Implementation Assignment #3

Alireza Mostafizi

Introduction

I coded this assignment in MATLAB. Attached to this report, you can find the MATLAB codes. The assignment is to implement 1) a Decision Stump as a weak learning algorithm, 2) Bagged Decision Stump using, and 3) Boosted Decision Stump incorporating AdaBoost algorithm.

The data used for this assignment is the SPECT data which describes the diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. The data has 22 binary features, and is classified into two categories of normal and abnormal. There are 267 observations in total, 80 of which are used as training data set, and the rest 187 observations are used as testing data set. Table 1 shows the summary of the dataset.

	Training Dataset		Testing Dataset	
Number of Instances	80		187	
	Class 0	Class 1	Class 0	Class 1
	40	40	15	172

Table 1. Data Characteristics

Table 2 shows the number of instances for both training and testing data set which has a specific feature. As you can see in the table, almost all of the features, except the ones that are marked, are fairly incorporated in the training and testing processes, so the decision stump will not be biased towards any specific feature.

		Features																				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Testing Data	Class 0	97	134	103	128	103	134	132	95	119	109	138	128	83	122	146	118	157	158	136	119	109	103
	Class 1	90	53	84	59	84	53	55	92	68	78	49	59	104	65	41	69	30	29	51	68	78	84
Training Data	Class 0	51	67	59	63	56	70	59	58	65	57	64	60	52	64	74	66	72	74	65	62	61	54
	Class 1	29	13	21	17	24	10	21	22	15	23	16	20	28	16	6	14	8	6	15	18	19	26

Table 2. The number of instances over features

In this report, each part of the assignment will be addressed in the following separate sections, and the end results will be discussed at the end of every section.

Part I: Decision Stump

Methodology

The decision stump is one-level decision tree. To pick the best feature, I used maximum Information Gain Algorithm as following:

$$I(Y.X) = H(Y) - H(Y|X)$$
$$= H(Y) - \sum_x P(X = x)H(Y|X = x)$$

Where H represents the entropy calculated as following for a categorical random variable that can take k different values:

$$H(y) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

Where $p_i = P(y = v_i)$ which equals the probability of the categorical variable of y being equal to ith category.

Therefore the objective of finding the best feature for the decision stump can be formulated as following:

$$X^* = \arg\max_X I(Y, X)$$

Results

Doing so on the 22 features presented in the data set, we can state that feature 13 has the most information gain. It also leads to the best accuracies for both training and testing data sets. Figure 1 shown the information gain of different features in the dataset. The maximum value which associates with feature 13 is marked with the red arrow on the graph.

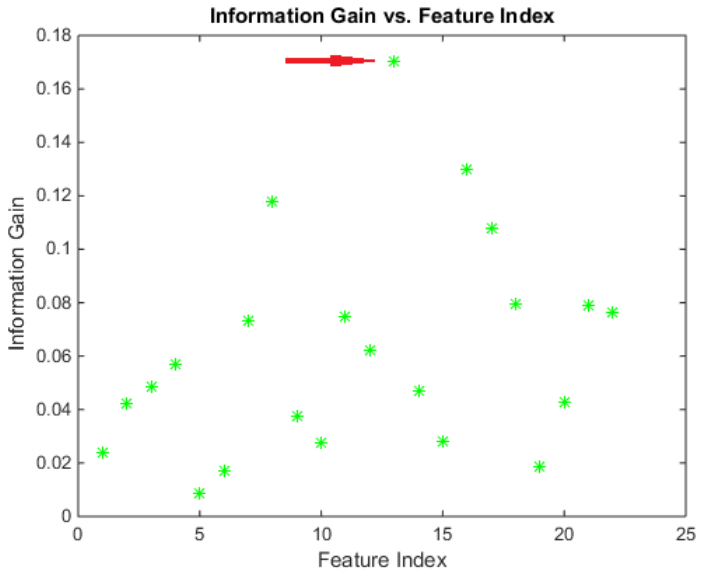
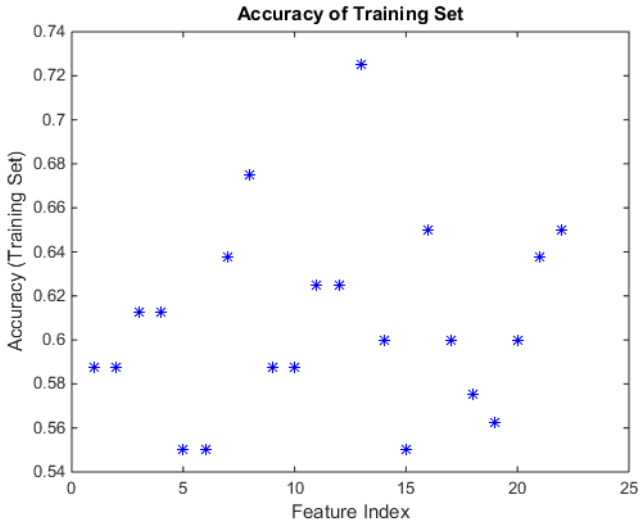
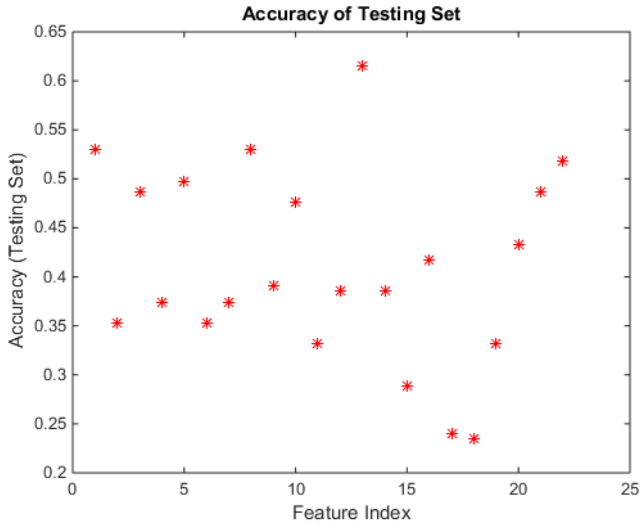


