**Support Vector Machine (SVM) Classifier**

Machine learning algo in which we are working in vector space.

Complex.

**Binary classifier:** Separates only into two groups at a time (+ and -)

**Objective:** To find the best separating hyperplane also referred to as decision boundary.

A graph of mathematical equations

Description automatically generated with medium confidence

A drawing of a graph

Description automatically generated

Green line is the best separating hyperplane. The distance of the perpendicular bisector of the hyperplane must be the greatest to the associated data points for the hyperplane to be the best separating hyperplane.

A drawing of arrows and lines

Description automatically generated

The purple line is also a hyperplane but the distances between its perpendicular bisector to the associated data points are not the greatest possible so it is not the best separating hyperplane.

A graph drawn on a white board

Description automatically generated

After finding the best separating hyperplane, we can take unknown data points. If a point lies on the right of the best separating hyperplane it is a positive sample and if it lies on the left it is a negative sample.

Since SVM is a binary classifier, we need to have linear data to use it. If data is non-linear then we can’t find the best separating hyperplane. Below is an example of data points where we can’t find the best separating hyperplane.

A graph of mathematical equations

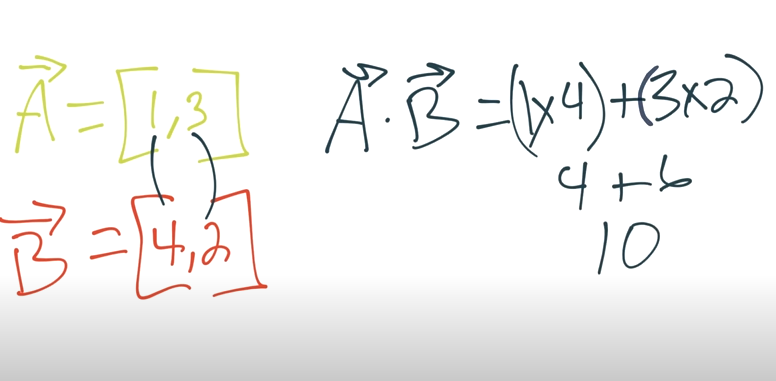
Description automatically generated with medium confidence

**Understanding Vectors:** Direction and Magnitude

A graph of mathematical equations

Description automatically generated

Dot Product A.B



**Support Vector Assertion**

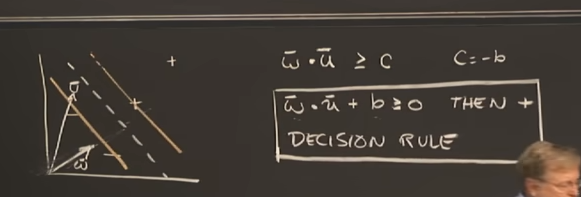
A diagram of mathematical equations

Description automatically generated with medium confidence

‘W’ is a vector pointing towards the decision boundary (best separating hyperplane) and is perpendicular to the decision boundary. W is used to determine whether an unknown point ‘U’ is on the right hand (+) side of the decision boundary or the left hand side (-). For this we project ‘U’ onto ‘W’ using the formula U.W + b. If U.W + b is greater than or equal to 0 then ‘U’ is on the right side (+) else it is on the left side (-). If U.W + b = 0 then ‘U’ is on the decision boundary.

**Finding ‘W’ and ‘b’: (b = bias)**

We know that the unknown ‘U’ is any input point (x1,y1). We must first know what the values for ‘W’ and ‘b’ are to find the solution to the equation U.W + b.



to calculate ‘W’ and ‘b’, we have to put a constraint.

A black background with white symbols

Description automatically generated

Let’s assume that the unknown ‘u’ is some positive point then the result of the equation U.W + b will be 1 or greater than 1 as shown in the image above. Vice-versa if ‘u’ is some negative point.

Introducing a new quantity – Ysub\_i for mathematical convenience  
A blackboard with white text

Description automatically generated

Multiplying both equations by Ysub\_i (both equations become the same)  
A black board with white writing on it

Description automatically generated

A white text on a black background

Description automatically generated

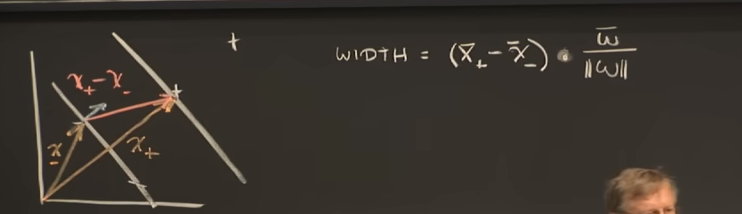
A blackboard with white text

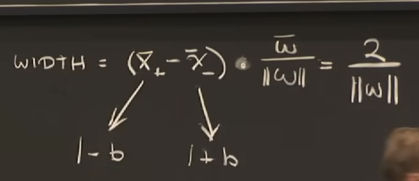
Description automatically generated

Gutters: Line on closest positive and negative samples. Our aim is to find the decision boundary so that the perpendicular line to the decision boundary separating the two gutters is as wide as possible.



