A naive Bayes classifier on 1998 KDD Cup

Chris Fleizach
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
cfleizac@cs.ucsd.edu

Satoru Fukushima
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
sfukushi@cs.ucsd.edu

1 Introduction

The 1998 KDD Data cup provides a large dataset that has a number of features which can be learned to attempt to predict potential respondents to a mailing. It is our goal to show that the naive Bayes classifier may be accurate enough to successfully choose who will reply to the mailing. By using cross validation, we hope to establish a basis for the expected performance. We also analyze the space and time complexity of the classifier in order to compare with the theoretical thresholds of the naive Bayes algorithm.

2 Preprocessing

Before training and classifying, a number of pre-processing decisions had to be made. The data used in the 1998 KDD Cup was severely malformed in more than one respect. To try to reduce potential problems, we resorted to a number of heuristics aimed to maximize the ability to classify the data correctly. Firstly, samples that did not have the correct number of features were ignored. Next, data fields that were left blank were ignored in the training part completely if they were real valued features. If a missing value was in a discrete valued feature, the most common instance of that feature was substituted. All characters also had to be converted to numbers and finally syntactic consistency had to be maintained in order for correct Matlab usage.

3 Naive Bayes algorithm

3.1 Real-valued and discrete-valued features

We implemented the naive Bayes algorithm in Matlab so that it was possible to train and test data. It was able to deal with real-valued features as well as discrete-valued features. Real-valued features were discretized using ten bins with an interval that took the maximum and minimum values and divided by ten.

3.2 Time complexity

The theoretical time complexity for learning a naive Bayes classifier is O(Np), where N is the number of training examples and p is the number of features. To empirically test the implemented algorithm's time complexity, we used an artificially created data set. We varied the number of samples with a fixed number of features and the maximum size of bins, shown in Figure 1. Also, we varied the numbers of features with a fixed number of samples and the maximum size of bins, shown in Figure 2, and tested varying number of bin sizes with a fix number of samples and features, shown in Figure 3. The figures show the time is linear in respect to the number of samples, features, and the maximum size of bins.

In our algorithm, we first had an outer loop for each feature. Within this loop, we enumerated the number of samples, finding the correct bin (or discretized bin) based on whether it was a real valued or discrete feature. This led to a running time of O(Np). In our first iteration of the algorithm, the running time was O(Npq), because all values of the bins had to be searched in order to find the correct bin for the value of the specific feature, which was shown in Figure 1-3. Later, we realized that if a different data structure was used we would not need to search through all q values of each feature because an index could be used to do a fast lookup. We employed the use of a Java Hashtable through the Java Interface in Matlab in order to accomplish this task. This was able to reduce the overall time complexity to meet the theoretical bound. This same format was used in the testing of the data, which also resulted in an O(Np) running time.

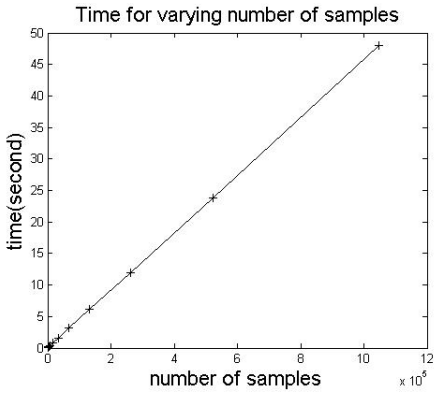

Figure 1: Time complexity regarding to the number of sample

3.3 Space complexity

The theoretical space complexity for naive Bayes algorithm is O(pqr), where p is the number of features, q is values for each feature, and r is alternative values for the class.

We were able to achieve the same theoretical space complexity, because in our implementation, we require the number of features (p) * the maximum number of bins (q) * the number of classes. Since the number of classes was two for the project, the space complexity of the implementation is O(2pq) = O(pq).

