Performance Comparison of the Automatic Data Reduction System (ADRS)

Dan Patterson^a, David Turner ^a, Arturo Concepcion ^a, and Robert Lynch ^b

^aDepartment of Computer Science, California State University, San Bernardino, CA, USA;

^bSignal Processing Branch, Naval Undersea Warfare Center, Newport, RI, USA

ABSTRACT

In this paper, real data sets from the UCI Repository are mined and quantized to reduce the dimensionality of the feature space for best classification performance. The approach utilized to mine the data is based on the Bayesian Data Reduction Algorithm (BDRA), which has been recently developed into a windows based system by California State University (see http://wiki.csci.csusb.edu/bdra/Main_Page) called the Automatic Data Reduction System (ADRS). The primary contribution of this work will be to demonstrate and compare different approaches to the feature search (e.g., forward versus backward searching), and show how performance is impacted for each data set. Additionally, the performance of the ADRS with the UCI data will be compared to an Artificial Neural Network (ANN). In this case, results are shown for the ANN both with and without the utilization of Principal Components Analysis (PCA) to reduce the dimension of the feature data. Overall, it is shown that the BDRA's performance with the UCI data is superior to that of the ANN.

Keywords: Classification, Feature search, BDRA, ADRS, ANN, PCA

1. INTRODUCTION

In this paper, the Bayesian Data Reduction Algorithm (BDRA) [1,2,3] is trained and tested on seventeen real-life data sets that are contained at the University of California at Irvine's (UCI) Repository of Machine Learning Data Bases [4]. The BDRA is a probabilistic classifier that employs feature selection techniques (i.e., utilizing both forward and backward sequential searching [5,6]) to improve classification performance. The BDRA requires all data to be discrete so any continuous valued features must first be discretized. To do this a method of percentiles is used to discretize each feature. For example, to obtain ternary valued features a set of thresholds are computed for each feature so that its values are placed in three discrete cells that are equally sized, with respect to numbers of data. Once all features are discretized, the BDRA then automatically reduces each feature based on the probability of error conditioned on the training data, and finds the quantization of the features corresponding to best performance. Typically, the initial number of discrete levels used to quantize each feature affects this final "best" quantization point. The BDRA tends to work best in cases where there is insufficient data to support feature complexity, due to the curse of dimensionality. In other words, the BDRA is a worthy contender in cases in which a reliable decision boundary separating the classes can be reliably determined using a selected feature subset, but in which this is infeasible when all features are used.

All performance results for this paper were generated using the Automatic Data Reduction System (ADRS)¹, which has recently been developed at California State University into a windows based system implementation of the BDRA (see http://wiki.csci.csusb.edu/bdra/Main_Page). In this work, the BDRA is also compared to an Artificial Neural Network (ANN) classifier, where testing errors were computed for four different configurations of the ANN. With that, a Principal Components Analysis (PCA) of the data was performed to determine if reducing the dimension of the feature space, prior to training the ANN, would improve its overall performance [7]. Results are demonstrated by comparing the probability of error, for each method, in classifying the UCI data sets.

¹ Details on ADRS can be found in the Appendix, and it is also referred to as the Bayesian Data Reduction System (BDRS).

2. PROBABILITY OF ERROR

The primary metric used for determining the performance of each classifier is the probability of error, P(e), which is estimated using the hold-out ten percent method of training and testing. In this method the data are partitioned into ten disjoint sets for testing with the remaining data (the remaining nine sets not used for testing) used for training each classifier. Data stratification is utilized to insure that the training and test sets have roughly the same class proportions. Final tabulated P(e) values are obtained by averaging over error probabilities computed with the ten disjoint partitions. The error probability is computed by counting the number of test feature vectors, conditioned on each class, k, that caused wrong decisions (i.e., a class other than k was decided). Then, the wrong decision count for class k is normalized by its respective total number of test feature vectors. This is repeated for all classes so that the desired error probability is obtained by summing the results for each class (i.e., given an assumed equal prior probability of each class). Finally, the hold-out ten percent method was applied six times so that final performance results are based on an average of sixty trials.

3. MISSSING FEATURE VALUES

A number of the data sets found at the UCI Repository contain missing feature values. Therefore, before discussing results of applying the BDRS to the real-life data a few comments are made about modifications that were made in order to account for the possibility that the data of either class contains missing feature information. As it turns out, slightly more than one third of the UCI data sets used in this work have missing features. Basically, by missing features it is meant that all feature vectors of the k^{th} class are assumed to be made up of either or both of two observation types: features which are represented by specific values, and missing features which have no values. For example, with three binary features a possible feature vector that is missing a single feature might appear as (1,1,x) where x represents the missing value. In this case x can have the value of 0 or 1 so that this feature vector has a cardinality of two. In general, all missing features are assumed to appear according to an unknown probability distribution.

