HW 06 – Support Vector Machines

07/22/2020

ISHAN JOSHI

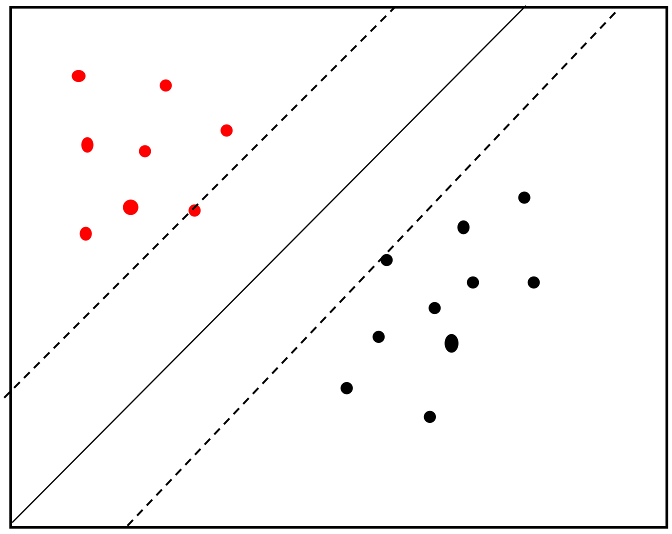
USC ID 3451334123

Support Vector Machines

Introduction

Support Vector Machines work on the presence of margins between the separating hyperplane or line and the closest point on the positive or negative classified data points. This is the case only when the data is linearly separable.

Pictorally,



The solid line shows the separating hyperplane in two dimensions. The dotted lines present the fattest margins on either side of the linearly separable data. The benefit of finding the fattest margin ensures that any future classifications from the test dataset will always have their positive and negatively classified points on either side of the hyperplane, and the probability of incorrect classifications will reduce to a minimum. This will help us achieve high levels of accuracy in classification applications.

Algorithm

The algorithm defines the separating hyperplane mathematically as:

$$w'x + b = 0$$

Where w is the weight matrix and b the bias parameter.

This equation depends on the canonical assumption that ||w||=1, which is the usual assumption for general mathematics.

However, for further computations we need to make a different assumption:

$$|w'x_n \ + b = 1|$$

Here x_n is the closest point to the hyperplane out of the entire dataset. This reduces the mathematical problem to a maximization one where we need to maximize 1 / ||w|| or conversely, we minimize the weights matrix.

Instead of using the absolute function, we can simplify the computation by using N easy computations where we multiply the output label to the above equation. Since the labels are +1 or -1 in the given dataset, multiplying it would do the same job as the absolute value function.

Therefore, our final objective function and constraints are:

Minimize: 0.5 * w' * w

Such that: $(w'x_n + b) * y_n \geq 1$

We then use the Duality Principle and Lagrange Formula to solve the above parameters as a Quadratic Programming Problem.

Execution

1. We use a quadratic programming solving package called cvxopt to find the alpha values that correspond to the support vectors, or datapoints closest to the separating hyperplane. For calculating the weights matrix, we only consider alpha values which are positive or above a predefined threshold. The weights can then be determined as follows:

$$w = \sum_{i=1}^{n} alpha(n) * y_n * x_n$$

   Here, i – Number of support vectors
   Alpha(n) – alpha value corresponding to support vector n
   Y – label
   X – Datapoint

2. The bias term is calculated as follows:

$$b = y_n - w'x_n$$

Here, y and x can correspond to any datapoint.

All the above calculations occur only at optimality.

Kernels:

In cases where the input data is non-linear, or outliers are present, we cannot calculate the quadratic programming inputs the same way. To tackle the non-linearly separable data specifically, we can use kernels. The kernel we use is the radial basis function (RBF) kernel.

Mathematically:

$$RBF = \exp\left(\gamma||x - x'||^2\right)$$

Here x and x' are two separate datapoints. The output of this function is used to calculate the Q matrix that is given to the QP solver as input. The rest of the procedure remains the same as with linearly separable data.

Outputs

Part a:

```
Support Vectors: [array([0.24979414, 0.18230306]), array([0.3917889 , 0.96675591]), array([0.02066458, 0.27003158])]

Weights 2 [[ 7.25005616 -3.86188932]]

[[2.21297168]] 44

Equation: [[ 7.25005616]

 [-3.86188932]] * x + [[2.21297168]] =1
```

Part b:

RBF Kernel:

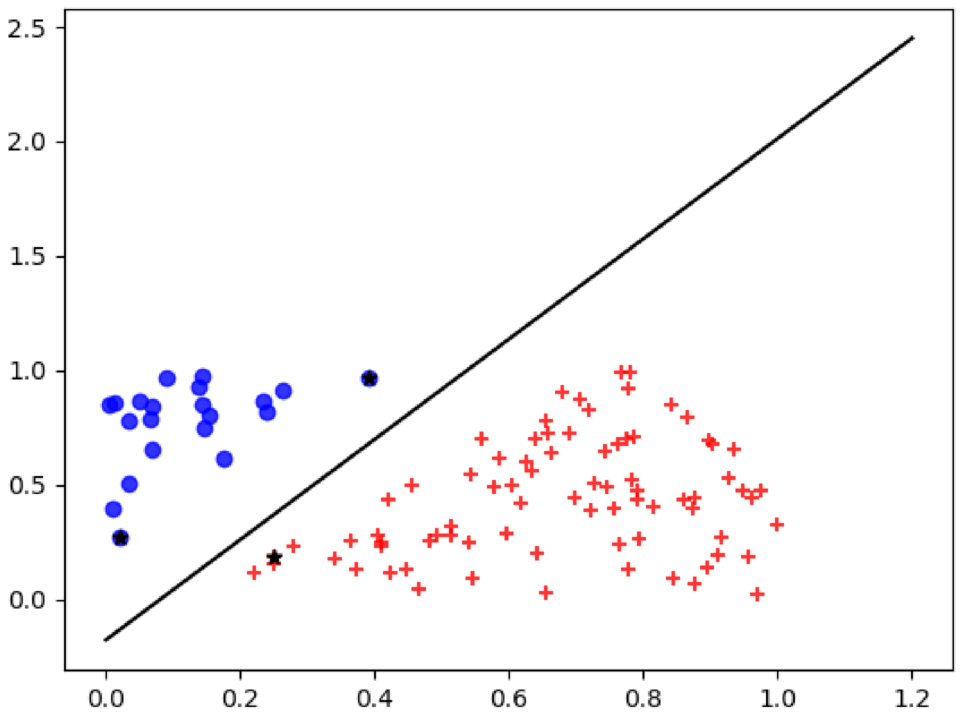
```
Support Vectors: [[ -8.47422847   5.15621613]

 [-10.260969     2.07391791]

 [  1.3393313  -10.29098822]

 [  9.67917724   4.3759541 ]

 [ -9.46760885   2.36139525]

 [ -6.80002274  -7.02384335]

 [  9.90143538  -0.31483149]]
```

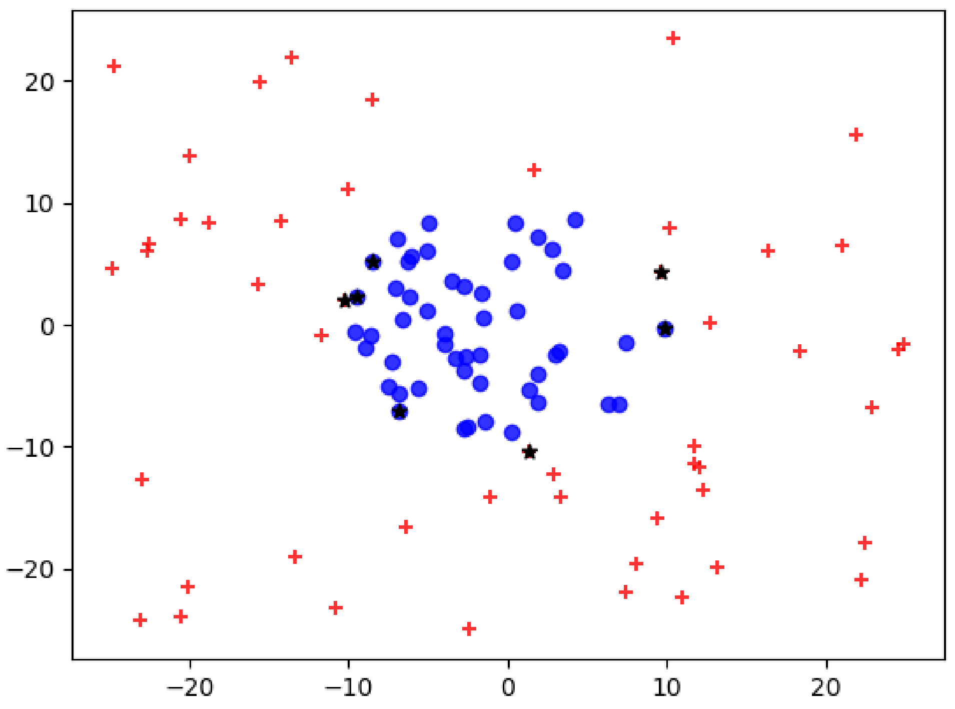
Equation in terms of kernel Function:

$\sum_{i=1}^{n} alpha(n) * Y(n) * K(Zi, Z)$ + -1.00000121 = 1

Where, K(Zi,Z) is the kernel function output of the variable Z with a support vector point Zi. The bias is calculated by plugging a random point Zm from the input dataset.

The black star points in the following graphs show the support vectors:


2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2



Code Challenges:

- Running the QP solver with the correct input parameters was difficult, even more so on a windows setup. The library required math kernel libraries to build wheel files and thus requires a manual installation.
- For the non-linear case, plotting the decision boundary is challenging in two dimensions. However, it is possible to do so in 3D.
- Determining the threshold value for choosing the alpha values is be done based on the values we receive as output from the QP solver. In the case of the assignment, all alphas are positive. Thus, we use the threshold of e^{-5} for choosing alpha values.
- The other factor we need to determine is the gamma operator in the RBF kernel calculation. To achieve optimal results, we use 0.00045 as the value for gamma.