(yellow lines are the two gutters, white dotted line is the decision boundary. Gutters must be as wide as possible to the decision boundary)





If X is a positive sample we get 1 - b and if X is a negative sample we get 1 + b using the equation below by subbing +1 and -1 in place of y and getting the values for +X.W and -X.W

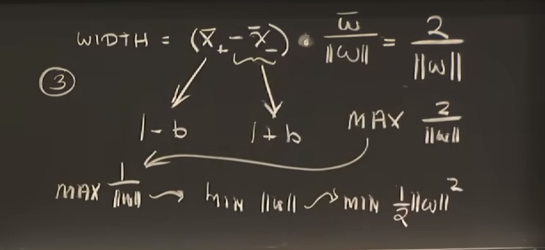
By further simplifying we get 2/||W|| as 1 + b and 1 – b equals 2.

So width of the street is 2/||W|| (2 over magnitude of W)

A white text on a black background

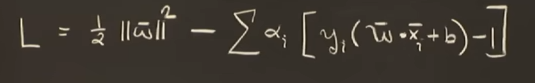
Description automatically generated

**To maximize the distance:** To maximize the distance 2/||W||, it is okay to maximize 1/||W|| and for that we have to minimize ||W|| and to minimize ||W|| it is okay to minimize ½||W||^2 as it is mathematically convenient.



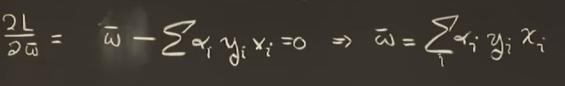
Therefore, to get maximum width we have to minimize ½||W||^2

**Lagrange:**



L = ½||W||^2 – summation of all the constraints

**Derivative of L with respect to W:**



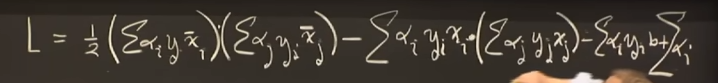
We put the derivative equal to 0 because we are finding extrema points (max or min) and at those points derivative = 0 and by that equation we can get the value of W.

**Derivative of L with respect to b:**

A white chalk drawing on a black background

Description automatically generated

**Subbing in the value of W in the original expression of L:**

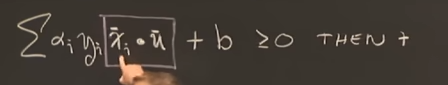


**Simplifying:**

A blackboard with white writing on it

Description automatically generated

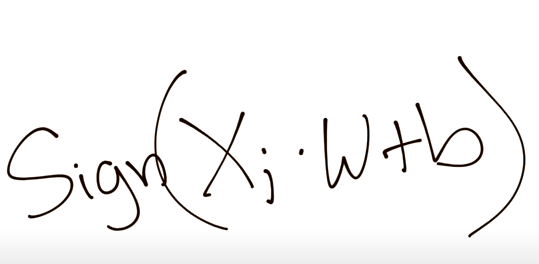
**So finally subbing in the value of W in the decision rule**



So what we can conclude from this is that the decision rule is totally dependent on the dot product of vector X and unknown vector U.

**Support Vector Machine Optimization**

**Formula for classification of feature set**

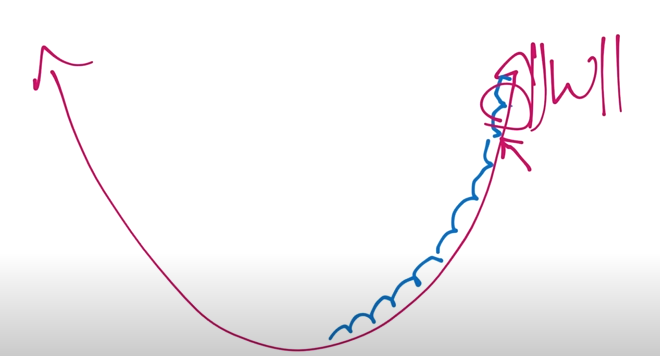


Classification is easy once we optimize for what W and b are.

**Two Main Optimization Objectives:**



<https://www.youtube.com/watch?v=bGCafQT5h1s&list=PLQVvvaa0QuDfKTOs3Keq_kaG2P55YRn5v&index=24> Watch from (8:30 onwards)



The SVM is a Convext Optimization Problem and Python has several modules as shown in the image below to solve this problem. We will be using the logic explain in the video in above link instead to build the SVM by hand and won’t be using these modules currently as we are in learning phase.

