A Combination of Decision Trees and Instance-Based Learning

Master's Scholarly Paper
Peter Fontana, pfontana@cs.umd.edu
March 21, 2008

Abstract

People are interested in developing a machine learning algorithm that works well in all situations. I proposed and studied a machine learning algorithm that combines two widely used algorithms: decision trees and instance-based learning. I combine these algorithms by using the decision trees to determine the relevant attributes and then running the noise-resistant instance-based learning algorithm with only the attributes used in the decision tree. After using decision tree algorithms and instance-based learning algorithms from WEKA and data from the UCI Machine Learning Repository to test the combined algorithm, I concluded that this combination of the two algorithms did not produce a better algorithm. [3,6,9] My belief for this is that the attributes that one algorithm feels are relevant are likely different from the attributes that the other algorithm feels is relevant.

Introduction

Generating a Machine Learning algorithm that performs well in general is an open problem. Currently, there does not exist a general-purpose Machine Learning algorithm that performs well in all situations. However, many algorithms perform well in many situations, each with their own strengths and weaknesses. Two of these widely used and well-developed Machine Learning algorithms are instance-based learning, developed by Aha, Kibler and Albert, [2] and decision trees, initially developed by Quinlan [8] [4,7].

While both are powerful and effective machine learning tools, both have their weaknesses. Instance-based learning is poor at recognizing and dealing with irrelevant attributes [4,8], and decision trees are not very resistant to noise, even after pruning the tree. However, since decision trees select nodes based on how well certain attributes separate instances and ignore attributes that do not distinguish the data very much [7], decision trees are good at dealing with irrelevant attributes. In addition, as stated in Mitchell [7], Instance-based learning is good at dealing with noise. [7]

One method to deal with this, as described by Aha [1] involves augmenting an instance-based algorithm by giving weights to the attributes and having the algorithm learn the feature weights, and then using the weighted features when computing the similarity between two instances. It would handle irrelevant attributes by giving them low weights. [1]

Also, people have considered combining learning algorithms together. Domingos [4] cites this approach as multi-strategy learning. [4] One such combination is Lazy Decision Trees, where decision trees are constructed in a lazy fashion, using an instance-based approach to form a unique decision tree for each instance [5].

I predicted that combining these two algorithms together would result in a better, more general-purpose Machine Learning algorithm. I predicted that I could use a decision tree to determine which attributes were relevant and then use instance-based learning that only considered the attributes used in the decision tree to classify in a noise-resistant way, with

improved performance. This specific combination of decision trees and instance-based learning has not been done before. My hypothesis is that the learning algorithm that uses instance-based learning on only the attributes selected by the decision tree will perform better in general compared to the decision tree alone and compared to the instance-based learner alone. This is because I predict that this algorithm will combine the ability of the decision tree to ignore irrelevant attributes with the noise-resistance of the instance-based learner.

Methods

To test my hypothesis, I utilized an implementation of C4.5 Decision Trees, both with pruned and unpruned decision trees, (WEKA's (Waikato Environment for Knowledge Analysis) J4.8) and an implementation of the IB1 Instance-based learner with 1-nearest neighbor (WEKA's IB1) and with 3-nearest neighbor (WEKA's IBK with the nearest parameter set to 3) [6,9]. First, I obtained various data sets from the UCI (University of California at Irvine) Machine Learning Repository such that the class variable was a nominal attribute. [3] I then randomly partitioned the data into a 75% training set and a 25% test set. I then with each data set, ran it on WEKA's IB1 and WEKA's IB1 3-Nearest Neighbor (WEKA's IBK with the nearest neighbor set to 3) and on WEKA's decision tree algorithm (J4.8) using both its pruned and unpruned decision trees (J4.8 produces an unpruned tree by setting the unpruned parameter to true) [6,9]. For each run I recorded the percent of examples of the test set classified correctly.