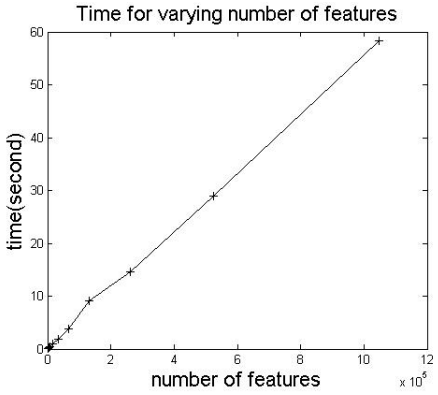

Figure 2: Time complexity regarding to the number of feature

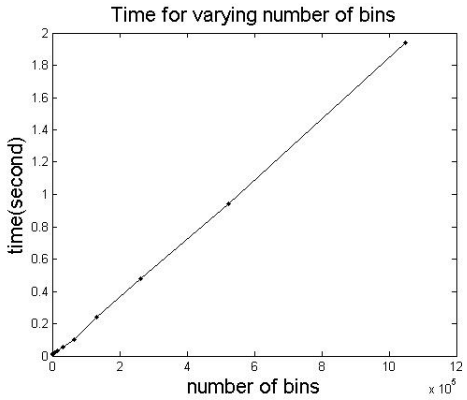

Figure 3: Time complexity regarding to the number of bins

4 Results

The success of our naive Bayes classifier was difficult to quantify. In our implementation, higher accuracy was obtained when a smaller set of training samples were used, whereas the recall increased as the number of training samples grew. Therefore, we analyzed performance with regards to both large and small sample sizes to determine the effect it would have.

For both cases, we estimated the performance by conducting cross validation on the KDD learning data set. Ten fold cross validation was used, which trained the classifier with 9/10ths of the data and tested with the remaining tenth for each separate segment. Using a tenth of the data though did not offer as high an accuracy as we had seen with even smaller training samples.The accuracy and the performance numbers were averaged and are shown in Table 1 and 3.

	Positive	Negative
True	216.0	6259.3
False	1758.1	266.6

Table 1: means of true/false - positive/negative with a large training set

	Positive	Negative
True	1673.3	58766.0
False	22391.0	2670.1

Table 2: means of true/false - positive/negative with a small training set

	tenth training set	9/10th training set
Accuracy (mean)	0.7069	0.6816
Recall (mean)	0.3767	0.4267
Precision (mean)	0.0642	0.0683
F-score (mean)	0.1098	0.1177
F-score (s.d.)	0.0047	0.0110

Table 3: Measurements on classification results

We also ran cross-validation using the opposite strategy, where ten percent of the data was used to train and ninety percent was used for testing. The results are shown in Table 2 and 3. This strategy aimed to do better in accuracy, whereas the previous strategy was able to achieve much better recall.

The dichotomy between accuracy and recall was troubling. We concluded that learning the dataset with a high degree of accuracy may not be possible. It seemed that as more data was learned, the classifier became aware of new instances that would cause an affirmative case to be detected. Thus, after learning on 80,000 samples, for example, the classifier had a large base of differing possibilities that all lead to a yes prediction. The result was an abnormally high number of false positives (as well as an increased number of true positives). This provided better recall, but lower accuracy.

On the other hand, if we only learned with a few samples, the classifier only knew what a yes prediction would look like from a limited set. Thus, most predictions were negative, and because most of the answers were negative, this provided a higher accuracy.

In this assignment, at least, it seemed more important to achieve a better recall, since we aimed to discover the people who would be most likely to respond to the mailing. The key insight being that we wished to reduce the number of false negatives. In other words, recall was more important than accuracy, because if a person was not identified as being a respondent, then a potential donation could be missed, whereas if there were false positives, it would not negatively affect donations.

5 Prediction

From our cross validation results mentioned in the previous section, we obtained a mean F-score of 0.1098 with its standard deviation of 0.0047 for a tenth training set and a mean F-score of 0.1177 with its standard deviation of 0.0110 for 9/10th training set, each on average. Therefore we decided to use a larger set in training phase. The fact that the size of the training set used in the cross validation with the 9/10th training set is similar to the size of the actual hidden test set allows us to say with a degree of certainty (≈ 0.95) that the F-score will be within 0.1177 ± 0.0110 as long as the tested set follows similar formatted patterns. In the CUP98VAL dataset, we predict that there will be 21.0% positive results and 79.0% negative results, which is obviously skewed towards being too optimistic in stating a prediction, but nevertheless manages to achieve a better F-score.

6 Conclusion

We observed a number of surprising and sometimes disheartening results. We were not able to achieve a high accuracy of predictions, most likely due to the nature of the dataset and the relative difficulty of selecting appropriate features to learn. On the other hand, we were able to learn a classifier that increased it's recall as the sample size grew. For example, if we trained on 1000 samples, our recall was around 20%, while using cross validation on the entire dataset, our recall improved to about 42%. Unfortunately, this did not extend to accuracy, which decreased as more samples were learned. Naive Bayes classifiers are powerful tools for learning data, but as we have discovered they need to be excessively coddled and fine tuned, if they are to be useful in all cases of performance.