SVM

Support Vector Machine is a supervised machine learning algorithm which can be used for both regression and classification tasks but is most commonly used for solving classification problems.

SVM comes in the form of SVC (Support Vector Classification) and SVR (Support Vector Regression)

For linearly separable data SVMs work amazingly well.

For data that’s almost linearly separable, SVMs can still be made to work pretty well by using the right value of C (the error rate) which controls the tradeoff between smooth decision boundary and classifying training points correctly.

For data that’s not linearly separable, we can project data to a space where it is almost linearly separable and then work to classify classes.

Types of SVM

Linear SVM: Linear SVM is used for linearly separable data. If a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

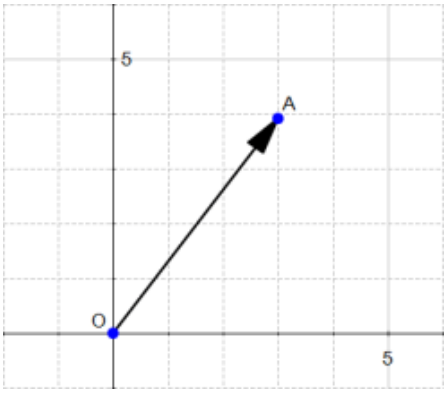
Non-linear SVM: Non-Linear SVM is used for non-linearly separated data. If a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier. Non-linear SVM means that the boundary that the algorithm calculates doesn't have to be a straight line. The benefit is that you can capture much more complex relationships between your data points without having to perform difficult transformations on your own. The downside is that the training time is much longer as it's much more computationally intensive.

Both Regression and Classification problems can have linear and nonlinear data. So whether its regression problem or classification problem kernel tells whether we want a linear or nonlinear separation.

Important Terms

Vectors

Vectors are mathematical quantity, an object that has both a magnitude and a direction. A point in the 2D plane can be represented as a vector between origin and the point.



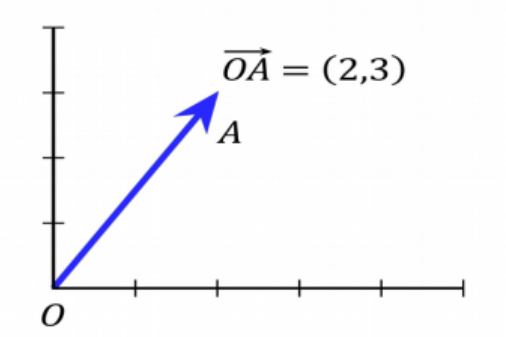
If we say that the point at the origin is the point O(0,0), then the vector above is the vector OA

Length of a vector

The magnitude or length of a vector x is written ‖x‖ and is called its norm.

Fig.-1

$\vec{OA}$ is a vector and length between O and A is its magnitude.



Length of vector $x(x_1, x_2, x_3)$ *is calculated as :*

$$\|x\| = \sqrt{x_1^2 + x_2^2 + x_3^2}$$

Direction of a vector

Direction of vector $x(x_1, x_2, x_3)$ *is calculated as:*

$$\left\{ \frac{x_1}{\|x\|}, \frac{x_2}{\|x\|}, \frac{x_3}{\|x\|} \right\}$$

cos(θ)=u1/ ‖u‖=3/5=0.6

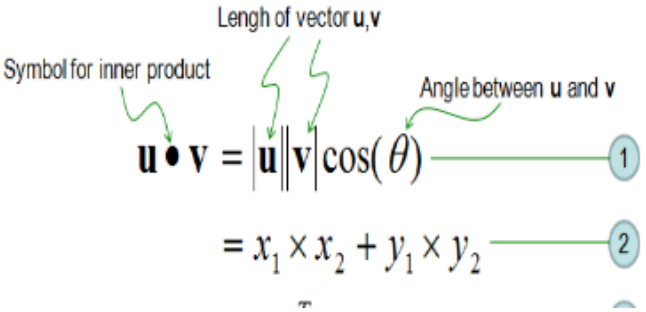
cos(α)=u2/ ‖u‖=4/5=0.8

The direction of u (3,4) is the vector w (0.6,0.8)

Dot Product

Dot product between two vectors is a scalar quantity. It tells how to vectors are related.

Two vectors u and v and their dot product is calculated as:

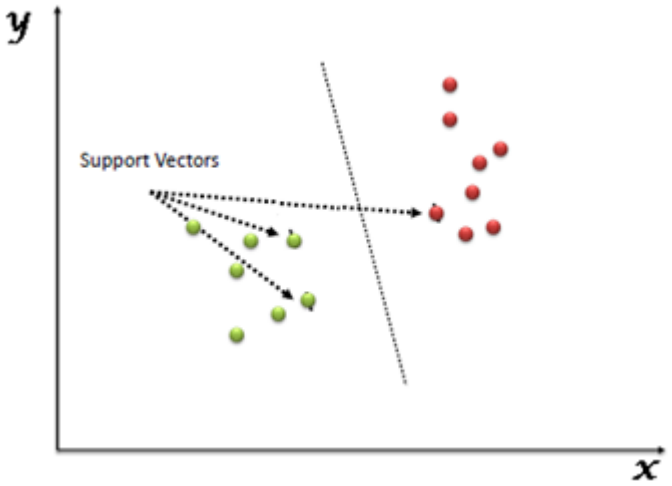


Support Vectors

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector.

