DMW C2 Assignment-1

Shubham Kumar IIT2018115
Raktim Bijoypuri IIT2018125
Aditya Kamble IIT2018126
Aakashdeep IIT2018128
Tejas Mane IIT2018135

VI Semester, Department of Information Technology,

Indian Institute of Information Technology, Allahabad, Prayagraj.

Algorithm:

SVM Classification Algorithm:

SVM is a widely used algorithm in pattern classification problems. SVM classifier classifies the data points into two different classes with a decision boundary or a hyperplane. Minimum distance of a data point from the hyperplane is called the margin and those points which are closest to the hyperplane are called the support vectors. Sometimes, it is easier to find a linear hyperplane to separate the training data points into two different classes but sometimes the data points are distributed in such a way that it becomes very difficult to find a linear hyperplane which separates the data points into two different classes. In such cases soft margin or kernel trick is used. In the case of soft margin, we find a linear hyperplane with minimum number of misclassifications. In the case of kernel trick, we do not find a linear hyperplane, instead, some features are generated from within the dataset and then a non-linear hyperplane is found which separates the data points in two different classes.

In this assignment we have implemented a variation of the SVM algorithm and it is more or less the same as the implementation of paper 2.

- The update on gammaB will be counted only when the update is above the minC percent which by default is set to 0.1%. So for particular GammaB our code will keep on searching for GammaA in the order of heuristics mentioned in paper 2 till the minC doesn't occur in Gammas.
- There could be the case that none of the values for GammaB will result in proper change in Gammas; the case will be counted as pass. In such cases since the gammas are exactly the same as last iteration, searching the gammaB using it's heuristic as discussed in paper 2 will result in same iteration therefore, we will select the next gammaB randomly from all gammas and do this on the subsequent iteration as well till the gamma's does not effectively change.
- Even on selecting random gamma's it's not guaranteed that the gamma's will change; for this we adjusted the termination condition of code. The iteration will terminate if maximum allowed iterations are completed or the number of continuous passes exceed the maxP limit

- Here maxP is defined as maxP = max(maxP2, (int)(0.1*itr))
- The MaxP2 will be received from the user through an argument of class initiation which by default is set to 10.
- On algorithm mention paper 2 the values of p1 & p2 will be calculated from the average of the respective support vectors, but there could be the iteration (usually among the earlier iterations) where there is no support vector for either p1 or p2 in such case the algorithm can go heywire also in paper 1 they didn't face such conditions, In order to stop this we decided to ignore such changes and count it as pass and also adjust out initial parameter v1, v2 and epsilon such that: epsilon/(v2*m) and 1/(v1) are sufficiently greater than zero, where m is the number of samples in the dataset.

Observation:

We have implemented both RBF and linear kernel for this assignment. The values taken by us are v1=0.0001, v2=0.09, epsilone = ⅔ and itr = 10000 for the following result and for RBF kernel standard deviation was taken 1. In the randomly generated dataset we have usually found an MCC score of about 0.43 in linear kernel and 0.36 in RBF kernel.