Figure 1. Information Gain over different Features

Figure 2 shows the accuracy of different decision stumps for both training and testing datasets. As shown in the figure, feature 13 also leads to the highest accuracies for both training and testing datasets.





Accuracy of Training Set Accuracy of Testing Set

Figure 2. Accuracies of different decision stumps for both training and testing datasets

Table 3 shows the accuracy rates of decision stump of feature 13 on training and testing datasets. As expected the accuracy rate of the decision stump on the training data is higher than that on testing dataset.

	Training Data	Testing Data
Accuracy Rate	0.7250	0.6150

Table 3. Accuracy Rate of Decision Stump of Feature 13.

Discussion

Since feature 13 has the highest information gain, this feature is selected for the decision stump. Feature 13 also provides the highest accuracy rates. In case of training data set, it is more or less expected since the nature of information gain algorithm is associated with the concept of accuracy, and since we applied information gain algorithm on the training dataset, we could expect this feature to also lead to highest accuracy. On the other hand, regarding the testing dataset, it was neither guaranteed nor expected for this feature to lead to highest accuracy rate. This phenomenon means that feature 13 has highly discriminative characteristics on this classification problem which lead to the highest accuracy on the testing data set, as it did on the training dataset too.

Part II: Bootstrap Aggregating, Bagging

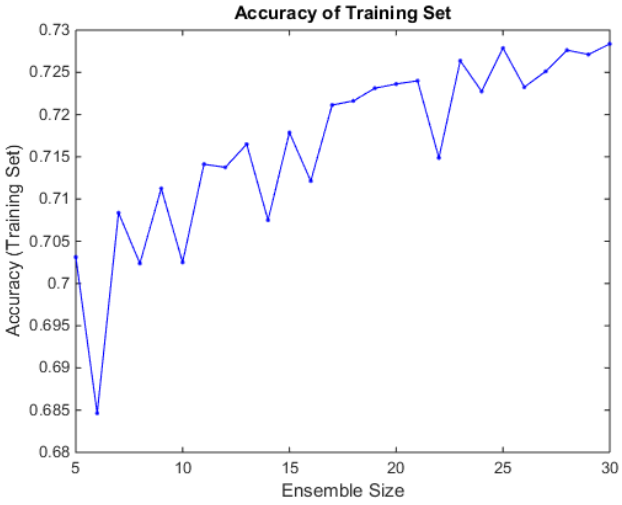
Methodology

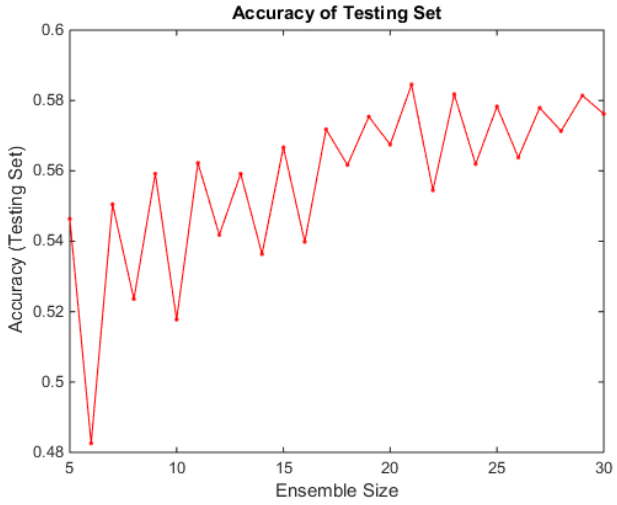
To address the problem of weak classifier, decision stump in this case, I implemented bagged decision stump with the following approach. The accuracy rates also has been analyzed over different ensemble sizes. Because of stochastic nature of bagging approach, for each ensemble size, the reported accuracy rate of the predictions is the average of 100 random runs to increase the robustness of the results.

1. Create T (ensemble size) bootstrap samples of the training dataset.
    - Each sample has the size of training dataset, 80 instances in this case
    - Each sample comes from random sampling with replacement of the training set