Then, I took the decision trees tested by WEKA, and implemented a Java program that took a WEKA decision tree (saved in an individual file) and the corresponding .arff file (the file format that WEKA uses to store and read data from), and made a new .arff file that contained the original data but with only the attributes that were used in the decision tree (and the class attribute). [6,9] This algorithm considered an attribute relevant if it was looked at anywhere in the decision tree when it made a decision. Even if only some of the possible values for the attribute were considered, the algorithm would still use all the values of the attribute. For example, if there was an attribute `Color`, and the only node with `Color` in the decision tree was whether `Color == Blue` or `Color != Blue`, the algorithm would still store the exact value of `Color` for all of the data instances. I ran this algorithm using the pruned decision tree and the unpruned decision tree and on the training data files and the test data files, producing 4 .arff files per data set. This way I used the same training and test set partitions for all tests with the same data set. I then ran IB1 (both the 1-nearest neighbor version and the 3-nearest neighbor version) on these revised data sets, and recorded the results. The data tables and charts plotting the data results are in the Results section of this paper. For some more information on the data, see the Data section of this paper.

Data

All the data was obtained from the UCI Machine Learning Repository. [3] I used the `Iris`, `cpu-performance`, `Spambase`, `soybean` and `glass` data sets from the Repository. [3] Most of the data sets were used as-is by the learning algorithms, but I modified two of those data sets. The first data set was the `cpu-performace` data set. It's class attribute `PRP` (Published relative performance), was continuous. To make it discrete, the documentation with the data file gave performance ranges. I then wrote a script to take the original file and produce a file that produced a discrete `PRP` ordinal attribute by converting each

value into its range. Since ERP (Estimated Relative Performance) was a similar attribute, but not the class attribute, I dealt with it in three different ways: I left it as a continuous attribute, I discretized it like I discretized the PRP attribute and I produced a data set without the ERP attribute. [3] Also, the model attribute was a set of strings, so I changed the attribute category from just string to a nominal attribute that contained all of the model names as possible values.

The glass data set also had an interesting property. Each instance had an ID number as an attribute. While this attribute is usually irrelevant, it happened in this data set the instances were sorted by class, and the ID numbers were assigned in increasing order to the instances. Since testing divided this data set into random instances, it made the ID attribute extremely relevant since the ranges of ID numbers corresponded to all instances of the class. So I ran the algorithm on the glass data set with the ID attribute and recorded the results, and then I made another version of the data set that did not contain the ID attribute and ran the data through the algorithms and produced a different result.

All the results of the data including all 3 runs with the cpu-performance data set and both runs with the glass data set are given in the results section.

Results

Below are data tables and charts with the results. The first two tables gives the percent of instances correct on the 25% test set. Note that a * in the data set entry (in the table and in the chart) indicates that the pruned decision tree and unpruned decision tree were identical.

Data Set	Pruned Decision Tree	Unpruned Decision Tree	IB1	IB1-3NN
Iris *	97.0588%	97.0588%	97.0588%	97.0588%
CPU-Performance (no ERP)	52.6316%	64.9123%	66.6667%	56.1404%
CPU-Performance (Discrete ERP)	62.5000%	64.2857%	67.8571%	69.6429%
CPU-Performance (Continuous ERP)	63.0435%	71.7391%	71.7391%	67.3913%
Spambase	92.7046%	92.9715%	89.5018%	88.9680%
Soybean	93.0818%	91.1950%	88.6792%	89.3082%
Glass *	100.0000%	100.0000%	94.0299%	95.5244%
Glass (No ID)	70.1493%	70.1493%	64.1791%	64.1791%

Table 1: Percent Correct on Test set for original learning algorithms.

Data Set	IB1 with Pruned Tree	IB1-3NN with Pruned Tree	IB1 with Unpruned Tree	IB1-3NN with Unpruned Tree
Iris *	97.0588%	97.0588%	97.0588%	97.0588%
CPU-Performance (no ERP)	63.1579%	54.3860%	68.4211%	57.8947%
CPU-Performance (Discrete ERP)	62.5000%	64.2857%	64.2857%	67.8571%
CPU-Performance (Continuous ERP)	73.9130%	63.0435%	71.7391%	63.0435%
Spambase	89.2349%	89.1459%	89.8577%	89.0569%
Soybean	71.6981%	75.4717%	80.5031%	78.6164%
Glass *	98.5075%	98.5075%	98.5075%	98.5075%
Glass (No ID)	65.6716%	65.6716%	65.6716%	65.6716%