With respect to missing features an often-used approach is to "fill-in" missing values by estimates obtained from all known feature values (e.g., such as the sample mean) [7]. In fact, this method is used for the ANN classifier employed here (if the feature is discrete the median is used instead). However, to model missing feature information in BDRA the Mean-Field algorithm, [1], is utilized that adapts to the missing feature information by estimating it from the available training data. However, the method for accomplishing this is not based on the optimal approach to dealing with this problem. In fact, the optimal approach turns out to be impractical to implement due to computational considerations.

4. RESULTS

Table I lists the characteristics of each UCI data set used, classification results for the BDRA, the ANN, and the ANN using PCA to reduce the dimension of the data. Listed in the columns of this table, from left to right, are the Data set name, the total number of classes, the total number of data points, the number of continuous valued features, the feature search direction used in the BDRA, the number of discrete levels used to initially quantize continuous valued features in the BDRA, the estimated probability of error on test samples for the ANN, the estimated probability of error on test samples for the BDRA, the number of features selected by the BDRA, the number of dimensions extracted by PCA, and the estimated probability of error on test samples for the ANN using PCA.

TABLE I

Data characteristics and classification performance. The columns have the following meaning: Data set name (*UCI Data Set*); Total number of classes (*number of class labels*); Total number of data points (*number of samples*); Total number of features (*number of features*); Number of continuous valued features (*number of cont. features*); Best feature search direction used in the BDRA (*search direction*); Number of discrete levels used to initially quantize continuous valued features (quant. of cont. features); Estimated probability of error on test samples for the ANN (*testing error ANN*); Estimated probability of error on test samples for the BDRA (*testing error BDRA*); Number of features selected by the BDRA (*number of selected features*); Number of dimensions extracted by PCA (*number of extracted dim*); Estimated probability of error on test samples for the ANN using PCA (testing error ANN+PCA).

	number			number		quant.			number	number	testing
UCI	of	number	number	of		of	testing	testing	of	of	error
Data	class	of	of	cont.	search	cont.	error	error	selected	extracted	ANN +
Set	labels	samples	features	features	direction	features	ANN	BDRA	features	dim	PCA
annealing	5	798	38	6	backward	3	0.158	0.133	2	2	0.250
balance scale	3	625	4	0	backward	NA	0.075	0.219	4	4	0.159
coagulopathies	2	32	59	27	backward	10	0.122	0.109	3	4	0.133
credit	2	690	15	6	forward	3	0.178	0.143	3	2	0.368
ecoli	8	336	7	7	backward	3	0.310	0.214	4	6	0.212
glass	6	214	9	9	forward	3	0.485	0.439	6	6	0.476
hepatitis	2	155	19	6	forward	3	0.319	0.181	2	4	0.313
ionosphere	2	351	34	32	forward	10	0.129	0.149	6	30	0.246
iris	4	150	4	4	backward	3	0.087	0.063	2	3	0.027
lenses	3	24	4	0	forward	NA	0.425	0.200	2	4	0.300
liver disorder	2	345	6	6	backward	3	0.346	0.365	6	5	0.326
lung cancer	3	32	56	0	forward	NA	0.578	0.531	56	24	0.400
diabetes	2	768	8	8	forward	3	0.296	0.236	2	5	0.281
schizophrenia	2	51	37	33	forward	3	0.217	0.137	3	1	0.280
tic tac toe	2	958	9	0	backward	NA	0.126	0.265	6	9	0.213
wine	3	178	13	13	backward	10	0.034	0.094	4	1	0.222
yeast	10	1484	8	8	backward	10	0.563	0.544	4	8	0.496

Table I demonstrates overall classification results for the BDRA and the ANN using the UCI data. It can be seen that overall performance of the BDRA is superior to the ANN in that the BDRA achieved a lower probability of error with 12 out of the 17 UCI data sets used. Thus, to determine the impact of dimensionality reduction on the data (which is a very important aspect in the BDRA's performance), PCA was applied to the data before training the ANN. In this case, it can be seen can see that overall PCA had a minimal impact on the performance of the ANN. Specifically, comparing the results of ANN to those of ANN+PCA reveals that PCA improved performance of the ANN in some cases, while in others it degraded performance. Further, in comparing the BDRA to the ANN+PCA it can be seen that the BDRA stills has a lower probability of error with a majority (10 out of 17) of the UCI data sets. Note, also that this is significant because training time of the BDRA in the forward search mode can be thousands of time samples faster than that of an ANN.