2. Learn the best decision stump for all the samples generate in previous step
3. Predict the outcome of the training set and testing set using majority vote on the learned hypotheses of previous step
4. Calculate the accuracy rate for both training and testing set
5. Do the first 4 steps for 100 times
6. Average the 100 accuracy rates calculated

Results

Figure 3 shows the accuracy of the algorithm for different ensemble sizes for both training and testing dataset.

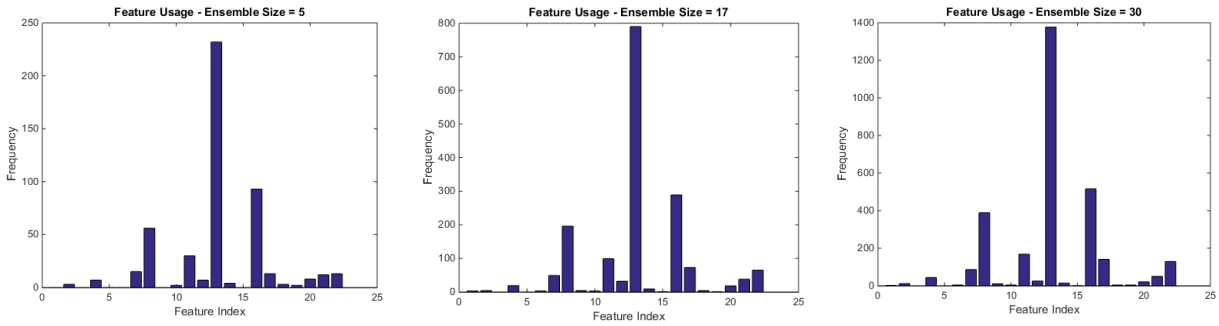




Accuracy of Training Set Accuracy of Testing Set

Figure 3. Prediction Accuracies versus Ensemble Size - Bagging

It is also important to take a look at the features which are used to make decision stumps. Figure 4 shows the histogram of the features used for making decision stumps with ensemble sizes of 5, 15, and 30.



Feature Usage for Ensemble size of 5 Feature Usage for Ensemble size of 17 Feature Usage for Ensemble size of 30

Figure 4. The Feature Usage for different ensemble sizes

Discussion

As expected, bagging, in this case, did not greatly improve the performance, especially for low ensemble sizes. The maximum accuracy rate observed with ensemble sizes up to 30, was 0.728 for training dataset, and 0.585 for testing data set. However, as you can see in the graphs, there is a general increasing, and therefore, it is expected that the algorithm outperforms the weak learner in higher ensemble sizes.

It should also be noted that it improves the training dataset accuracy more because of the fact that training dataset is the foundation that the algorithm is working with, and in general it is mostly expected to work better with training dataset, rather than testing data set.

Looking at the histogram of the features used in making decision stumps, we can state that, the relative use of the features are fairly constant over the ensemble sizes. This means that predictions, which are related to the relative number of different decision stumps, slightly changes as the ensemble size increases. As we see in the histograms, features 13, 8, and 16 are most used ones, and this pattern can be found in all three sample feature usage above. One implication of this phenomenon is that most ensemble members are similar.

