Udit Mehrotra
Net Id : umehrot2

Assignment 4 - Report

Classification Algorithms Used

The classification methods used are:

- Naïve Bayes
- Adaboost with Naïve Bayes

Naïve Bayes:

It performs probabilistic prediction, i.e., predicts class membership probabilities, for each tuple required to be classified. Is foundation is based on the Bayes theorem of probability:

$$P(H|X) = \frac{P(X|H) * P(H)}{P(X)}$$

Where X -> the data sample whose class label is unknown
H -> hypothesis that X belongs to a class C

Here is the working of the classifier:

1.) Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector X = (x1, x2,…, xn).

2.) Suppose there are m classes C1, C2, …, Cm.

3.) For classifying a tuple X in test set, we need to find the maximum values for the probability P(Ci | X) among all the classes, using the formula:

$$P(Ci|X) = \frac{P(X|Ci) * P(Ci)}{P(X)}$$

Where P(X | Ci), P(Ci) are calculated from training data.

4.) Here P(X) is constant for all classes, therefore it is ignored. Hence formula becomes:

$$P(Ci|X) = P(X|Ci) * P(Ci)$$

5.) Another assumption that is made in calculating P(X | Ci) is that all the attributes are conditionally independent. Therefore P(X | Ci) can be computed using:

$$P(X|Ci) = \prod_{k=1}^{n} P(x_k \mid Ci)$$

Implementation:

1.) Load the test and train data set into the memory, as lost of tuples, where each tuple is represented as a list of <attribute>:<value> pairs. For this step

we also need to find the maximum number of attributes by reading both training and test files.

2.) Fetch the unique values associated with each attribute in both training and test files. This is required to apply Laplacian correction.

3.) Built the naïve bayes classifier, on training data by calculating the frequency of occurrence of each <attribute>:<value>:<class> pairs.

4.) Predict using the classifier on the test data set. Apply Laplacian for smoothing the attributes by adding 1 to the numerator, and number of unique values (for the attribute) to the denominator while calculating individual probabilities.

Adaboost:

This is an ensemble classification method, which is used to predict on the test data using an ensemble set of classifiers formed, to increase the accuracy of the classifier. Each classifiers prediction is given weightage based on its accuracy.

How boosting works?

1.) Weights are assigned to each training tuple in dataset D.

2.) A series of k classifier is learned iteratively by using different datasets sampled with replacement from the training set D.

3.) After a classifier M_i is learnt, it is used to predict on training set, and recompute the weights of the training tuples such that the subsequent classifier M_{i+1} pays more attention to training tuples that were misclassified by the earlier classifier M_i.

4.) The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy (obtained on training set).

Implementation:

These are the steps that were followed

1.) Initialize the weight of each tuple in D to 1/d, where d, is the number of tuples in a data set.

2.) for i = 1 to k do the following:

- Sample D with replacement according to tuple weights to obtain D_i. This was done using probabilistic sampling, depending upon the weights, to obtain sample having same number of tuples as in the training data set.

- Build classifier on the sampled data D_i, using the classification model. In our case, the Naïve Bayes model used in the first part was used to build the model.

- Predict using the built classifier on the original training data set D, and store the predicted labels.

- Calculate the error incurred by the classifier on training data, using the following formula:

$$\textbf{Error}(M_i) = \sum_{i=1}^{d} w_i * \textbf{err}(X_i)$$

Where $err(X_i) = 1$, if tuple is misclassified else 0

w_i is the weight of the corresponding tuple

- If $Error(M_i) > 0.5$, skip the current run.

- For each tuple in D, which is correctly classified, multiply the weight of tuple by $Error(M_i)/(1-Error(M_i))$. This process is carried out to reduce the weights of correctly classified tuples and hence providing a way to pick misclassified tuples while sampling in the next iteration.

- Normalize the weight of each tuple – this step is performed so that the whole weight is normalized and the sum of new weights of all the tuples will be equal to 1.

- Calculate and store the classifier weight using the formula:

$$w_i = \log \frac{1 - Error(M_i)}{Error(M_i)}$$

- Predict using the classifier on test data, and store the predicted labels.

3.) Apply ensemble classifiers built in the previous step.

- Initialize weight of each class to 0.

- for each tuple in the test data set, do the following:

- Get the predicted class of the tuple using all the k classifiers built in the previous step, and total the weights of the classifiers predicting class +1 and class -1.

- The label assigned is the label for which sum of classifier weights is more.