As a final note to the BDRA's performance, notice that the algorithm appears to favor a backward feature search direction about as often as it does a forward one (9 backward to 8 forward). Further, for data sets with continuous valued features (13 out of 17 data sets), of these cases most of the time it was found that the BDRA prefers to start training with 3 discrete levels per feature as opposed to 10 (9 out of 13 times). The reason that this occurs is due to the inherent

² NA means that no continuous valued features are found with the data.

suboptimality of the BDRA's dimensionality reduction method. In other words, optimal data quantization by the BDRA is impossible due to computational complexity (except only in very low dimensional cases), so that it is necessary to reduce the data in a sequential manner (i.e., either in a forward or backward search mode). In a suboptimal sequential searching mode, the likelihood is increased that that training stops on a local, as opposed to a global, minimum value of the probability of error space. Thus, when classifying real data, performance of the BDRA tends not to depend on any specific search mode, or initial number of discrete levels per dimension.³ This makes it necessary to try different search modes, and initial quantization levels, to determine a best overall configuration by trial and error. Fortunately, this can easily be accomplished in the BDRA with straightforward programming methods.

5. CONCLUSION

In this paper, performance results have been demonstrated using ADRS and real data sets from the UCI Repository. The approach utilized is based on the Bayesian Data Reduction Algorithm (BDRA), which has been recently developed into a Windows based system by California State University (see http://wiki.csci.csusb.edu/bdra/Main_Page) called the Automatic Data Reduction System (ADRS). The primary contribution of this work was to demonstrate and compare different approaches to the feature search (e.g., forward versus backward searching), and show how performance is impacted for each data set. Additionally, the performance of the ADRS with the UCI data was compared to an Artificial Neural Network (ANN). In this case, results were shown for the ANN, both with and without the utilization of Principal Components Analysis (PCA), to reduce the dimension of the feature data. Overall, it was shown that the BDRA's performance with the UCI data was superior to that of the ANN, both with and without PCA.

REFERENCES

- [1] R. S. Lynch, Jr. and P. K. Willett, "An Algorithmic Approach to Mining Unknown Clusters in Training Data," Proceedings of the SPIE Symposium on Defense and Security, Orlando, FL, April 2006.
- [2] R. S. Lynch, Jr. and P. K. Willett, "Bayesian Classification and Feature Reduction Using Uniform Dirichlet Priors," IEEE Transactions on Systems, Man, and Cybernetics, vol. 33, no. 3, June 2003.
- [3] R. S. Lynch, Jr. and P. K. Willett, "Performance Considerations for a Combined Information Classification Test Using Dirichlet Priors," *IEEE Transactions on Signal Processing*, vol. 47, no. 6, June 1999, pp. 1711-1715.
- [4] C. J. Merz and P. M. Murphy, "UCI Repository of Machine Learning Databases," *Department of Information and Computer Sciences*, University of California, Irvine, CA, 1996.
- [5] J. Kittler, "Feature Set Search Algorithms," *Pattern Recognition and Signal Processing*, C. H. Chen, ed., Sijthoff and Noordhoff, Alphen aan den Rijn, The Netherlands, 1978, pp. 41-60.
- [6] A. Jain and D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Size Performance,"
- IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 2, February 1997, pp. 153-158.
- [7] C. M. Bishop, Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995.

³ Although this has not been formally proved for the BDRA, the literature on feature reduction indicates that a backward feature search mode would perform best under optimal conditions. This is true because when reducing the data the backward search mode pays more respect to the statistical dependency amongst the dimensions. Thus, it has a better chance of finding a more accurate model for the data.

APPENDIX: OVERVIEW OF THE AUTOMATIC DATA REDUCTION SYSTEM⁴

The folowing is an overview of the main components of the Bayesain Data Reduction System developed at California State University, at San Bernardino. The BDRA is used to generate a classification model from training data (a set of classified observations). The resulting classification model is applied to new observations to predict their class memberships. To enable the practical use of the BDRA, our BDRA program provides the following three modes of operation.

- evaluation mode
- training mode
- classify mode

Evaluation mode

When evaluation mode is specified, the program generates a report that describes how well the BDRA classifies test data given specific parameters that define the operation of the BDRA.

The program takes the following inputs.

- a set of classified observations
- parameters defining the operation of the algorithm (Complete Commnad Line usage information can be found at http://wiki.csci.csusb.edu/bdra/Command_line_usage.)