Therefore, since this learning algorithm is fairly stable, and as it is known that bagging approach works better with the unstable base learners, it is not expected bagging to have noticeable improvement on the performance. Work with high variance and low bias. Works with the unstable which ours is not.

Part III: Boosting, AdaBoost

Methodology

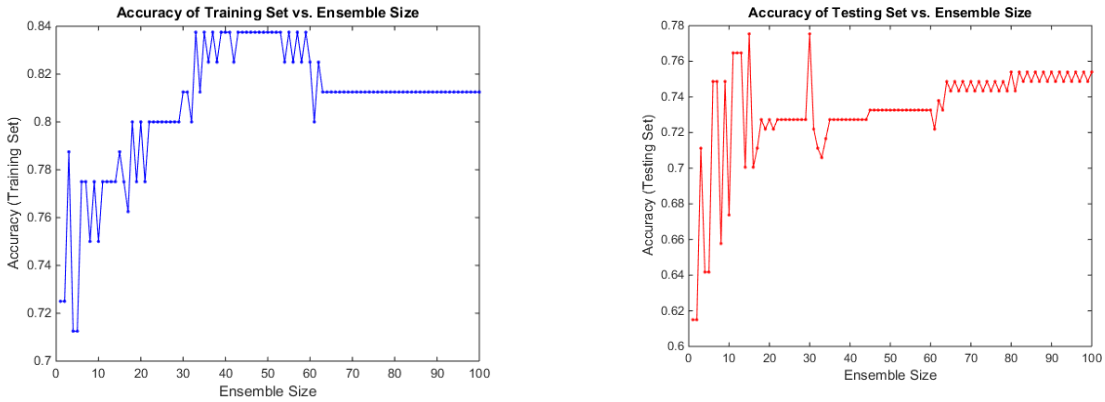
Unlike the Begging approach which individual classifiers were independently learned, boosting approach takes the performance of the previous classifier into consideration. In other words, the new classifier is more biased towards the instances that were incorrectly classified before. It utilizes a weighting system to address the importance of the observations. The approach I took can be summarized as following:

1. Uniformly initialize the weights throughout the training observations
   - $w_i = \frac{1}{\#\ of\ instances\ in\ training\ dataset}$
2. Learn the decision stump based on the training instances and their weights
3. Find the error rate of classifications, ϵ_l
4. Calculate α
   - $\alpha_l = \frac{1}{2}\ln\frac{1-\epsilon_l}{\epsilon_l}$
5. Predict the class for both training and testing dataset
6. Update the weights
   - $D_{l+1}(i) = D_l(i) * \begin{cases} e^{\alpha_l} & h_l(x_i) \neq y_i \\ e^{-\alpha_l} & h_l(x_i) = y_i \end{cases}$
7. Normalize the weights in a way that the sum of the weights equals to one
8. Analyze the performance of the classifier
   - Predict the outcome for training and testing dataset, using the weighted vote on all the hypotheses
   - Calculate the accuracy based on the final weighted predictions
9. Go to step 2

Following these steps, the results of the AdaBoost approach are presented in the next section.

Results

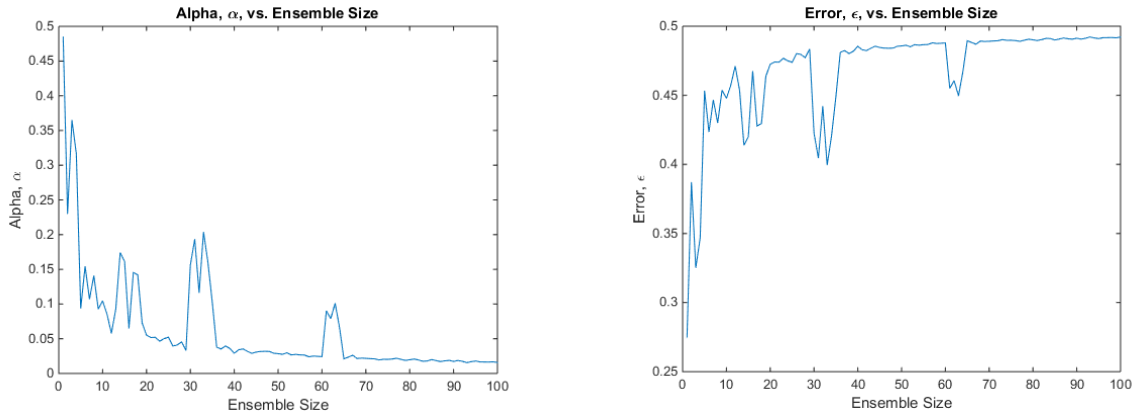
Figure 5 shows the accuracy rates of the AdaBoost approach versus the ensemble sizes.



