

Hybrid Voting Algorithms Using Selected Models for Categorical Data

E. Graubins and D. Grossman
Department of Computer Science
Illinois Institute of Technology
Chicago Illinois 60616
{eug.grossman}@ir.iit.edu

ABSTRACT

Although voting methods are a viable way to improve classification algorithm performance, these have usually been applied to complete training datasets. We propose a new voting methodology which is based on the success of each individual classifier as it is applied to particular classes in a training dataset. We test some specific variations on this theme and have found as much as a 12.8% improvement in effectiveness over current voting algorithms.

1. INTRODUCTION

Improving model effectiveness is a key goal of classification algorithms. Voting algorithms, by combining results from different classifiers, may outperform individual classifiers. Such algorithms have been shown to improve overall effectiveness, but current algorithms focus only on how an algorithm performs against the entire dataset. We have seen enough variance of algorithms across classes (e.g.; one algorithm might be better at selecting high-risk loan applications while another might be better at selecting low-risk loan applicants). We have found that, on some datasets, effectiveness can be improved by as much as 12.8% by using a hybrid voting algorithm instead of traditional voting algorithms.

Section 2 provides a brief overview of prior work, Section 3 describes our hybrid voting algorithms and; Section 4 presents our results. Section 5 provides a summary and directions for future work.

2. PRIOR WORK

A motivation for using voting algorithms is given in [SWCI01, GB95, BS00, SB00, KEGSST98, LP03, WM96, BK99]. Several algorithms were tested over the entire dataset and, in some cases, significant variances in the effectiveness of these algorithms were observed.

2.1 Voting Algorithms

A discussion of various voting approaches is given in Leung and Parker [LP03]. Although voting is commonly thought of as majority consensus, the authors, expound that the winner in such a competition need not always be the one with the most votes. They present results that encompass a number of voting approaches including plurality, anti-plurality, plurality-elimination, Borda Count, and pairwise comparison.

Plurality voting has appeal because of its simplicity and has universal appeal because of this characteristic. The final value of

the classifier is the majority value after the output of the algorithm set is tabulated. The winner is the one with the most votes.

The anti-plurality method employs counts for the least desirable classifier value. The winning value for the classifier is the one with the least number of last place votes.

The plurality-elimination approach uses a series of iterations where the value with the least number of first place votes is eliminated from consideration. The final classifier value is the one that survives this culling process.

In the Borda Count method, a set of possible classifier values is ranked by order of the number of votes received. These values are assigned a point value according to the position each occupies in the list, with the last position receiving a 0, the next to last receiving a 1, and so on. The candidate with the largest point value is the winner.

In the pairwise comparison method, each candidate value is matched head-to-head against each other candidate. The winner of each comparison gets one point for a win. For a tie, each candidate receives half a point. The candidate with the highest point value is the winner.

The data for testing consisted of 39 UCI data sets. The algorithm utilized was the Tree package [BFOS84] implemented in the "R" language [Gen97]. The compiled results show that voting may improve overall performance but additional research should be conducted. Several ancillary observations, however, have probative value and are relevant to the research proposed further this paper. The first is that two-class problems are discriminant problems, which are easier to classify, and the second is that plurality voting coupled with an iterative elimination approach may be used to home in on the best result. Bauer and Kohavi [BK99] present research pertaining to the application of a number of voting methods. The study uses two mining algorithms, specifically MC4 and Naïve-Bayes and three voting algorithms, namely Bagging, AdaBoost, and Arc-x4.

Bagging and boosting meld the decisions of different algorithms to arrive at a single result. While both algorithms use the same underlying concept of voting, the way each derives its models is different. In bagging, each candidate model is given equal weight when votes are cast. In boosting, the more successful models are given additional weight, thus introducing a bias in vote casting.

The bagging algorithm implements its voting approach by creating, from one training data collection, several training

data sets. It uses the original training data, partitions it, duplicates some rows, deletes some, thus arriving at a set of data sets which are each used to create a model that represents the particular data used to train it. During classification, each model is applied to the test data and values for the classifier are collected from each of the models. These model results are used to vote for the final value of the classifier. The bagging algorithm is given by [WF00].

