**SUPPORT VECTOR MACHINES(SVM)**

[Ajay Yadav](https://medium.com/@ajayphroust.ay?source=post_page-----c9ef22815589--------------------------------)

[Oct 20, 2018·10 min read](https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589?source=post_page-----c9ef22815589--------------------------------)

Introduction:

Support Vector Machine are perhaps one of the most popular and talked about machine learning algorithms.They were extremely popular around the time they were developed in the 1990s and continue to be the go-to method for a high performing algorithm with little tuning. In this blog we will be mapping the various concepts of SVC.

Concepts Mapped:

***1. What is SVM?***

***2.The ideology behind SVM.***

***3.Intuition development.***

***4.Terminologies used in SVM.***

***5. Hyperplane(Decision surface ).***

***6.Hard margin SVM.***

***7.Soft margin SVM.***

***8.Loss Function Interpretation of SVM.***

***9.Dual form of SVM.***

***10. What is Kernel trick?***

***11.Types of kernels.***

***12. Pros and cons of SVM.***

***13. Preparing data for SVM.***

***14. Model application***

1. What is SVM?

Support vector machines so called as SVM is a ***supervised learning algorithm*** which can be used for classification and regression problems as support vector classification (SVC) and support vector regression (SVR). It is used for smaller dataset as it takes too long to process. In this set, we will be focusing on SVC.

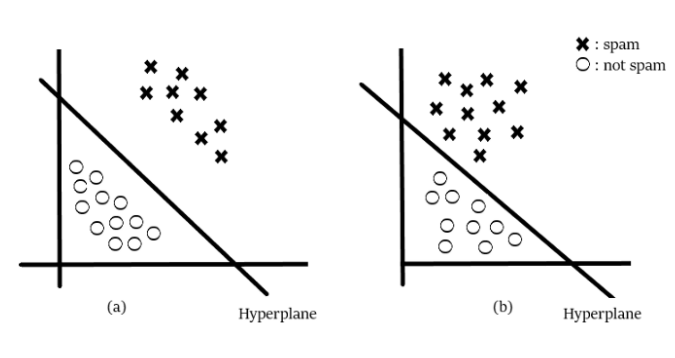
2. The ideology behind SVM:

SVM is based on the idea of finding a hyperplane that best separates the features into different domains.

3. Intuition development:

Consider a situation following situation:

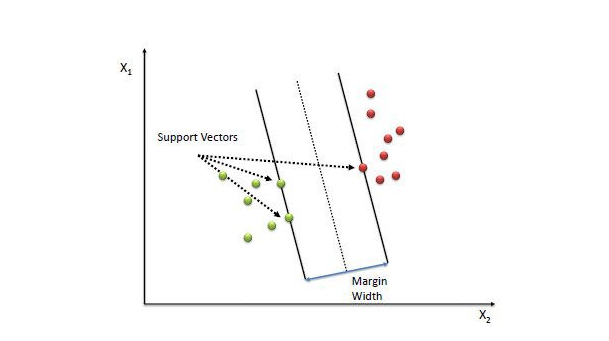
There is a stalker who is sending you emails and now you want to design a function( hyperplane ) which will clearly differentiate the two cases, such that whenever you received an email from the stalker it will be classified as a spam. The following are the figure of two cases in which the hyperplane are drawn, which one will you pick and why? take a moment to analyze the situation ……



I guess you would have picked the fig(a). Did you think why have you picked the fig(a)? Because the emails in fig(a) are clearly classified and you are more confident about that as compared to fig(b). Basically, SVM is composed of the idea of coming up with an ***Optimal hyperplane*** which will clearly classify the different classes(in this case they are binary classes).

4. Terminologies used in SVM:

The points closest to the hyperplane are called as the ***support vector points*** and the distance of the vectors from the hyperplane are called the ***margins***.



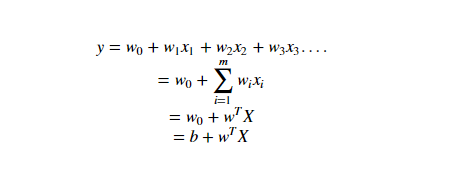
The basic intuition to develop over here is that more the farther SV points, from the hyperplane, more is the probability of correctly classifying the points in their respective region or classes. SV points are very critical in determining the hyperplane because if the position of the vectors changes the hyperplane’s position is altered. Technically this hyperplane can also be called as **margin maximizing hyperplane**.

# 5. Hyperplane(Decision surface ):

For so long in this post we have been discussing the hyperplane, let’s justify its meaning before moving forward. The hyperplane is a function which is used to differentiate between features. In 2-D, the function used to classify between features is a line whereas, the function used to classify the features in a 3-D is called as a plane similarly the function which classifies the point in higher dimension is called as a hyperplane. Now since you know about the hyperplane lets move back to SVM.