Accuracy of Training Set Accuracy of Testing Set

Figure 5. Prediction Accuracies versus Ensemble Size – AdaBoost

As shown in the graph, AdaBoost approach greatly improves the performance of the weak learner. In other words, the accuracy rate has been increased by almost %12 for training dataset and %13 for testing dataset. Now, let's take a look at the trend of α and ϵ as the ensemble size increase. Figure 6 shows the variation of these parameters.



Alpha vs. Ensemble Size Epsilon vs. Ensemble Size

Figure 6. The variation of alpha and epsilon with respect to ensemble size

As you can see value of alpha decreases over ensemble size which shows the convergence of the classifier. It basically means that the new hypothesis does not have a great impact on the final predictions. In addition, the error rate also stabilizes around 0.5 which means that the newly learned classifiers are acting close to random classifier, and there is nothing much more to learn, and thus, the performance of the weighted combined classifier remains unchanged.

Besides, let's take a look at the features selected in each stage of the algorithm to build a decision stump. Figure 7 shows the index of the feature selected in each iteration.

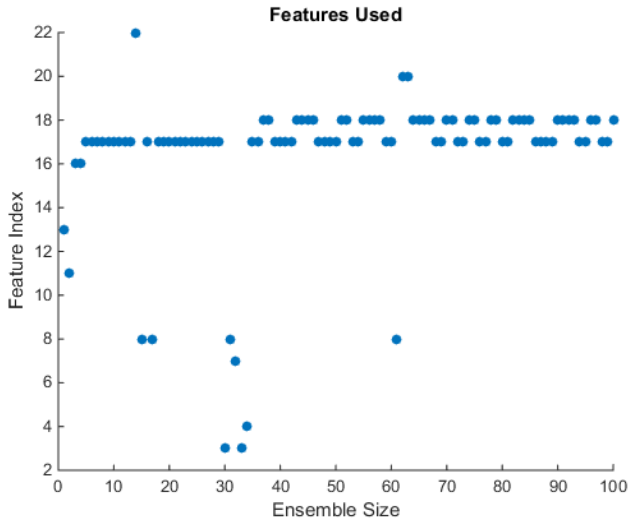


Figure 7. Features Used in each iteration of AdaBoost

As you can see the algorithm starts with feature 13 to make the decision stump, since at the beginning the weights are uniformly distributed. Then, based on the incorrectly classified instances, and the new weights, the algorithm moves to the other features. At the end, the algorithm oscillates between feature 17 and 18, and the performance of the whole classifier remains almost constant.

It is also worth mentioning that the features that this approach is investing one, are mostly the features that have high information gain (refer to figure 1).

Discussion

One of the highlighted results of this section is that AdaBoost improves the performance of the weak classifier in a very better way than nagging does. This is mostly because of the fact that bagging, unlike boosting, does not work well with the classifiers that are stable. It exponentially increases the accuracy of the training dataset, as shown in figure 5. In total, and with 100 iterations, AdaBoost increased the accuracy of both testing and training sets by almost %12, and it is expected to increase the accuracy rate of the training dataset to %100.

This approach not only leads to higher accuracy rates, but is also very faster than bagging for this particular case.

Looking at Figure 6, the values of alpha and epsilon follow a logical pattern. The error rate of the newly learned classifier increases, although the error rate of the combined weighted classifier decreases, and stabilized around 0.5, which implies that the new classifiers are behaving almost randomly, and improving trend is stopped. The same implication can be drawn from values of alpha when they approach to zero.

Another interesting point is the sequence of the features that are used in AdaBoost approach. Figure 7 shows that the algorithm starts with feature 13, and ends oscillating between features 17 and 18.

All in all, expectedly, AdaBoost performs faster and more accurately than Bagging does.

Notes on running the code:

1. Main code is “Main.m,” and the others are functions.
2. Run the “Loading Data” part in the main code first
3. Run the code associated to each part of the assignment separately to get the results and graphs. The “Main.m” code is categorized in 4 parts.
    a. Loading Data
    b. Decision Stump
    c. Bagging
    d. Boosting