are far from x* will play essentially no role in the predicted class label for x*. This means that the **radial** kernel has very **local** behavior, in the sense that only nearby training observations have an effect on the class label of a test observation.



Increase in gamma is gradually resulting in overfitting with complex boundaries source: Chris Albon

> *Deploying kernels with with support vectors for non linear boundaries is called* ***Support Vector Machine.***





Example of using Kernels: Left — Polynomial kernel of degree 3; Right — Radial Kernel

Learning in python

Like all other models, SVM is also included in sklearn package. We have various parameters in sklearn.svm where SVC function is for classification and linearSVC for regression. As we talked about classification here, lets check out the important parameters in SVC function that might greatly effect the results.

C: This is a regularization parameter that we have in eq-2 as explained above. default C value is 1

kernel: Values that go into this parameter are **'linear', 'poly', 'rbf', 'sigmoid', 'precomputed',** default**='rbf'** (rbf- radial basis function)

degree: If a polynomial kernel is in use, then we have to specify the degree here. For rest of the kernels this will not be used. default is 3

gamma: this is another important parameter while using radial, polynomial or sigmoid kernels.

Let us understand how this is implemented in python with a tutorial. I have taken breast cancer anlaysis data set from kaggle for classification. For data cleaning and EDA—please check this blog if you are interested.

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,f1_score

c_range = [0.01,0.1,1,10,100]

eval_metric = pd.DataFrame(columns = ['C_parameter','Accuracy'])
eval_metric['C_parameter'] = c_range

j = 0
```

```
for i in c_range:
 svm_linear = SVC(kernel = 'linear', C = i, random_state = 0)
 svm_linear.fit(x_train,y_train)
 y_pred = svm_linear.predict(x_test)
 eval_metric.iloc[j,1] = metrics.accuracy_score(y_test,y_pred)
 j += 1

print(eval_metric)
```


Output of above snippet.

I have taken a linear kernel with different values of C and fit the train data into the model. We observe that for C being 1 or 10 or 100, the model has not improved much and got a decent accuracy score. We can similarly change different parameters like kernel to rbf or poly etc., and pick the one that fits the requirement.

Hope this helps! thank you!!

References: An Introduction to Statistical Learning: With Applications in R

Machine Learning Svm Support Vector Machine Hyperplane Kernel