Udit Mehrotra
Net Id : umehrot2

Model Evaluation Methods

NAÏVE BAYES

Naïve Bayes led data:

Measure	led.train	led.test
Accuracy	0.8414	0.8369
Error Rate	0.1586	0.1631
Sensitivity	0.6254	0.5897
Specificity	0.9365	0.9476
Precision	0.8126	0.8347
F-1 score	0.7068	0.6912
F-b 0.5 score	0.7667	0.7707
F-b 2 score	0.6556	0.6265

Output Values:
Training: 399 239 92 1357
Test: 207 144 41 742

Naïve Bayes breast_cancer data:

Measure	breast_cancer.train	breast_cancer.test
Accuracy	0.7333	0.7358
Error Rate	0.2667	0.2642
Sensitivity	0.4821	0.4483
Specificity	0.8468	0.8442
Precision	0.587	0.52
F-1 score	0.5294	0.4815
F-b 0.5 score	0.5625	0.5039
F-b 2 score	0.5	0.461

Output Values:
Training: 27 29 19 105
Test: 13 16 12 65

Naïve Bayes poker data:

Measure	poker.train	poker.test
Accuracy	0.7214	0.6637
Error Rate	0.2786	0.3363
Sensitivity	0.9933	0.976
Specificity	0.0306	0.0091
Precision	0.7225	0.6737
F-1 score	0.8365	0.7972
F-b 0.5 score	0.7642	0.7182
F-b 2 score	0.924	0.8956

Udit Mehrotra
Net Id : umehrot2

Output Values:
Training: 742 5 285 9
Test: 448 11 217 2

Naïve Bayes adult data:

Measure	adult.train	adult.test
Accuracy	0.7975	0.7935
Error Rate	0.2025	0.2065
Sensitivity	0.8177	0.7956
Specificity	0.7909	0.7929
Precision	0.5608	0.5489
F-1 score	0.6653	0.6496
F-b 0.5 score	0.5984	0.5852
F-b 2 score	0.7491	0.73

Output Values:
Training: 323 72 253 957
Test: 5924 1522 4869 18641

ADABOOST

Note: The values listed here are for the output where an increase in accuracy was encountered for the test data.

Adaboost led data:

Measure	led.train	led.test
Accuracy	0.8452	0.8571
Error Rate	0.1548	0.1429
Sensitivity	0.6771	0.6838
Specificity	0.9193	0.9349
Precision	0.7869	0.8247
F-1 score	0.7279	0.7477
F-b 0.5 score	0.7622	0.7921
F-b 2 score	0.6965	0.708

Output Values:
Training: 432 206 117 1332
Test: 240 111 51 732

Adaboost breast_cancer data:

Measure	breast_cancer.train	breast_cancer.test
Accuracy	0.7556	0.7453
Error Rate	0.2444	0.2547

Udit Mehrotra
Net Id : umehrot2

Sensitivity	0.5536	0.5172
Specificity	0.8468	0.8312
Precision	0.62	0.5357
F-1 score	0.5849	0.5263
F-b 0.5 score	0.6055	0.5319
F-b 2 score	0.5657	0.5208

Output Values:
Training: 31 25 19 105
Test: 15 14 13 64

Adaboost poker data:

Measure	**poker.train**	**poker.test**
Accuracy	0.7157	0.6711
Error Rate	0.2843	0.3289
Sensitivity	0.9545	0.9499
Specificity	0.1088	0.0868
Precision	0.7313	0.6855
F-1 score	0.8281	0.7963
F-b 0.5 score	0.7672	0.7259
F-b 2 score	0.8996	0.8819

Output Values:
Training: 713 34 262 32
Test: 436 23 200 19

Adaboost adult data:

Measure	**adult.train**	**adult.test**
Accuracy	0.8393	0.8313
Error Rate	0.1607	0.1687
Sensitivity	0.5848	0.5912
Specificity	0.9223	0.9073
Precision	0.7108	0.6689
F-1 score	0.6417	0.6276
F-b 0.5 score	0.6814	0.6518
F-b 2 score	0.6063	0.6053

Output Values:
Training: 231 164 94 1116
Testing: 4402 3044 2179 21331

Udit Mehrotra
Net Id : umehrot2

Parameter chosen

Naïve Bayes:

- Laplace was applied to each attribute for smoothing its values, to handle cases where the frequency is zero for a particular <Attribute>:<Value>:<Class> combination. This is done to prevent the complete probability from becoming zero, because of just one conditional probability for an <Attribute>:<Value>:<Class> combination being zero.

- While calculating conditional probabilities during prediction, we add 1 to the numerator, and the count of the number of unique values associated with the attribute to the denominator.

Adaboost:

- For Adaboost, the number of iterations chosen was k = 5.

- This values was chosen because, it was giving decent accuracy for all the data sets provided, while significantly increasing the accuracy for few data sets. Also, the time taken to build the model and predict was not much. Choosing a larger value, would take much more time to build the model and predict.

Conclusion

The conclusion that can be drawn is that Adaboost can increase the accuracy of the basic classifier. As can be seen from the results, it did increase the accuracy in most of the data sets. However, the results obtained are not always consistent. Sometimes, the algorithm ends up providing lesser accuracy than the basic classifier. This might be due to the Randomness element associated with Adaboost. Depending upon the sample tuples picked in each iteration, and their associated weights the accuracy of Adaboost can vary.