Support Vectors if removed would alter the position of the dividing hyperplane. Since these vectors support the hyperplane they are called support vectors.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors and hence algorithm is termed as Support Vector Machine.



The support vectors can be on the negative and positive hyperplanes or outside them.

Hyperplane

A hyperplane is a decision boundary that differentiates the two classes. Hyperplane is basically a plane that linearly divide the n-dimensional data points in two classes

The dimension of the hyperplane depends on the number of input features in the dataset. The complexity of the solution will increase with the rising number of features.

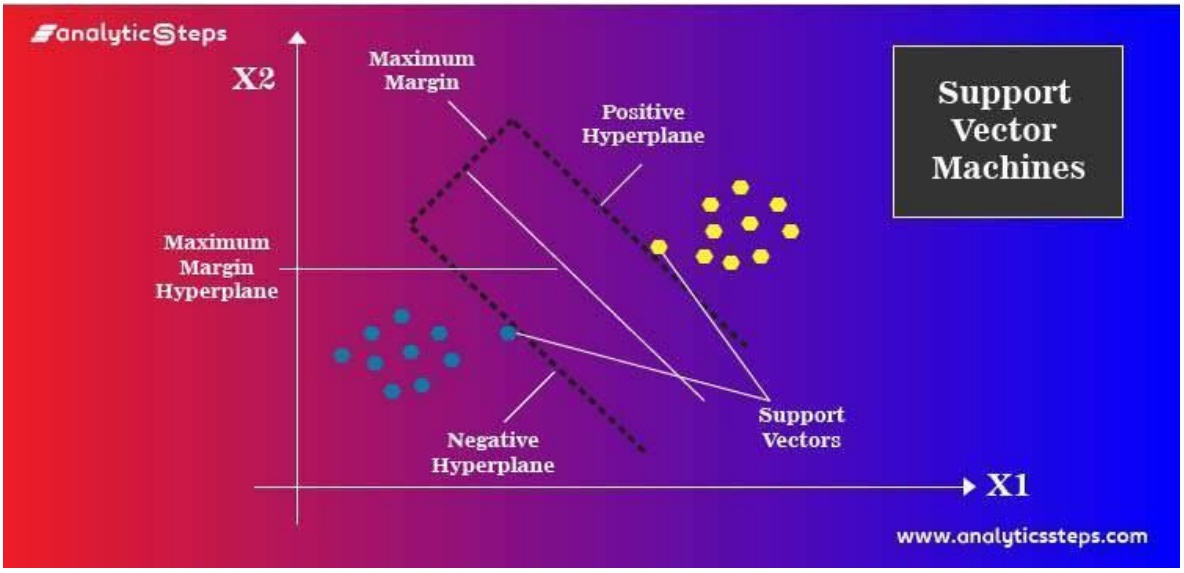
In one dimension, a hyperplane is called a point

In two dimensions, a hyperplane is a line

In three dimensions, a hyperplane is a plane

In more than three dimensions, you can call it a hyperplane. So hyperplane is a line in more than 3 dimensions.

In SVM there are three lines which helps to classify data points which are a + ve hyperplane, a - ve hyperplane and a maximum margin hyperplane. + ve and - ve hyperplane are also called boundary lines and maximum margin hyperplane is also called decision boundary.



If the positive hyperplane and negative hyperplanes are at a distance of +e and –e respectively, equation of the two lines are

