**Lab 5 – Support Vector Machine**

1. Suppose that the following are a set of points in two classes:





Plot them and find the optimal separating line. What are the support vectors, and what is

the margin?

Ans.

1

1

X

#

2

2

M=.36

.5

Showing only nearest 2 points from 2 classes to the Classification Line (other data points are at (2,1) and (1,2) etc. [Use (x,y) i.e. top variable / number is x and bottom is y]. You can either write a program to calculate M using Python or just do a hand calculation. Note that the separation line intersects axes at 1.5. The distance of X or # from the separation line (i.e. M) can be easily calculated using Pythagoran theorem i.e. c^2 = a^2 + b^2 for any right angle triangle etc. # and X shown are 2 support vectors ([1,0] is also a support vector). But the key point is the line that is parallel to the Classification line BUT goes thru the support vectors.

Computation – M^2 + M^2 = 0.5^2 -> 2M^2 = 0.25 -> M = sqrt(0.25/2) = 0.36.

1. Use Perceptron code to classify the data in #1. Compare the classification line with the corresponding classification line that you obtained with SVM in problem 1. Explain the differences between the classification line and corresponding M’s for both cases.

**Ans.** The classification line learned by Perceptron may not be exactly in the middle of the two classes as shown above for the SVM case. Also, the classification line may not be parallel to the line connecting 2 support vectors in the figure shown above in Problem 1 (there was no concept of Support Vectors in Perceptron). This is because Perceptron was not designed to optimize M. It was designed just to classify the two classes.

In case of SVM, the learning was designed to maximize M and also ensure that the classification line is parallel to a line connecting 2 or more support vectors points on both sides of the classes i.e. +ve and -ve support vector points. Moreover, the classification line is exactly in the middle of the 2 classes.

1. Now consider the following non-linear labeled data:

Positive labels –



Negative Labels



Now use the following transformation function to transform the data so that it becomes linear:



Show your transformed data and classify using straight lines.

**Ans.** See below:

A close up of text on a white background

Description automatically generated

1. Consider 2 circles – an inner circle surrounded by an outer circle. Obviously, they cannot be classified with SVM. However, you can apply the Kernel Trick (use a polynomial K) to convert the data into a higher dimensional space (3 in this case) – you can use np.dot etc to transform the data.
2. Use equations for an outer circle and make say 10 data points on the circle. Do the same for an inner circle.
3. Apply a polynomial Kernel to convert the data. Then plot the data.
4. Run SVM on data in 2b.