Let’s say there are “m” dimensions:

thus the equation of the hyperplane in the ‘M’ dimension can be given as =



where,

Wi = vectors(W0,W1,W2,W3……Wm)

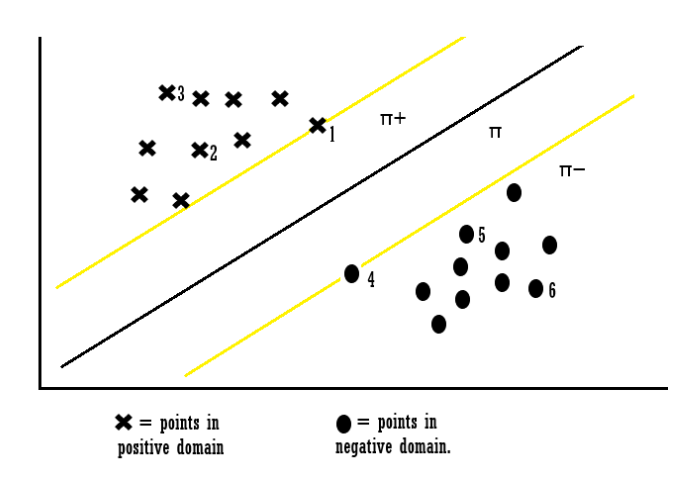
b = biased term (W0)

X = variables.

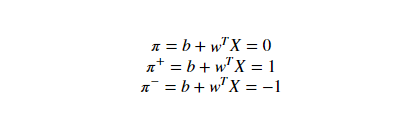
# 6. Hard margin SVM:

Now,

Assume 3 hyperplanes namely (π, π+, π−) such that ‘π+’ is parallel to ‘π’ passing through the support vectors on the positive side and ‘π−’ is parallel to ‘π’ passing through the support vectors on the negative side.



the equations of each hyperplane can be considered as:

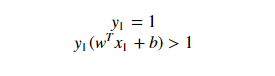


for the point X1 :



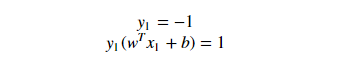
Explanation: when the point X1 we can say that point lies on the hyperplane and the equation determines that the product of our actual output and the hyperplane equation is 1 which means the point is correctly classified in the positive domain.

for the point X3:



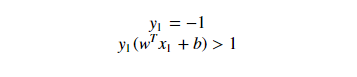
Explanation: when the point X3 we can say that point lies away from the hyperplane and the equation determines that the product of our actual output and the hyperplane equation is greater 1 which means the point is correctly classified in the positive domain.

for the point X4:



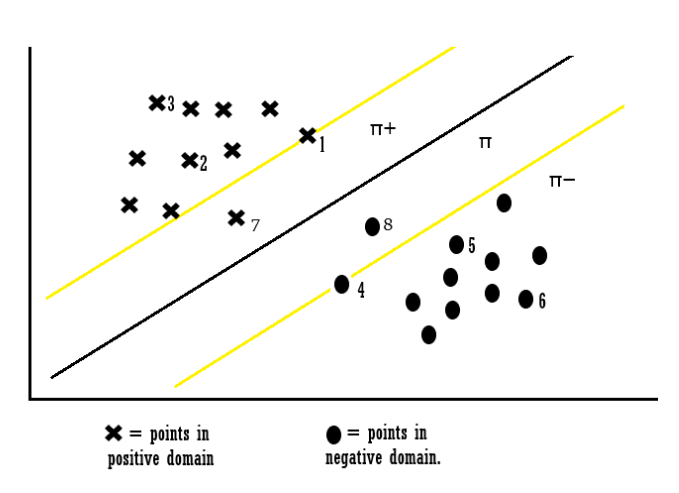
Explanation: when the point X4 we can say that point lies on the hyperplane in the negative region and the equation determines that the product of our actual output and the hyperplane equation is equal to 1 which means the point is correctly classified in the negative domain.

for the point X6 :

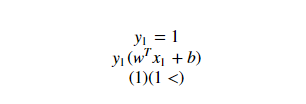


Explanation: when the point X6 we can say that point lies away from the hyperplane in the negative region and the equation determines that the product of our actual output and the hyperplane equation is greater 1 which means the point is correctly classified in the negative domain.

Let’s look into the constraints which are not classified:



for point X7:



Explanation: When Xi = 7 the point is classified incorrectly because for point 7 the wT + b will be smaller than one and this violates the constraints. So we found the misclassification because of constraint violation. Similarly, we can also say for points Xi = 8.