Wx+b=+e

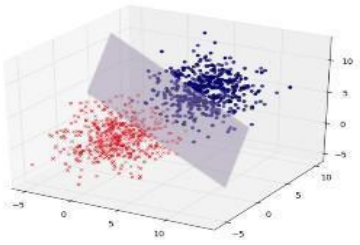
Wx+b=-e

The line equation and hyperplane equation are same which is

wx+b = 0 or ax+c = 0

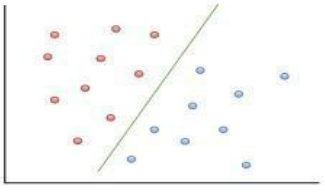
$\mathbf{w}^T\mathbf{x} = 0$

Hyperplane



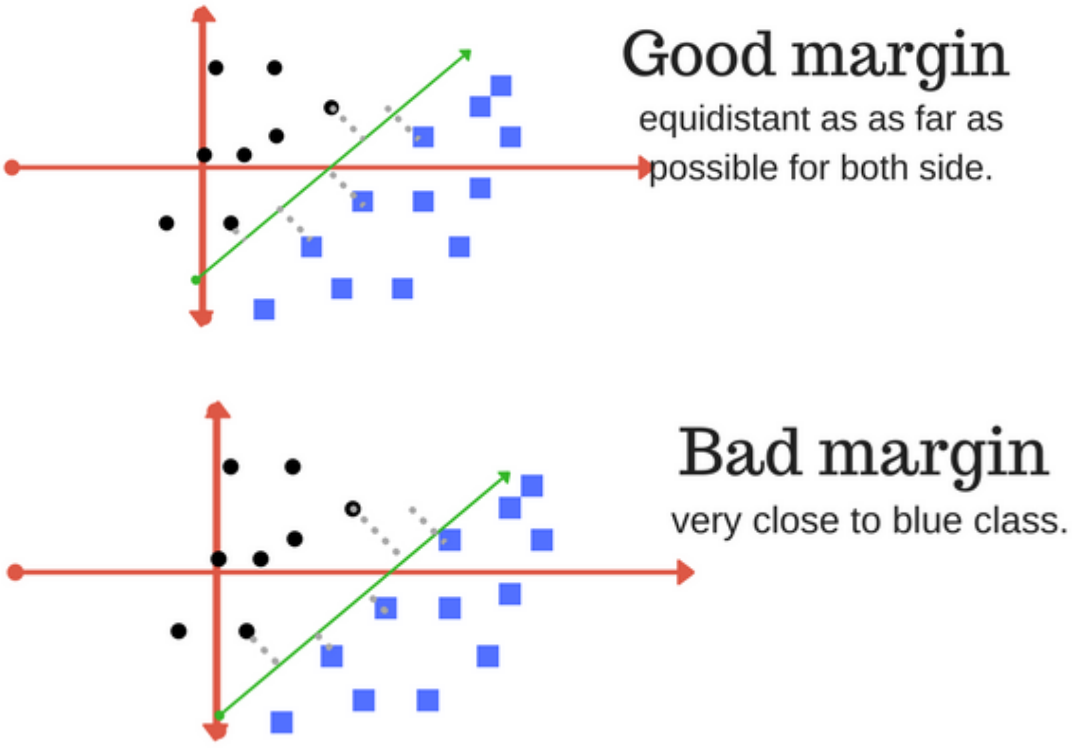
$y = ax + b$

Line



Margin

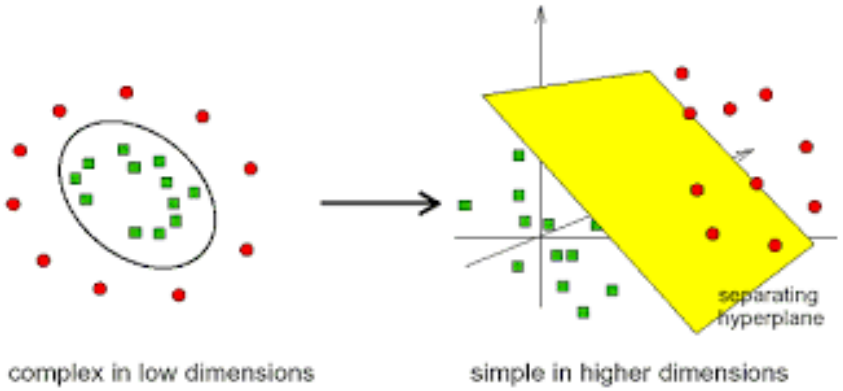
A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes. A good margin allows the points to be in their respective classes without crossing to other class.



Kernel

SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form.

A kernel helps us find a hyperplane in the higher dimensional space without increasing the computational cost. In simple words the kernel trick helps to projects the data into higher dimension where it can be separated by a hyperplane and then project back to lower dimensions.



Some important kernels are Linear kernel (the decision boundary would be linear and two-dimensional), Polynomial kernel (this allows curved lines in the input space), Radial Basis Function (It creates complex regions within the feature space), Sigmoid and Precomputed

C (error rate)

This allows you to dictate the tradeoff between:

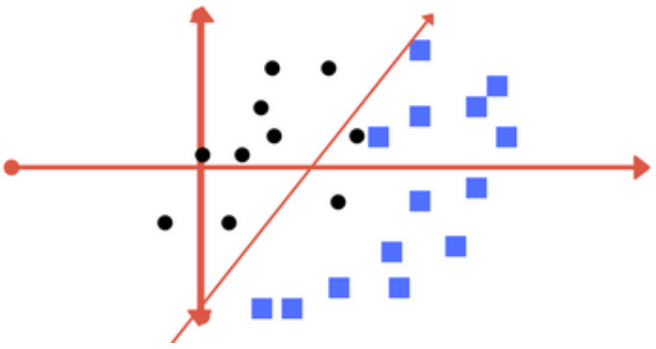
- Having a wide margin.

- Correctly classifying training data.

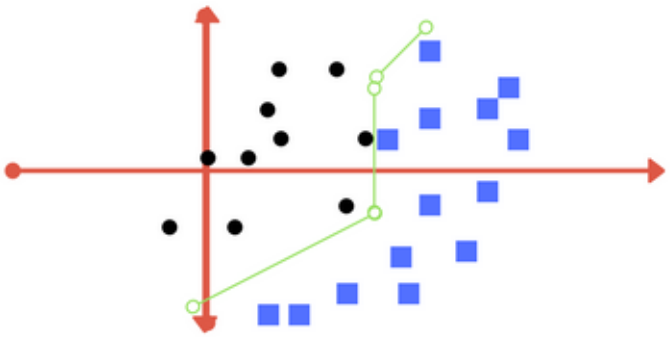
It controls the tradeoff between smooth decision boundary and classifying training points correctly.

The regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM how much you want to avoid misclassifying each training example.