While bagging generates a series of models and uses their output in a democratic voting scheme, boosting weighs the models and gives preference to those which produce better results. The boosting algorithm is given by [WF00].

Among many variants on the concept of boosting, Bauer and Kohavi use Adaboost (Adaptive Boosting), which is suited specifically for classification and Arc-x4 [Bre97], another ensemble method which builds a series of experts by iteratively training and reweighing. Arc-x4 differs from Adaboost in that it uses a simpler function in weighing each expert [RH00].

A voting approach to stock market classification is also described by George T. Albanis, Roy A. Batchelor, [AB00] who use voting algorithms to merge five algorithms: Linear Discriminant Analysis, Probabilistic Neural Network, RIPPER Rule Induction, Learning Vector Quantization, and Oblique Classifier.

3. Methodology

Our approach relies on the possibility that different classifiers perform very differently on different classes inside of a dataset. Consider Table 1. This Table provides accuracy rates for the Car Evaluation Dataset from the UCI repository [UCI04]. Using five different algorithms we see a standard deviation of accuracy for class “Acceptable” of 5.46, class “Good” of 29.23, class “Unacceptable” of 1.37, class “Very Good” of 18.24. Hence there is enough deviation within a class to lead us to select particular algorithms solely based on their performance for a given class. During training, we identify a weight to apply to an algorithm for a particular class. This weight is then applied on the test dataset when an algorithm selects a particular class.

A straightforward algorithm results from this approach. We call it HybridVoting and it is given below.

Algorithm: Hybrid Voting, steps:

1. Apply all of the candidate algorithms to a given training dataset.
2. Measure their effectiveness for each class in the dataset
3. Compute an algorithm weight for each class and algorithm combination. based on the performance of the algorithm.
4. Select the highest weight algorithm
5. Remove the class from the training dataset.
6. Repeat procedure from Step 1 until two classes remain.
7. Classify the remnant in a non-voting fashion.

We compute a final confidence for that class for a given classifier in step 3. This constitutes a “vote” for a given classifier.

We now describe an iterative approach, which works the same as our HybridVoting algorithm – the core difference is that once we select an algorithm for a class in the training phase we (as with decision tree algorithms) reduce the dataset to eliminate all entries that pertain to that class. The resulting set of algorithm-class couplets is applied to a test data set. The steps comprising the new algorithm are given below:

1. Apply all of the candidate algorithms to a given training dataset.
2. Measure their effectiveness for each class in the dataset
3. Compute an algorithm weight for a given class based on the performance of the algorithm (we will describe some weights to use here shortly).
4. For the algorithm that performs best for this class, develop a rule that if this class is predicted by this algorithm it will simply be the choice used in the final result.
5. Remove the “winning” algorithm from the data set
6. Remove all training data associated with the “winning class”
7. Return to step 2 for the remaining classes and algorithms
8. Choose the class with the highest confidence based on the “votes” for each class from each of the algorithms
9. Sequentially apply each algorithm and associated pair to a test data set.

The HybridVoting algorithm is described in the Sections Algorithm 1 and Algorithm 2. It consists of two phases: the training stage where a list of algorithms and associated classes is created; and an application phase where the list is applied to a dataset and the classes are removed.

