

# UNIVERSITY OF WATERLOO



Department of Electrical & Computer Engineering

## Pattern Recognition SYDE-675 Assignment – 3

**Prepared By:**

**Name:** Kunal Taneja

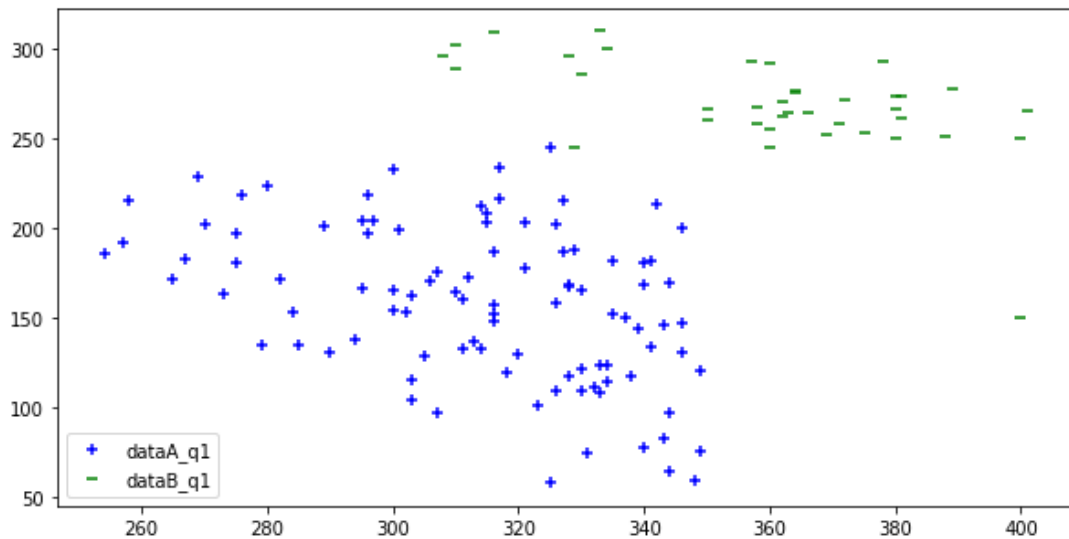
**Student ID:** 20790967

Winter 2019

## Question 1

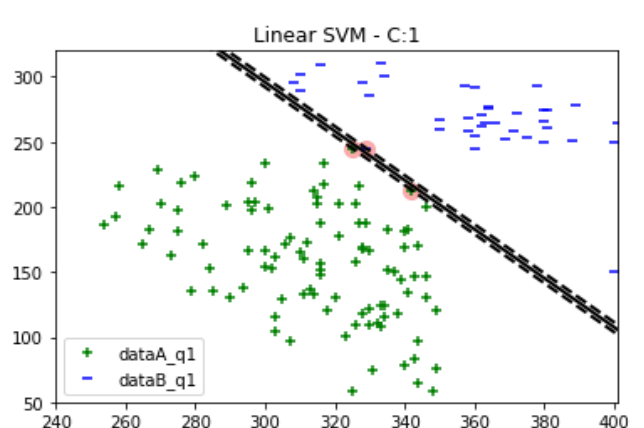
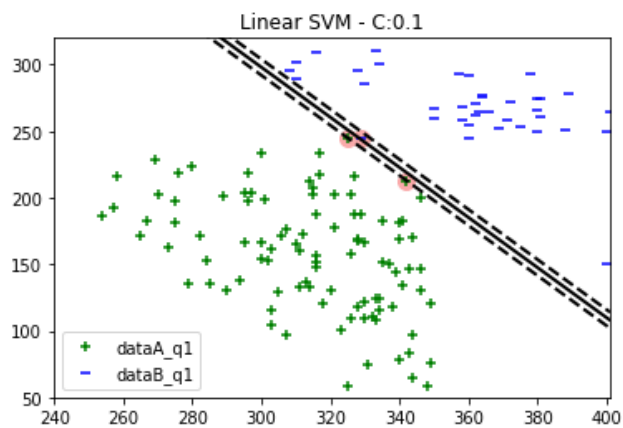
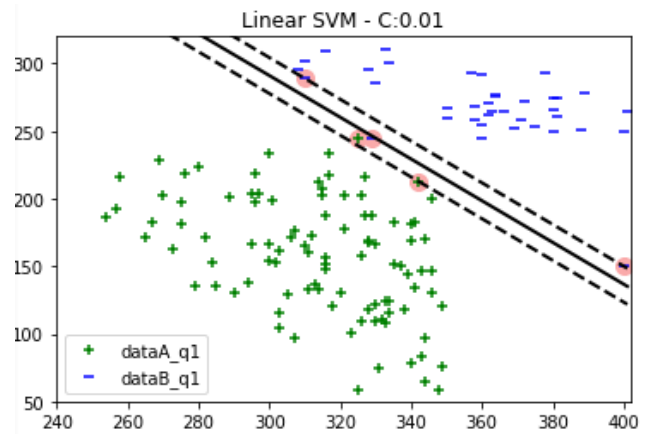
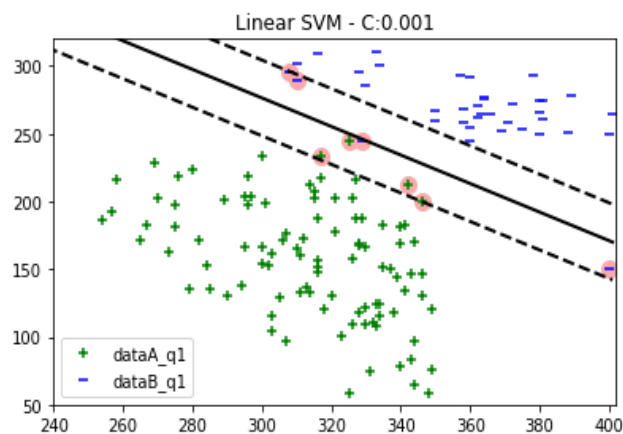
### Part 1:

First, we try to load the dataset and then visualise the data samples on a 2D plane using scatter plot. For q1\_classA and q1\_classB we get the following scatter plot.



From the above scatter plot we visualise that the data is linearly separable.

### Part 2 – Part 4:



**SVM** - A Support Vector Machine (SVM) is a discriminative classifier whose aim is to find a hyperplane in N-dimensional space which classifies all the data points in the space distinctively. In other words, given labelled training data, the algorithm finds an optimal hyperplane which categorizes test examples. In two-dimensional space the optimal hyperplane is a line dividing a plane in two parts where in each class lies on the either side of the line.

In this assignment sklearn's implementation based on based on libsvm is used with linear kernel and C as the penalty parameter(sklearn.svm.SVC).

On comparing the decision boundaries, we find when the value of C is increased, the penalty while training the linear SVM is increased that is SVM model tries to classify almost all the points correctly and hence overfits because the optimization for linear kernel will choose a smaller-margin hyperplane as described in the figure. The gutter width starts decreasing with increasing value of C.

The figure displays the optimal decision boundary for particular C value along with the gutter widths with data samples lying within the gutter circled. Outliers are also circled with pink color.

For example, when  $C = 0.001$  the penalty is quite low, hence more of the data samples are allowed to be misclassified. But when  $C=0.01$ , the model tries to classify most of the data samples correctly while allowing some edge cases to be misclassified such that no overfitting takes place.

As the value of C is increased beyond 0.01 to 0.1 and even 1, the SVM model overfits as it tries to penalise the misclassified examples heavily forcing model to fit all the data perfectly and give minimal training error. As it can be seen from the figure, the model for  $C = 0.1$  and  $C = 1$  tries to classify the points on the boundary line correctly causing small margin hyperplanes.

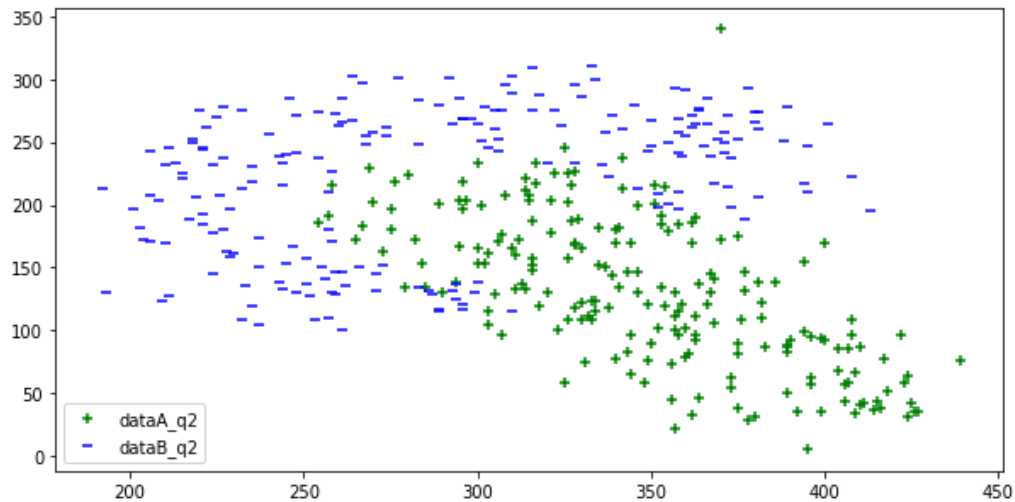
Hence, for this dataset  $C = 0.01$  is the best C value as it tries to classify most of the data correctly without any overfitting of the model. Lower value of C ensures larger-margin separating hyperplane with some misclassifications. But for  $C = 0.1$  and 1, overfitting of the model takes place hence it causes small margin separating hyperplane.