A large value of c means you will get more training points correctly. In other sense large value of c means you will get more intricate decision curves trying to fit in all the points. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.



low regularization value



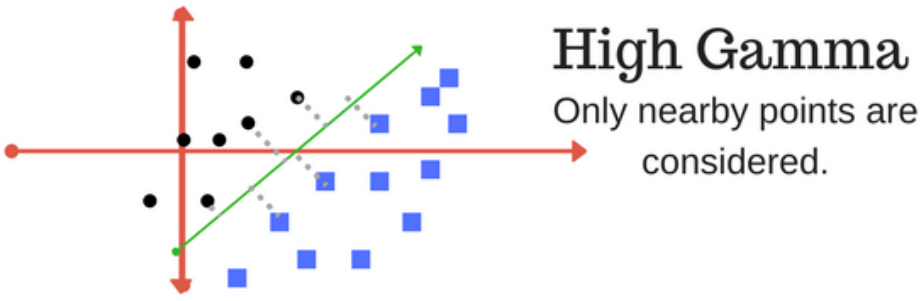
high regularization value

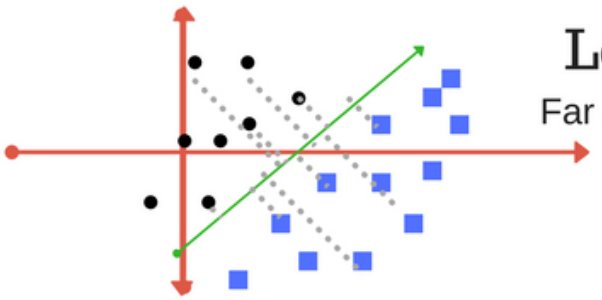
Gamma

It defines how far the influence of a single training example reaches. If it has a low value it means that every point has a far reach and conversely high value of gamma means that every point has close reach.

If gamma has a very high value, then the decision boundary is just going to be dependent upon the points that are very close to the line which effectively results in ignoring some of the points that are very far from the decision boundary. This is because the closer points get more weight and it results in a wiggly curve as shown in previous graph. On the other hand, if the gamma value is low even the far away points get considerable weight and we get a more linear curve.

In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.





Low Gamma

Far away points are also considered.

SVC

Separation of classes. That's what SVC do. SVC creates best hyperplane that segregates the two classes and then it predicts/classifies the target using that hyperplane.

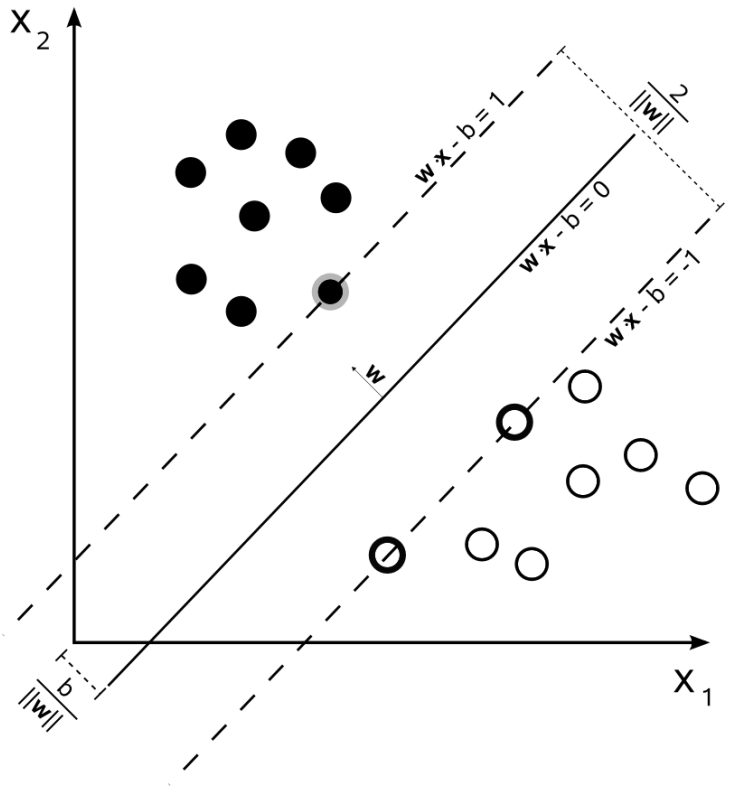
In its most simple type, SVC's are applied on binary classification. For multiclass classification, the same principle is utilized. The multiclass problem is broken down to multiple binary classification cases, which is also called one-vs-one. It basically divides the data points in class x and rest. Consecutively a certain class is distinguished from all other classes. The number of classifiers necessary for one-vs-one multiclass classification can be retrieved with the formula (n*(n-1))/2 with n being the number of classes. In the one-vs-one approach, each classifier separates points of two different classes and comprising all one-vs-one classifiers leads to a multiclass classifier.

How SVC works?