Thus from the above examples, we can conclude that for any point Xi,

**if Yi(WT\*Xi +b) ≥ 1:**

**then Xi is correctly classified**

**else:**

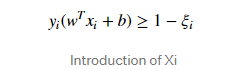
**Xi is incorrectly classified.**

So we can see that if the points are linearly separable then only our hyperplane is able to distinguish between them and if any outlier is introduced then it is not able to separate them. So these type of SVM is called**as hard margin SVM**(since we have very strict constraints to correctly classify each and every datapoint).

# 7. Soft margin SVM:

We basically consider that the data is linearly separable and this might not be the case in real life scenario. We need an update so that our function may skip few outliers and be able to classify almost linearly separable points. For this reason, we introduce a new **Slack variable (**ξ ) which is called Xi.

if we introduce ξ it into our previous equation we can rewrite it as



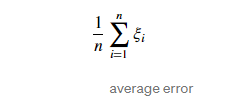
***if ξi= 0,***

***the points can be considered as correctly classified.***

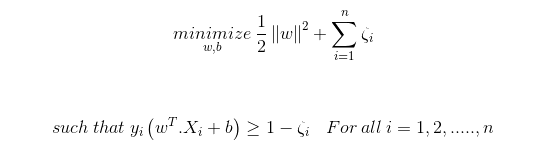
***else:***

***ξi> 0 , Incorrectly classified points.***

so if ξi> 0 it means that Xi(variables)lies in incorrect dimension, thus we can think of ξi as an error term associated with Xi(variable). The average error can be given as;



thus our objective, mathematically can be described as;

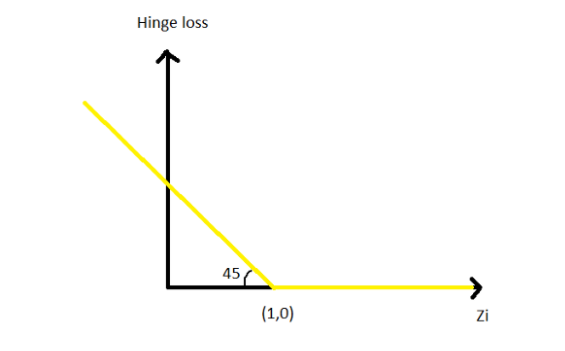


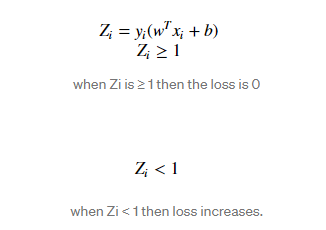
where ξi = ςi

**READING: To find the vector w and the scalar b such that the hyperplane represented by w and b maximizes the margin distance and minimizes the loss term subjected to the condition that all points are correctly classified.**

This formulation is called the Soft margin technique.

# 8. Loss Function Interpretation of SVM:

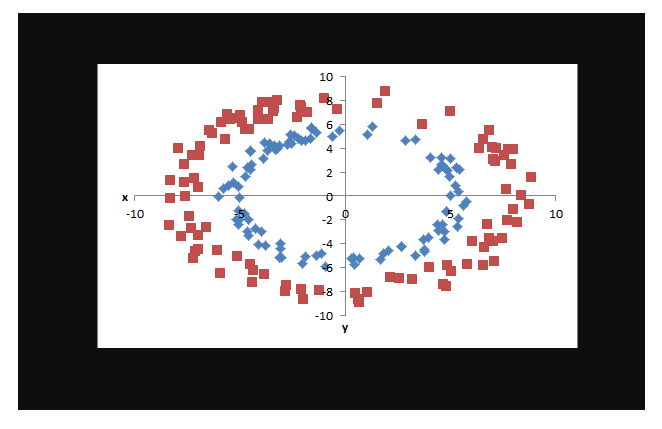




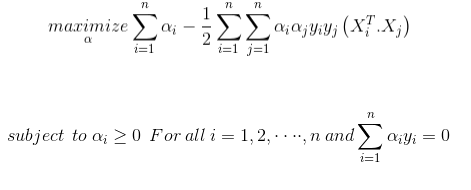
thus it can be interpreted that hinge loss is max(0,1-Zi).

# 9. Dual form of SVM:

Now, let’s consider the case when our data set is not at all linearly separable.



basically, we can separate each data point by projecting it into the higher dimension by adding relevant features to it as we do in logistic regression. But with SVM there is a powerful way to achieve this task of projecting the data into a higher dimension. The above-discussed formulation was the **primal form of SVM** . The alternative method is dual form of SVM which uses **Lagrange’s multiplier** to solve the constraints optimization problem.