For an example of command line usage within ADRS, the simplest form of the BDRA program invocation is illustrated by the following example.

```
java bdra.Main iris.txt
```

The above command assumes that the dependent jar files are in the classpath. If you run the above from inside a bat file under Windows, the classpath could be specified as follows.

```
set CLASSPATH=itext-2.0.1.jar;bdra.jar
java bdra.Main iris.txt
```

If working with a typical large data set, the chances are good that one needs to specify a maximum memory allocation that is larger than the default. The following bat file shows how this is done.

set CLASSPATH=itext-2.0.1.jar;bdra.jar

⁴ The Automatic Data Reduction System (BDRS) is Copyright 2007 California State University, San Bernardino. The BDRS is also patented by NUWCDIVNPT, U.S. Pat. Nos. 5,999,893; 6,397,200; and 6,789,070. The U.S. Government has a copyright license in this work pursuant to a Cooperative Research and Development Agreement with NUWCDIVNPT. The current state of the software is beta-test version 0.1; it is not intended for commercial use.

java -Xms1800m -Xmx1800m bdra.Main iris.txt

In this case, the commands are telling the Java virtual machine to make an initial allocation of 1800 MB of memory and to make the maximum amount equal to 1800 MB as well. This is a typical setting that was used on a computer with 2048 MB of memory (2 GB). Further, if working on linux/unix, then the above run can be accomplished through a shell script with the following contents:

```
export CLASSPATH=itext-2.0.1.jar:bdra.jar
java -Xms1800m -Xmx1800m bdra.Main iris.txt
```

A table of all command line settings can be found at http://wiki.csci.csusb.edu/bdra/Command line usage.

When the window size setting is set to 1, the program iterates through the set of classified observations given to it as input. For each of these observations, it runs the BDRA algorithm on the other observations in order to generate a classifier. It then applies this classifier to the given observation to predict its class membership. If it predicts incorrectly, the program increments its count of failures. This process can be expressed in the following pseudo code.

```
observations = set of classified observations provided as input
ob = an individual observation
ob.class = the class in which ob is a member
obs = set of observations with ob removed
```

```
for each ob in observation
obs = observations - ob
classifier = BDRA(obs)
class = classifier.classify(ob)
if ob.class != class
errors = errors + 1
```

The percentage of incorrectly classified observations is called the *test error*. The *training error* that is also reported is an internal estimate the BDRA makes for the classification error.

Each run of the program in evaluation mode generates a PDF document that summarizes the results of the run. When run in evaluation mode, the program generates a PDF document that summarizes how well the algorithm classifies test data. This page defines the contents of this PDF document.

If the user specifies a report name on the command line, the program will use this name for the generated pdf file. For example, consider the case that the user runs the program with the following command.

```
java -jar BDRA --report-name iris iris.bdra
```

In the above case, the program generates a file with name *iris.pdf*. If the report-name argument is omitted, the program generates a file with the name *report.pdf*. See <u>Command line usage</u> for more details.

The report contains the following sections, which are described below.

1) Program inputs

This section provides the following summary descriptions of the input data and the selected parameters that control the operation of the BDRA algorithm.

- Name of data set
- Time and date of run
- Number of categorical and continuous features
- Number of observations
- Number of observations with missing values
- Number of missing values
- The number of continuous bins
- The missing value approach used
- The search direction
- Search technique
- Probe level
- Window size

2) Global Measures

When window size is 1, this section includes the following.

- Number of runs
- Average time per run and standard deviation
- Average training error and standard deviation
- Average testing error and standard deviation

When window size is greater than 1, this section includes the following.

- Time to complete
- Training error
- Testing error

3) Feature Analysis

In this section, rows represent features, and the columns represent metrics.

When the command line window size is 1, this section includes the following columns.

- Feature name
- Whether feature is continuous or categorical
- Percentage of times eliminated
- Per feature quantizational complexity (how many times each feature was reduced to 1 bin, 2 bins, etc.)

When the window size is greater than 1, this section includes the following columns.

- Feature name
- Whether feature is continuous or categorical
- Was the feature eliminated?
- Bin definitions (for categorical features, show category groups; for continuous features, show boundaries for continuous values)

Retrieved from "http://wiki.csci.csusb.edu/bdra/Program outputs in evaluation mode"

Training mode

When training mode is specified, the program generates a classifier file that can be used to classify observations. The classifier file is used as input for the program when it runs in classify mode.

In evaluation mode, the program also generates a report in PDF format, which describes characteristics of the generated classifier. Go to http://wiki.csci.csusb.edu/bdra/Program_outputs_in_training_mode for a description of the contents of this PDF document.

Classify mode

When classify mode is specified, the program uses the classifier file to classify observations.

Although this mode can be used to classify classified observations, it is normally used as a decision support tool to classify unclassified observations. Go to http://wiki.csci.csusb.edu/bdra/Program outputs in classify mode for a description of the contents of the PDF document for this mode.

Flowchart

The following flowchart illustrates how the program is used in practice.

Although a report is only shown as the final output of the process, reports are generated each time the program is run. These have been omitted to simplify the diagram.