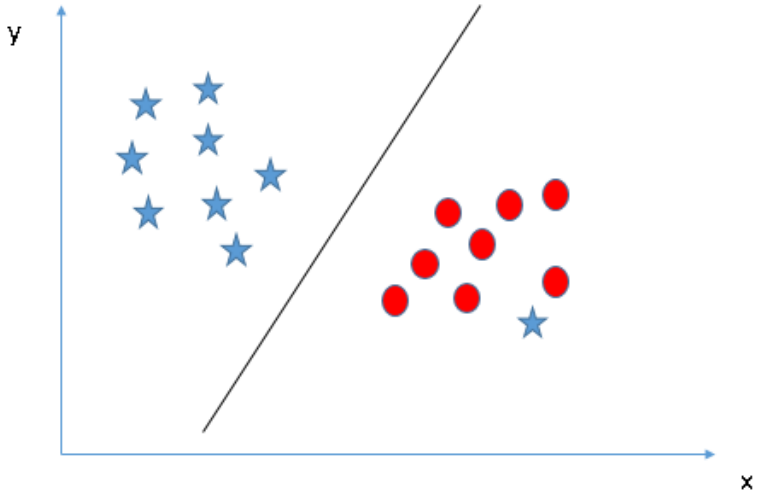
SVC first finds the points closest to the line from both the classes. These points are called support vectors. Using these support vectors, it gets the + ve class hyperplane and the - ve class hyperplane. Now there can be multiple hyperplanes/decision boundaries to segregate the classes in n-dimensional space but we need to find out the best hyperplane/decision boundary that helps to classify the data points most accurately. The best hyperplane is the one that has the maximum margin.

This means we want to maximize the distance between the hyperplane defined by wTx+b=1 and wTx+b=−1. This distance is equal to 2/‖w‖ which is also called the margin and hence thus the optimization problem in SVC is to maximize the margin (the maximum distance between the two decision boundaries) i.e.

max (2/‖w‖)



The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. This is also called soft margin as it tolerates a few dots to get misclassified



There are few rules for separating classes which are used during training and predictions

For each vector x(i) (data points in plane) one of the below is met.

if w.x+b=0 then we get the maximum margin hyperplane

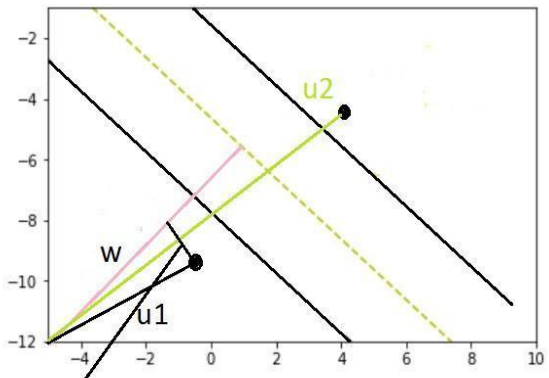
if w.x+b=1 then we get positive class hyperplane

if w.x+b=-1 then we get negative class hyperplane

if w.x+b > 1 then we get positive class data point

if w.x+b < 1 then we get negative class data point

w and b are parameters which are adjusted and learnt through the training process. The final values of w and b are used for predictions.



projection (P)

u1 and u2 are unknown points

w is final vector

if $w^T u1+b > 0$ then (+) class

if $w^T u1+b < 0$ then (-) class

if $w^T u1+b = 0$ then its on decision boundary

From Andrew Ng

P. $||\theta|| >= 1$ if yi=1

P. $||\theta|| <= -1$ yi=0

SVR

SVR is a regression algorithm and works on similar principles as SVM

In the case of SVR, the goal is to find a curve that minimizes the deviation of the points to it. In other sense the goal is to find a hyperplane with boundary lines (that are apart by a distance of ϵ) that has maximum number of points within the boundary lines.

SVR technique relies on kernel functions to construct the model. The commonly used kernel functions are Linear, Polynomial, Sigmoid and Radial Basis.

How SVR works?

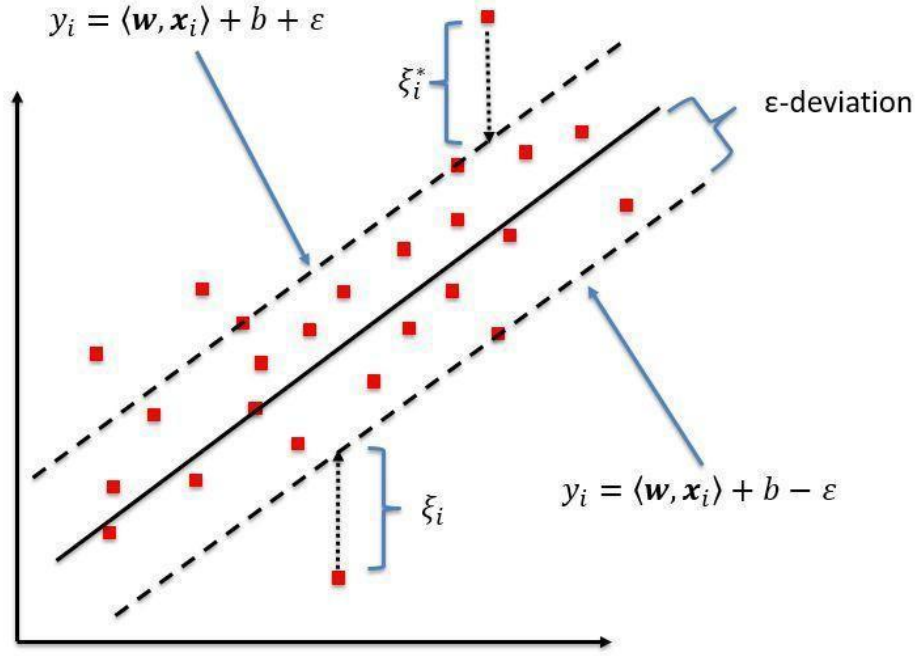
The basic idea behind SVR is not to care about the prediction as long as the error is less than certain value. Hence, our objective when we are moving on with SVR is to basically consider the points that are within the decision boundary lines and have the least error rate or are within the margin of tolerance. SVR gives us the flexibility to define how much error is acceptable in our model and will find a hyperplane to fit the data. As the algorithm doesn't work for all data points, SVR tries to fit as many data points as possible without violating the margin. Our best fit line is the hyperplane that has a maximum number of points.