Algorithm 1, hybridTrain

procedure hybridTrain

Input Algorithms 1:n // set of n algorithms to consider
 Input Classes 1:k // list of k classes in training and tests
 Input Dataset train // training dataset
 Input Dataset test // testing data set

Results 1:k // the results vector
 TopResults 1:k // store best algorithm and class

// loop until only 2 classes are left
 while classes remaining >= 2
 // inner loop iterates through set of algorithms being
 // considered
 for A in Algorithms
 train A using train dataset
 test A using test data set
 identify best performing class C
 place A,C in Results set
 end loop
 // obtain best Algorithm for Class from Results set
 (A,C) := BEST(Results)
 remove class C from training data set

```

remove class C from test data set
    // store the best algorithm and class
place (A,C) in TopResults set
    // null the class results list
Results := null
end loop
sort TopResults set, placing best A first
Results := null
// process the last 2 remaining classes
for A in Algorithms
    train A using train dataset
    test A using test dataset
    place A in Results set
end loop
identify best performing algorithm A
place A in TopResults set
return TopResults
end procedure

```

Algorithm hybridMerge accepts a set of algorithm and associated class couplets and applies them to a dataset. The best performing algorithm-class combination is applied first so the size and complexity of the dataset is depreciated as quickly as possible.

Algorithm 2, hybridApply

```

procedure hybridMerge
Input TopResults 1:k    // top performing algorithms
Input Dataset classifyApply // target data for classification
Dataset ClassifiedData // repository for classified data

ClassifiedData := null

// loop until only 2 classes are left
// inner loop iterates through set of algorithms
// generated in training phase
for A,C in TopResults
    if class remaining count == 2
        break
    classify dataset classifyApply using A for class C
    place C classified rows into ClassifiedData
    delete class C rows from classifyApply
end loop
// classify final 2 remmaing classes using last A
classify dataset classifyApply using A
place rows from ClassifyData into ClassifiedData
return ClassifiedData
end procedure hybridApply

```

The hybrid voting system uses an observed characteristic that manifests itself during classification processes: that different classification values attain different success rates when being categorized by different algorithms. We exploit this characteristic by the iterative application of a series of classification algorithms while, at the same time, whittling down the number of rows in the data set.

3.1 Algorithm Application Illustration

Consider a specific data collection, the Car Evaluation dataset the UCI data repository [UCI04]. This data was derived from a hierarchical decision model and is used to evaluate automobiles as being unacceptable, acceptable, good, or very good.

We randomly partitioned the data into four components: 50% for training, 25% for model selection, 25% for final testing. For classification, we use a collection of modeling algorithms from the Weka package [WF00]. We used the following classifiers: j48, Weka’s variant of C4.5 [Qui93]; multi-level perceptron, neural network; Naïve Bayes; decision table; and an SVM. These are trained using the mentioned training set and are used to classify the model performance data partition. The results are shown in Table 1 and are used as the beginning of the culling process inherent to the hybrid algorithm. The Car data set has four classes: acceptable, good, unacceptable, and very good.

Table 1. UCI Car Performance Results

Algorithm	Classifier Success Rate				
	Acceptable	Good	Unacceptable	Very Good	Total
C4.5	78.8	14.3	95.6	66.7	86.8
NN	73.7	90.5	98.7	100.0	92.6
Naïve Bayes	72.7	28.6	97.0	66.7	87.0
Decision Table	71.7	42.9	98.3	66.7	88.4
Vector Machine	84.8	57.1	96.0	100.0	91.7
Standard Deviation	5.46	29.93	1.37	18.24	2.69

As shown, Neural Network and Vector Machine both achieved a 100 % success rate in identifying the “Very Good” value. The results are analysed and the best performing algorithm for any given classifier value is identified. As shown, Neural Network and Vector Machine both achieved a 100.0% success rate in identifying the “Very Good” value. There were only 33 instances if this class in the training set.

The 33 instances are now removed. The model performance selection test set and the algorithm-class couplet are then stored in the results list for use in the application phase.

The classifiers are now trained again with the reduced training set and used to classify the performance selection set. For the three remaining classes, the best accuracy on a classifier/class combination was NN-“Good”. Again, we store this in the results set, and the “Good” class is removed from the training and test data sets.

Now there are two remaining classes. At this point, no more trimming of the dataset is done and a straight classification is performed for the remaining classes.

The set of algorithms and subject classes is applied to the final testing data. A final phase yielded a success result of 97.5%. The best individual classifier was 92.6% and a well known voting approach: bagging and boosting yielded 94.4%.

