**Data and Execution procedure**

The dataset considered for this experiment is taken from a file ‘Letter2Class.data’, which is present in the same folder where all other .m files are located and this data was accessed through these Matlab codes. In this data file the row header represents labels. The data consists of 16 types of attributes which represents a character ‘A’ or ‘X’. All the Matlab codes along with the data file are stored in a single folder. For all the questions same dataset is being used.

Using ‘importdata’ function the data is being read from the file. For all the 16 types of attributes representing ‘A’ or ‘X’ a total of 1576 instances are collected of which 789 instances belong to ‘A’ and remaining 787 instances belong to ‘B’.

**Question 1**

**Implementing bagging method and AdaBoost method using MATLAB**

**Bagging**

The bagging method (**b**ootstrap **agg**regat**ing**) is an ensemble learning algorithm that helps to improve the performance of machine learning algorithms. It is useful to reduce the variance of a prediction model.

The Algorithm consists on a bootstraping step, which is implemented by selecting a fixed number of subsets (bags) of the original data set. The subsets are in turn constructed by selecting a fixed number of samples from the original data set, randomly and with replacement.

Then, a parallel training is done for each bag, using a classifier.

Finally, the aggregation step is performed, which consists on using the set of classifiers to perform the selection.

This last step is generally performed by voting and will outperform the individual choices. However, for this assignment, the decision plane was requested, so the classifiers used were linear and then they were combined linearly to make the bagging classifier For this particular implementation, the classifiers used were the fitcdiscr from MATLAB, these classifiers use a fitted discriminant analysis model.

The implementation of the algorithm is in the bagging.m file and the function is called bagging. It takes a dataSet (struct) , NBags (integer) and ratio (double) as inputs and returns hyperplane (array of double, coefficients of the weights) and counters, a struct that contains integers (TP, TN, FP, and FN).

function [hyperplane, counters] = bagging(dataSet,NBags,ratio)

NBags is the number of bags to use and ratio is the proportion of samples with respect to the whole set that each classifier will use.

To try the function’s performance, a test script bagging test.m was implemented. This script can optionally create two clusters of 2D data and labels them and the function perceptron is called. With this, a graphical representation of the algorithm is obtained. The script can alternatively compute all the statistical measurements required for question 3 using the given data set.

**Adaboost**

Adaboost is short for **ada**ptative **boost**ing, this algorithm was created by Freund and Schapire in 1996. It also uses base classifiers and combines them. The main difference is that in adaboost the classifiers are trained sequentially and the performance of each individual (also called weak) classifier will affect the next one.

Samples are given a weight to affect the importance given to them by each classifier. For the first one, all weights are the same and equal to , where N is the number of samples. Then a weak classifier is trained, in this particular implementation, the single node tree classifier from Assignment 2 was used.

Then for the next classifier, the misclassified samples are weighted more so that the classifier favors these samples.

For each subsequent classifier the weights are updates as follows:

First and are calculated, these are the weighted measures of the error rates of each of the base classifiers on the data set and weighting coefficients that give greater weight to the more accurate classifiers when computing the overall output respectively, they are calculated as:

Where is the indicator function that is 1 when its argument is true and 0 otherwise.

The weights for the next classifier are calculated as:

After all classifiers have been trained, the prediction of the adaboost classifier will be

The implementation of the algorithm is in the file adaboost.m. The function is called adaboost. It takes a dataSet (struct) and NClassifiers (integer) as inputs and returns hyperplane (array of double, coefficients of the weights) and counters, a struct that contains integers (TP, TN, FP, and FN).

function [hyperplane counters] = adaboost(dataSet,NClassifiers)

To try the function’s performance, a test script adaboostest.m was implemented. This script creates two clusters of 2D data and labels them and the function adaboostest is called. With this, a graphical representation of the algorithm is obtained. The script can alternatively compute all the statistical measurements required for question 3 using the given data set.

**Question 2**

**Implementing a multilayer perceptron neural network using MATLAB.**

The multilayer perceptron (MLP) is a type of feed-forward Artificial Neural Network (ANN) which comprises of multiple layers of regression models.

Each layer is made of elements called neurons which are node that get a set of variables as inputs, multiplies them by constant numbers called weights, adds a constant to all of them, which is called bias, and then passed that sum into an activation function, for example a sigmoid or tanh. Each subsequent layer repeats the same process using the outputs of the previous layer as the inputs (feed forward fully connected topology). Until the final output is reached, which is in general a function to be approximated, or as in this case, a classification of the input of the network in a set of classes.

To train the network, forward and back-propagation is used, using the formula

Where is the activation function for the hidden layer, is the value at the output of neuron j and is the error at the output k, in this case it is just one output.

A two layer network was implemented where the number of neurons in the hidden layer can be chosen.