What we are trying to do here is basically trying to decide a decision boundary at 'ξ' distance from the original hyper plane such that data points closest to the hyper plane or the support vectors are within that boundary line. Thus the decision boundary is our Margin of tolerance to consider data points for the model or in simple terms that we are going to take only those points which have least error rate. Instances that fall within the margin do not incur any cost, that's why we refer to the loss as epsilon-insensitive. Instances outside the margin of a hyperplane incur costs in the optimization, so the optimization goal in SVC is to minimize this cost.

This gives us the optimization problem of

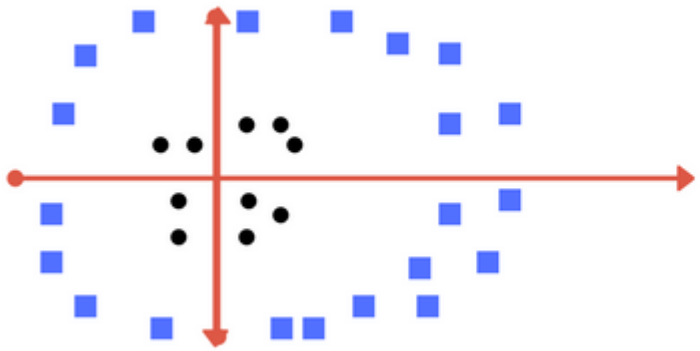
$$\min \quad R(\omega, \xi, \xi^*) = \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

where the parameter C is the regulator which is determined by the user and it influences a tradeoff between an approximation error and the weights vector norm $||\omega||$ and ξ_i and ξ_i^* are slack variables that represent the distance from actual values to the corresponding boundary values

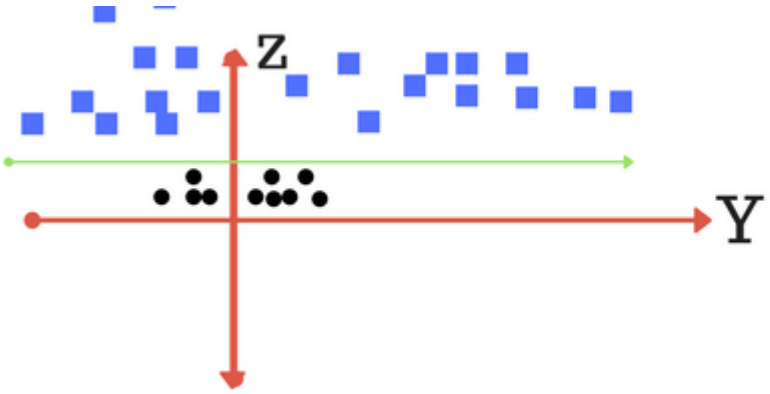

$y_i = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b + \varepsilon$
ξ_i^*
ε-deviation
ξ_i
$y_i = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b - \varepsilon$

Examples

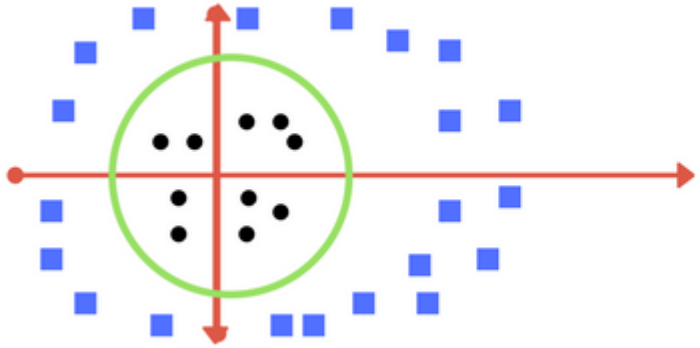
Example 1: Clearly, there is no line that can separate the two classes in this x-y plane. So what do we do?



We apply transformation and add one more dimension as we call it z-axis. Let's assume value of points on z plane, $w = x^2 + y^2$. In this case we can manipulate it as distance of point from z-origin. Now if we plot in z-axis, a clear separation is visible and a line can be drawn.



When we transform back this line to original plane, it maps to circular boundary as shown in below image. These transformations are called kernels.



Thankfully, you don’t have to guess/ derive the transformation every time for your data set. The sklearn library's SVM implementation provides it inbuilt.

Example 2: What if data plot overlaps? Or, what in case some of the black points are inside the blue ones? Which line among 1 or 2? should we draw?