C is the regularisation parameter. For large values of C, the optimization for linear kernel will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

## Question 2

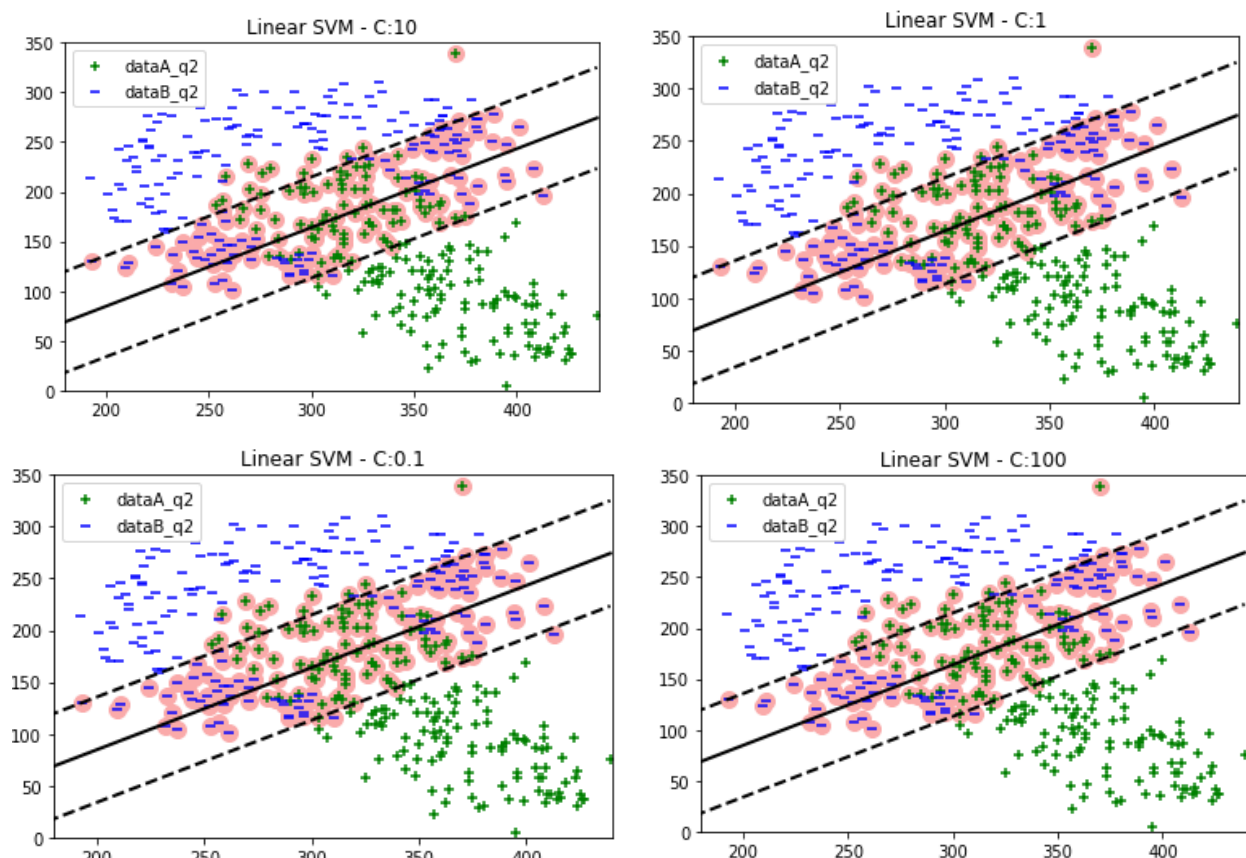
### Part 1:

The datasets classA and classB are loaded and visualised on the 2D plane using scatter plot.



From the above scatter plot we see that this data is not linearly separable and cannot be classified simply by using Linear SVM as there is no straight line that can help in visualising the margin between two classes.

### Part 2:



On visualising the figure, we see that whatever be the value of C, no decision boundary is able to separate the data. Moreover, after fitting the model with different values of C gave almost same results which can be shown by mean accuracy based on 10 times 10-fold results.