Weights were updated according to:

With

The implementation of the algorithm is in the file multilayerperceptron.m The function is called multilayerperceptron. It takes a dataSet (struct) , NNeuronsPerLayer (integer), learningRate (double) and ratioRandomSamples (double) as inputs and returns weights (array of double), which are the weights and bias of the final layer and counters, a struct that contains integers (TP, TN, FP, and FN).

function [weights counters] = multilayerperceptron(dataSet,NNeuronsPerLayer, learningRate,ratioRandomSamples)

To try the function’s performance, a test script multilayerperceptron test.m was implemented. This script computes all the statistical measurements required for question 3 using the given data set

.

**Question 3**

**Write a script to evaluate the bagging, AdaBoost, and MLP methods using MATLAB.**

All three testing scripts create two cluster of 2D data points, label them and call the implemented functions and graph the decision plane when useDataSet is false, the results can be seen in the following Figures.

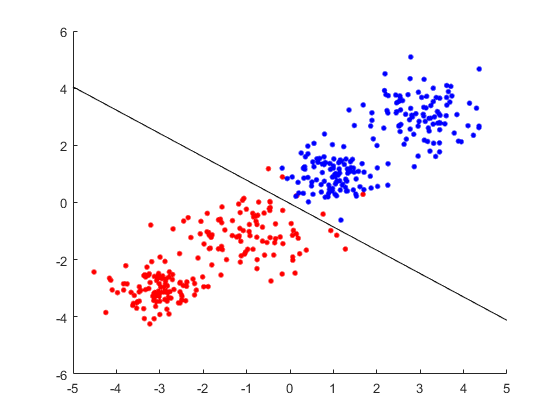


Figure 1: clusters and decision plane for bagging

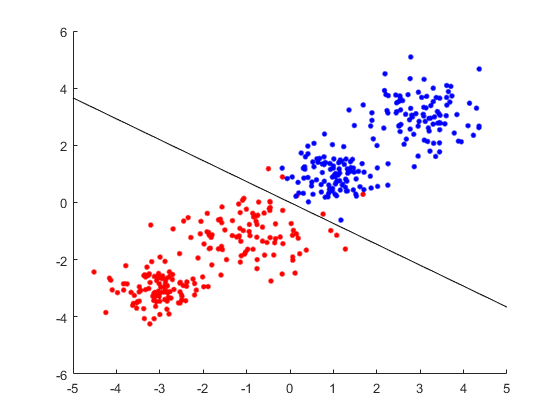


Figure 2: clusters and decision plane for adaboost

When useDataSet is set to true, all three test srcipts perform cross-validation and calculate accuracy, sensitivity, and specificity.

Standard deviation and average are computed in a separate worksheet to be able to see the numbers.

**Cross-validation**

To perform cross-validation, 5 runs for each algorithm are considered, each one using 70 % of the data selected randomly with reposition



Table 1. Cross validation for bagging, including average and standard deviation



Table 2. Cross validation for adaboost, including average and standard deviation



Table 2. Cross validation for multilayerperceptron, including average and standard deviation

**Accuracy**

To calculate accuracy, the following formula is used:

Where:

is true positive = X classified as X  
 is true negative = A classified as A  
 is false positive = A classified as X  
 is false negative = X classified as A

For bagging the accuracy is 0.9968

For adaboost the accuracy is 0.9581

For MLP the accuracy is 0.8515

**Sensitivity**

To calculate sensitivity, the following formula is used:

For bagging the sensitivity is 0. 9975

For adaboost the sensitivity is 0.9226

For MLP the sensitivity is 0. 8249

**Specificity**

To calculate specificity, the following formula is used:

For bagging the specificity is 0. 9962

For adaboost the specificity is 1

For MLP the specificity is 0.8828

**Discussion**

When running adaboosttest.m with 3 classifiers and 5 runs for cross-validations, some of the times the obtained hyperplanes were sets of NaN of + or – Inf. This may be due to the fact that some of the subsets generated to train the cross-validation batches may have been completely explained by less than 3 classifiers. When the next classifier doesn’t have any sample, this is that one of the classifiers was able to correctly classify all the samples, the algorithm has undefined values because of division by 0 so a modification could be done to the algorithm to stop it if this happens. To be able to obtain Table 2, a couple of runs of the script had to be performed.

The implementation of MLP was found to be very sensitive to all of its inputs parameters. Several different configurations were run and the one that got the best performance was 4 neurons in the hidden layer, learning rate 0.01 and 0.01 of the total samples to be randomly selected to update the weights. But as it can be seen in Table 3, the output layer’s coefficients are not stable. It has to be considered though, that there may be equivalent networks given that the combination of the weights on the hidden layers and these output layer’s weights may be the same for different output layer’s weights. Different configurations and/or activation functions should be tested to stabilize the algorithm.

**Auxiliary functions**

An auxiliary function labelsXAtoTarget1minus1 was implemented in the labelsXAtoTarget1minus1.m file which takes the dataset and returns an array of 1’s and -1’s for the labels “X” and “A” respectively.

function targetValues = labelsXAtoTarget1minus1(dataSet)