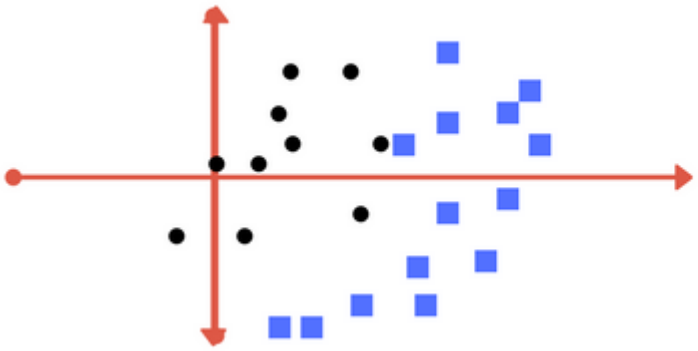


Image 1

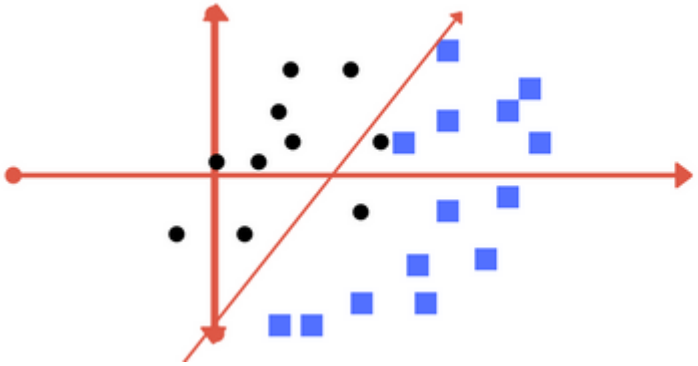
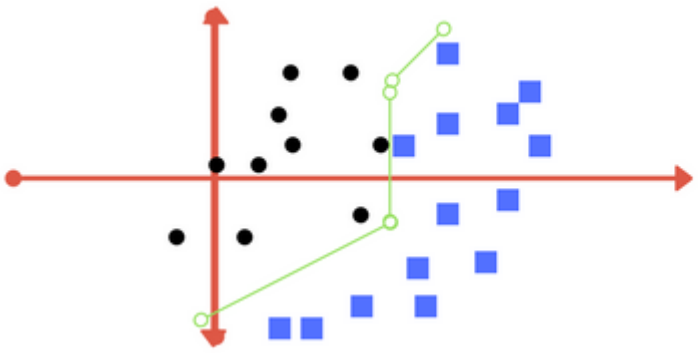
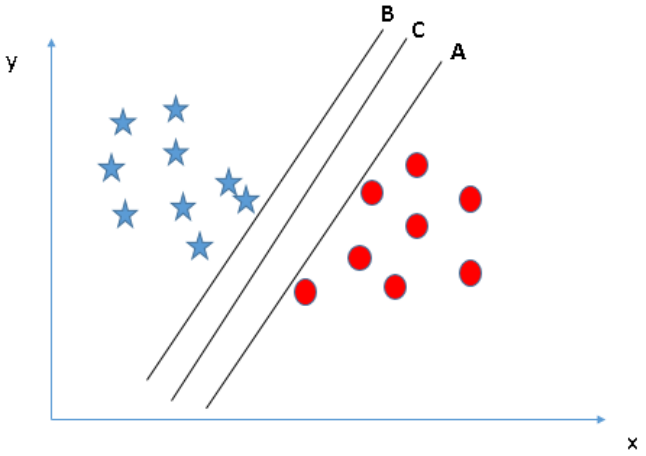


Image 2

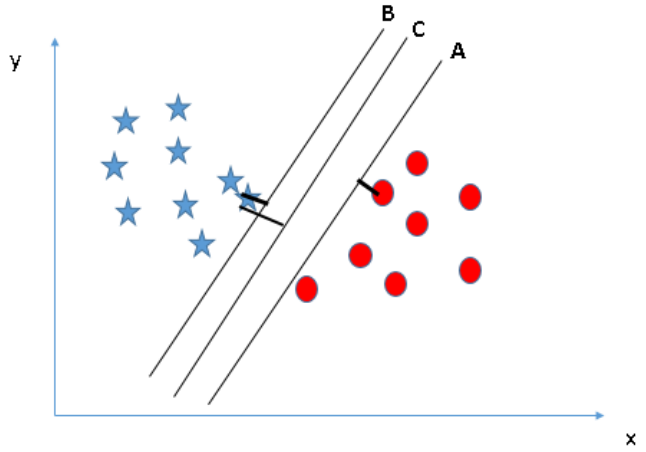


Which one do you think? Well, both the answers are correct. The first one tolerates some outlier points. The second one is trying to achieve 0 tolerance with perfect partition.

Example 3: Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

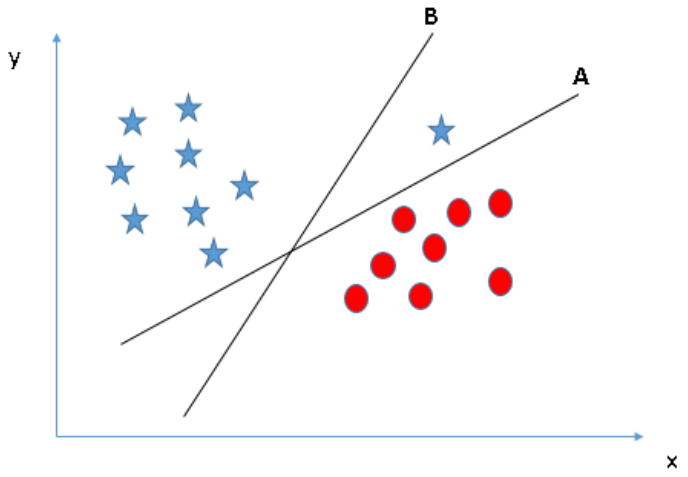


Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin.