Table 2. UCI Car Performance Results

Algorithm	Percentage Success
Decision Table	88.4
C4.5	86.8
NN	92.6
Naïve Bayes	83.8
Vector Machine	91.7
Decision Table – Bagging	91.2
Decision Table – Boosting	94.4
C4,5 – Bagging	90.7
C4,5 – Boosting	91.4
NN – Bagging	94.2
NN – Boosting	92.6
Naïve Bayes – Bagging	84.3
Naïve Bayes – Boosting	90.0
Vector Machine – Bagging	91.7
Vector Machine – Boosting	92.4
Hybrid	97.5

Standard deviations for each of the basic, bagging, and boosting success rates is given in Table 3.

Table 3. UCI Car Standard Deviations

Algorithm	Classes			
	Acceptable	Good	Unacceptable	Very Good
Basic	8.65	31.00	1.36	19.66
Voting	8.28	23.76	1.10	18.38
Total	8.90	25.51	1.70	18.29

As shown, the overall success rate of the hybrid voting process, is 97.5%. This compares favorably to the best overall success value for basic classification methods of 92.6% and the best rate for bagging and boosting variants of 94.2%.

4. Results

We used our algorithm on 5 datasets and compared our individual classifiers to our new voting algorithm. As can be seen

in Table 3, hybrid voting frequently outperforms these individual algorithms.

Table 3. Classifier and Hybrid Algorithm Results

Algorithm	Car	Heart	Iris	Nursery	Page Block
Decision Table	88.4	61.3	91.9	88.2	95.2
C4.5	86.8	54.7	91.9	92.7	97.7
NN	92.6	53.3	97.3	96.5	96.3
Naïve Bayes	83.8	54.7	97.3	90.5	94.2
Vector Machine	91.7	62.7	97.3	92.1	92.0
Decision Table Bagging	91.2	69.3	91.9	91.3	95.9
Decision Table Boosting	94.4	61.3	97.3	93.1	96.0
C4,5 Bagging	90.7	56.0	91.9	93.6	97.6
C4,5 Boosting	91.4	61.3	91.9	95.4	97.5
NN Bagging	94.2	56.0	97.3	95.9	96.1
NN Boosting	92.6	57.3	97.3	96.5	96.3
Naïve Bayes Bagging	84.3	57.3	97.3	90.6	94.3
Naïve Bayes Boosting	90.0	54.7	94.6	91.6	94.2
Vector Machine Bagging	91.7	68.0	97.3	92.2	92.3
Vector Machine Boosting	92.4	62.7	97.3	92.1	92.1
Hybrid	97.5	70.7	97.3	96.0	98.7

The results in Table 3 are summarized in Table 4.

Table 4, Algorithm Performance Summary

Dataset	Best Basic		Best Voting		Hybrid
Car	NN	92.6	Decision Table - Boosting	94.4	97.5
Heart	Vector Machine	62.7	Decision Table - Bagging	69.3	70.7
Iris	NN	97.3	NN - Bagging	97.3	97.3
Nursery	NN	96.5	NN - Boosting	96.5	96.0
Page Block	C4.5	97.7	C4.5 - Bagging	97.6	98.7

The algorithms used in the study by Bauer and Kohavi [BK99] were applied in various combinations to fourteen datasets from the UCI data repository. Multiple runs using combinations of algorithms, voting methods, voting methods with various settings, and dataset partitions were made. While the study perorates and analyzes minute nuances of collected results, the broad conclusion is that voting algorithms are facile ways of improving mining algorithm performance. Overall, the boosting algorithms were better than bagging – but they were not uniformly better. Bagging, on the other hand, improved algorithm performance consistently. For comparison, bagging reduced the overall average error rate by 10% for MC4 and 10.6% for Naïve-Bayes. By contrast, boosting proved to be very sensitive to the data being used. For example, it showed an error rate increase for the LED-24 dataset processed by MC4 in combination with AdaBoost of 3.1%. However, Naïve-Bayes with the same boosting technique was able to reduce the average absolute error rate by 24%. The authors conclude that voting methods are an effective way in which algorithm performance may be enhanced

5. Conclusion

We have described a novel voting algorithm based on classifier performance on individual classes within a dataset. We have observed an improvement of 1.0 to 12.8 % over individual classifiers.