Model	Accuracy (10x10 fold)
SVM with C = 0.1	0.8002439024390243
SVM with C = 1	0.8002439024390243
SVM with C = 10	0.7991245781213485
SVM with C = 100	0.7990154445487878

So, using Linear SVM on this dataset we can achieve around 80% of accuracy. But we can use these as weak classifiers and feed these weak classifiers into ADABOOST algorithm to create an ensemble of Linear SVM classifiers to get better accuracies.

Thus, we can choose any SVM model as the weak classifier for ADABOOST M1 algorithm.

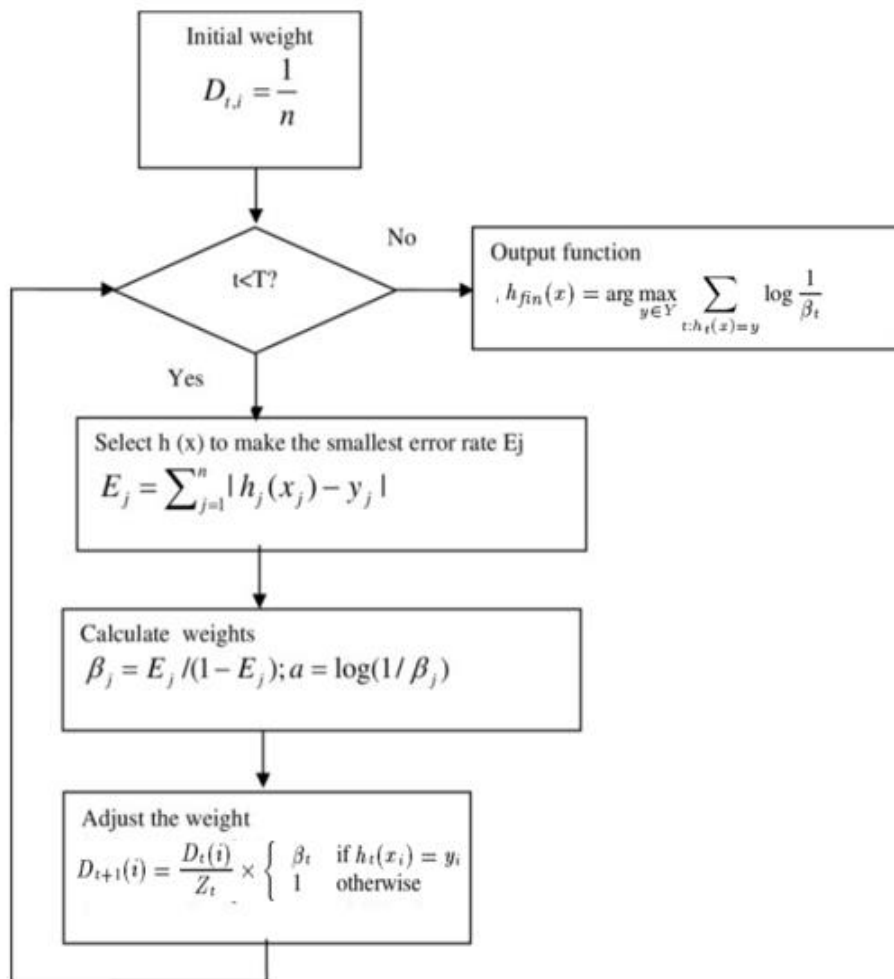
### Part 3:

**ADABOOST** - It focuses on classification problems and aims to convert a set of weak classifiers into a strong one. For this problem the weak classifier is Linear SVM. With the help of 50 weak learner classifiers we try to plot a decision boundary on this dataset such that it can distinguish between two classes in a better way. As this is an ensemble model, we may end up with a curvy decision boundary which is a resultant of 50 weak decision boundaries.

Firstly, we calculate the initial weights as  $1/(\text{number of samples used for training})$ . We select  $T=50$  weak learners. For each iteration, in our algorithm we draw only 100 samples from the dataset (randomly) to train each weak classifier. Calculate the training error by summation of weights of misclassified examples. Now, if the error is greater than 0.5 we discard this weak classifier as it cannot even achieve 50 percent accuracy in binary classification problem. So, if the training error is more than 0.5 we resample the training data and train a new classifier and repeat the same process till we get 50 weak learners with training error less than 0.5.