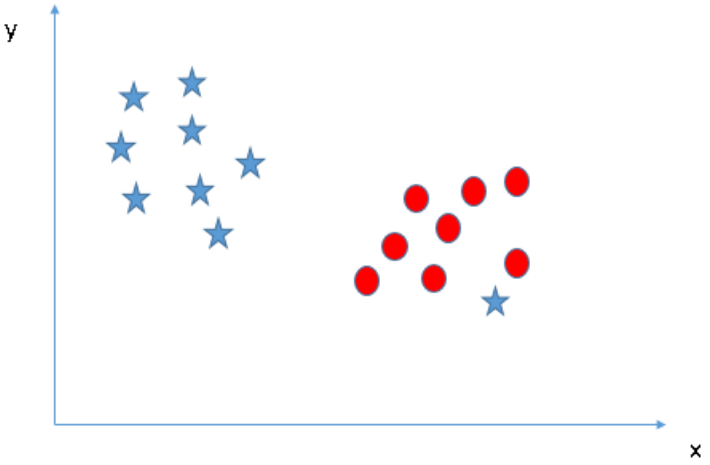
Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin, then there is high chance of miss-classification.

Example 4: Here, we have two hyper-planes (A, B) and all are segregating the classes well. Now, how can we identify the right hyper-plane?



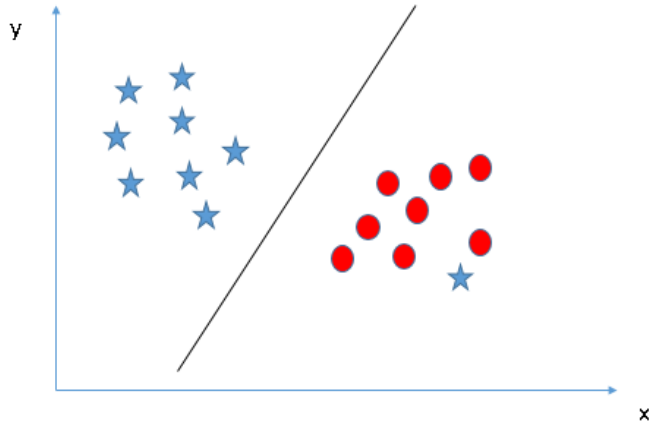
Some of you may have selected the hyper-plane B as it has higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

Example 5: Segregate the two classes using a straight line

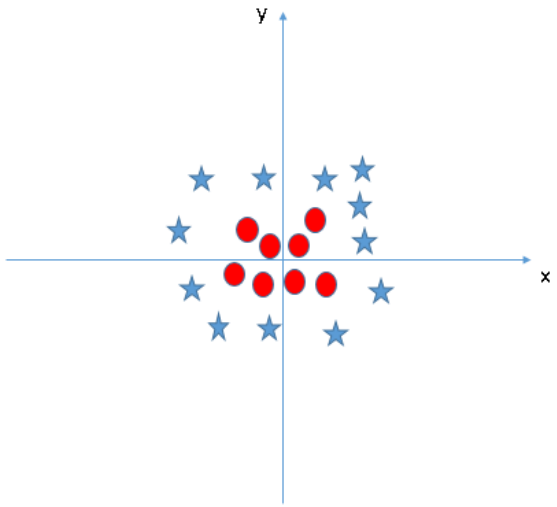


As I have already mentioned, one star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum

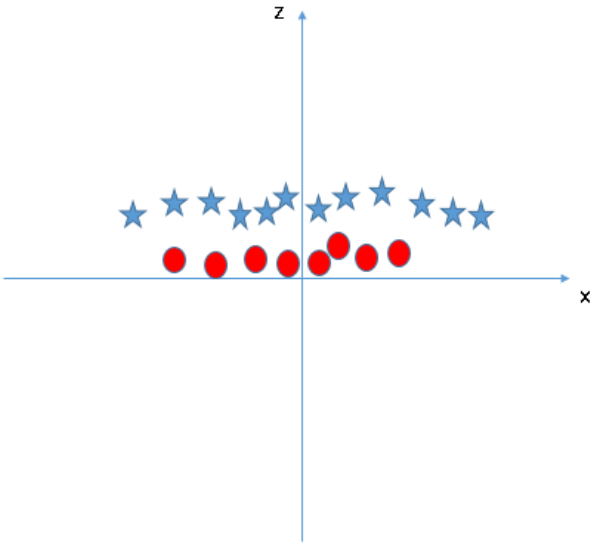
margin. Hence, we can say, SVM classification is robust to outliers.



Example 6: In the scenario below, we can't have linear hyper-plane between the two classes as this is a nonlinear hyperplane case, so how does SVM classify these two classes?



In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the kernel trick. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:


z
x