**Note:**

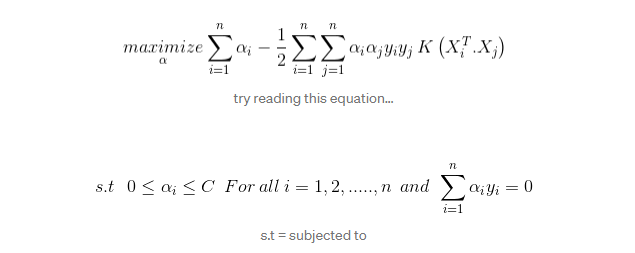
**If αi>0 then Xi is a Support vector and when αi=0 then Xi is not a support vector.**

Observation:

1. To solve the actual problem we do not require the actual data point instead only the dot product between every pair of a vector may suffice.
2. To calculate the “b” biased constant we only require dot product.
3. The major advantage of dual form of SVM over Lagrange formulation is that it only depends on the **α**.

# 10. What is Kernel trick?

Coming to the major part of the SVM for which it is most famous, the**kernel trick**. The kernel is a way of computing the dot product of two vectors **x** and **y** in some (very high dimensional) feature space, which is why kernel functions are sometimes called “generalized dot product.



Applying kernel trick means just to the replace dot product of two vectors by the kernel function.

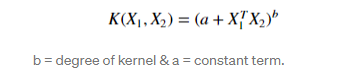
11. Types of kernels:

1. linear kernel
2. polynomial kernel
3. Radial basis function kernel (RBF)/ Gaussian Kernel

We will be focusing on the polynomial and Gaussian kernel since its most commonly used.

Polynomial kernel:

In general, the polynomial kernel is defined as ;



in the polynomial kernel, we simply calculate the dot product by increasing the power of the kernel.

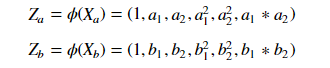
Example:

Let’s say originally X space is 2-dimensional such that

Xa = (a1 ,a2)

Xb = (b1 ,b2)

now if we want to map our data into higher dimension let’s say in Z space which is six-dimensional it may seem like



In order to solve the solve this dual SVM we would require the dot product of (transpose) Za ^t and Zb.

Method 1:

traditionally we would solve this by :



which will a lot of time as we would have to performs dot product on each datapoint and then to compute the dot product we may need to do multiplications Imagine doing this for thousand datapoints….

Or else we could simply use

Method 2:

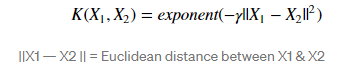
using kernel trick:



In this method, we can simply calculate the dot product by increasing the value of power. Simple isn’t it?

## Radial basis function kernel (RBF)/ Gaussian Kernel:

Gaussian RBF(Radial Basis Function) is another popular Kernel method used in SVM models for more. RBF kernel is a function whose value depends on the distance from the origin or from some point. Gaussian Kernel is of the following format;



Using the distance in the original space we calculate the dot product (similarity) of X1 & X2.

Note: similarity is the angular distance between two points.

## Parameters:

1. C: Inverse of the strength of regularization.

Behavior: As the value of ‘c’ increases the model gets overfits.

As the value of ‘c’ decreases the model underfits.

2. γ : Gamma (used only for RBF kernel)

Behavior: As the value of ‘ γ’ increases the model gets overfits.

As the value of ‘ γ’ decreases the model underfits.

# 12. Pros and cons of SVM:

## Pros:

1. It is really effective in the higher dimension.
2. Effective when the number of features are more than training examples.
3. Best algorithm when classes are separable
4. The hyperplane is affected by only the support vectors thus outliers have less impact.
5. SVM is suited for extreme case binary classification.

## cons:

1. For larger dataset, it requires a large amount of time to process.
2. Does not perform well in case of overlapped classes.
3. Selecting, appropriately hyperparameters of the SVM that will allow for sufficient generalization performance.
4. Selecting the appropriate kernel function can be tricky.

# 13. Preparing data for SVM:

## 1. Numerical Conversion:

SVM assumes that you have inputs are numerical instead of categorical. So you can convert them using one of the most commonly used “[**one hot encodin**](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html)**g ,**[**label-encoding**](http://scikit-learn.org/stable/modules/preprocessing_targets.html#preprocessing-targets)**etc**”.

## 2. Binary Conversion:

Since SVM is able to classify only binary data so you would need to convert the multi-dimensional dataset into binary form using ([**one vs the rest**](http://scikit-learn.org/stable/modules/multiclass.html#one-vs-the-rest)**method /**[**one vs one method**](http://scikit-learn.org/stable/modules/multiclass.html#one-vs-one)) conversion method.

# 14. Model Application:

Since this post is already been too long, so I thought of linking the coding part to my Github account([**here**](https://github.com/Ajayay/SVM_on_AFFR)).

# REFERENCES:

* [Andrew Ng’s notes on SVM.](http://cs229.stanford.edu/notes/cs229-notes3.pdf)
* [Sklearn page on SVM](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html)