While additional investigation and testing is required, these results show great potential for this approach. Future areas for study could cover issues such as:

- Determination of a more efficacious weighting calculation to discern which classes to remove.
- A method to predict whether a dataset could benefit from Hybrid algorithm application.

Although we show an improvement over single classifiers and have done comparisons against boosting and bagging variants of these classifier, Hybrid still needs to be compared to other voting algorithms such as plurality, anti-plurality, plurality-elimination, Borda Count, pairwise comparison, and others cited in Section 2.1.

6. REFERENCES

- [AB00] George T. Albanis, Roy A. Batchelor, 21 Nonlinear Ways to Beat the Stock Market, 2000, City University Business School Department of Banking and Finance, Frobisher Crescent, Barbican Centre, London EC2Y 8HB
- [AB99] George T. Albanis, Roy A. Batchelor, Combining Heterogeneous Classifiers for Stock Selection, 1999, City University Business School Department of Banking and Finance, Frobisher Crescent, Barbican Centre, London EC2Y 8HB
- [Bre97] Breitman, L., Arcing the Edge, Technical Report #486. Department of Statistics, 1997, University of California, Berkeley
- [BFOS84] L. Briedeman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, 1984
- [BK99] Eric Bauer and Ron Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. The journal Machine Learning Vol 36, Nos. 1/2, July/August 1999, pages 105-139.
- [BS00] Pavel B. Brazdil, Carlos Soares, A Comparison of Ranking methods for Classification Algorithm Selection. University of Porto, Porto, Portugal, 2000
- [GB95] J. Gama, P. Brazdil, Characterization of Classification Algorithms, University of Porto, Portugal, 1995
- [Gen97] R. Gentleman, R Project for Statistical Computation, 1997
- [GG04] Eric Graubins, David Grossman, Applying Hybrid Modeling to Predicting the Stock Market, International Federation of Classification Societies Conference 2004, (IFCS'04), 2004
- [KEGSST98] Michel A. King, John F. Elder, Brian Gomolka, Eric Schmidt, Marguerite Summers, Kevin Toop, Evaluation of Fourteen Desktop Data Mining Tools, University of Virginia, 1998
- [LP03] Kelvin T. Leung, D. Stott Parker, Empirical Comparisons of Various Voting Methods in Bagging, SIGKDD 03, 2003
- [Qui93] J.R. Quinlan, C4.5 Programs for Machine Learning, Morgan Kaufman, San Francisco, 1993
- [RH00] Jesse A. Reichler, Harlan D. Harris, Parallel Online Arcing with a Mixture of Neural Networks, Department of Computer Science, University Of Illinois, 2000
- [SB00] Carlos Soares, Pavel B. Brazdil, Zoomed Ranking: Selection of Classification Algorithms Based on Relevant Performance Information, University of Porto, Portugal, 2000
- [SWCI01] Kate A. Smith, Frederick Woo, Vic Ciesielski, Remzi Ibrahim. Modelling the Relationship Between Problem Characteristics and Data Mining Algorithm Performance Using Neural Networks. Intelligent Engineering Systems Through

Artificial Neural Networks, Volume 11, New York, Nov. 2001,
ASME

[UCI04] University of California, Irvine Machine Learning
Repository, <http://www.ics.uci.edu/~mlearn>

[WF00] Ian H. Witten, Eibe Frank, Data Mining, Academic Press,
San Francisco, CA, USA, 2000

[WM96] D.H.Wolpert, W.G.Macready, No Free Lunch Theorems
for Search, Technical Report SFI-TR-95-02-010, The Santa Fe
Institute, 1996.