Using training errors, we calculate beta values which further is used to damp the weights of correctly classified examples. After weight updating, we normalise the new weights such that sum of weights is equal to 1.  $Z_t$  is the normalisation factor. Finally, for correct classifications we perform summation of  $\log_{\text{beta}} \frac{1}{\text{error}}$  values. Major voting is performed for each classifier, that is if  $\log_{\text{beta}} \frac{1}{\text{error}}$  values of class 1 is higher than class 2 then prediction is made as class 1 for this classifier and vice versa.

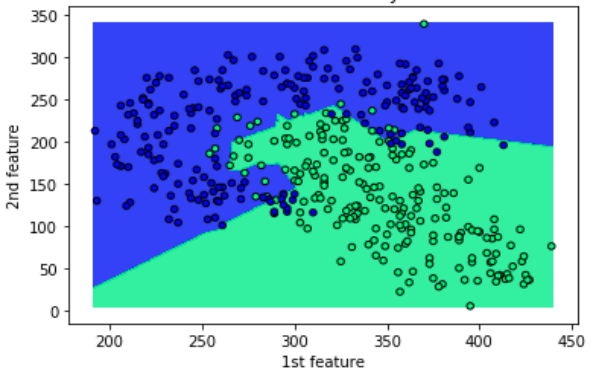
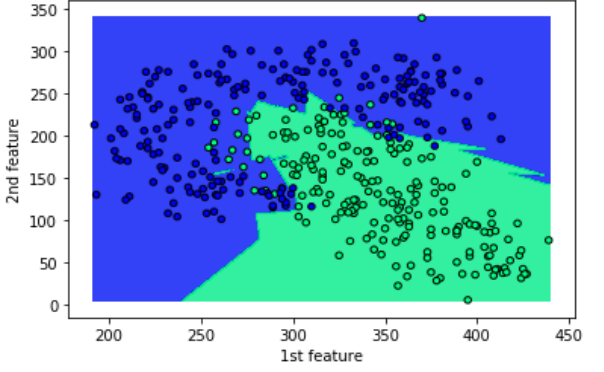
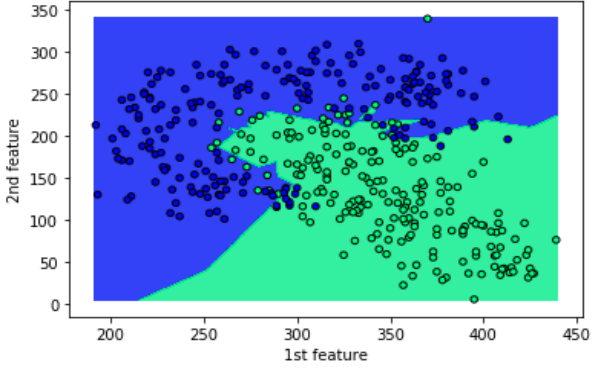
## ADABOOST M1 FLOWCHART:



## Part 4 – Part 5:

Following table shows the decision boundaries for ADABOOST M1 for different selections of weak classifiers. The decision boundary is visualised using the concept of meshgrid. A grid is formed using  $x_{\min}, x_{\max}, y_{\max}, y_{\min}$  and predictions are made for every point in this grid. A step size of 0.5 is chosen while forming the grid. ADABOOST is trained using the whole data and then the predictions are made over the meshgrid.

So, every point in the grid is classified either as classA or classB. Green colors are plotted for every classification in classA and blue for classB. As this is an ensemble of linear SVMs we get a curvy decision boundary which is a visualisation of 50 linear boundaries of 50 weak classifiers. We find that this ensemble works well over this data as in some cases we get accuracy of 90% (10 times 10 fold results for  $C = 100$ ). So we are able to successfully make a strong classifier using several weak classifiers.

Weak Classifier Used	Mean Accuracy & Variance (10 times 10 fold)	Decision Boundary ● Region A ● Region B
Linear SVM with C = 0.1	<b>Mean Accuracy:</b> 0.8910914634146342  <b>Variance:</b> 0.0022819444898869727	<p>ADABOOST M1 Decision Boundary SVM with C =0.1</p> 
Linear SVM with C = 1	<b>Mean Accuracy:</b> 0.8925121951219513  <b>Variance:</b> 0.002213474717430101	<p>ADABOOST M1 Decision Boundary SVM with C =1</p> 
Linear SVM with C = 10	<b>Mean Accuracy:</b> 0.8915182926829266  <b>Variance:</b> 0.0022823644779892927	<p>ADABOOST M1 Decision Boundary SVM with C =10</p> 
Linear SVM with C = 100	<b>Mean Accuracy:</b> 0.9007682926829269  <b>Variance:</b> 0.0021997737953599045	<p>ADABOOST M1 Decision Boundary SVM with C =100</p> 