Table 2: Percent Correct on Test sets for combined algorithms

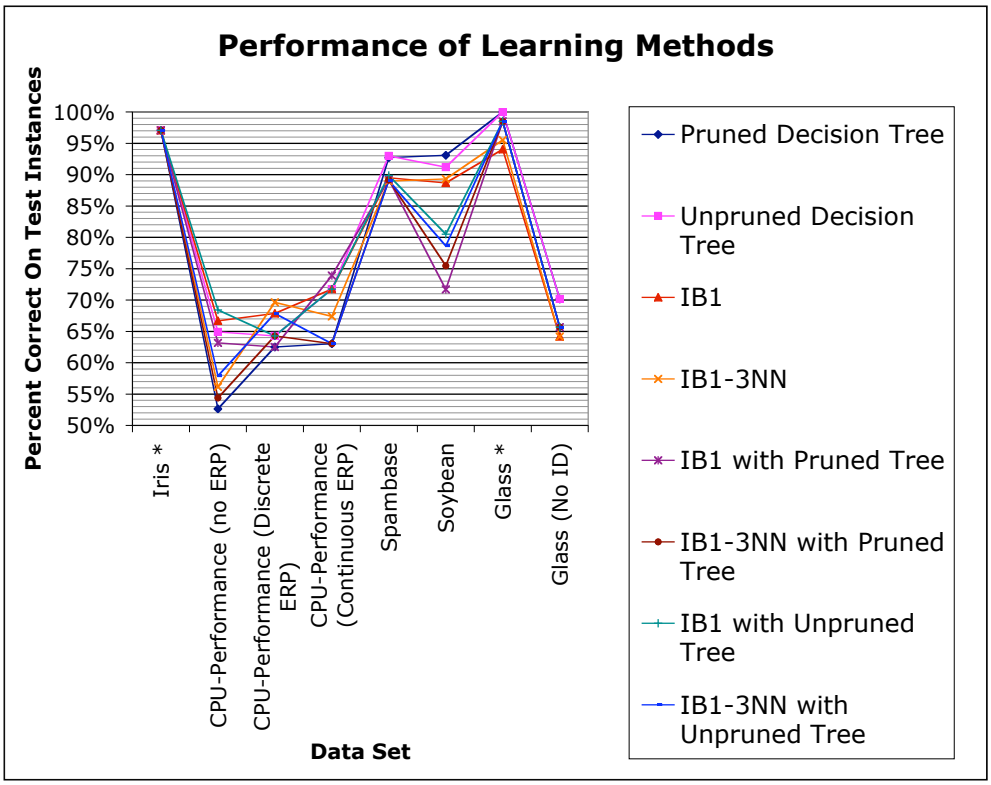

Chart 3: Plot of the percent correct of the various learning algorithms.

Now, I compare the differences between various algorithms. Since I am interested in determining if the combined algorithm does better than pruned decision trees alone or instance-based learning alone, I will give those differences in the tables below and will produce a few charts. Since unpruned decision trees are believed to overfit the data, I do not compare the combined learning algorithm to unpruned decision trees alone.

Data Set	(IB1 with Unpruned) - (Pruned DT)	(IB1 with Pruned) - (Pruned DT)	(IB1-3NN with Unpruned) - (Pruned DT)	(IB1-3NN with Pruned) - (Pruned DT)
Iris *	0.0000%	0.0000%	0.0000%	0.0000%
CPU-Performance (no ERP)	15.7895%	10.5263%	5.2631%	1.7544%
CPU-Performance (Discrete ERP)	1.7857%	0.0000%	5.3571%	1.7857%
CPU-Performance (Continuous ERP)	8.6956%	10.8695%	0.0000%	0.0000%
Spambase	-2.8469%	-3.4697%	-3.6477%	-3.5587%
Soybean	-12.5787%	-21.3837%	-14.4654%	-17.6101%
Glass *	-1.4925%	-1.4925%	-1.4925%	-1.4925%
Glass (No ID)	-4.4777%	-4.4777%	-4.4777%	-4.4777%

Table 4: The percent difference between the combined algorithm and the pruned decision tree. A positive number indicates that the combined algorithm improved performance.

Data Set	(IB1 with Pruned) - (IB1)	(IB1 with Unpruned) - (IB1)	(IB1-3NN with Pruned) - (IB1-3NN)	(IB1-3NN with Unpruned) - (IB1-3NN)
Iris *	0.0000%	0.0000%	0.0000%	0.0000%
CPU-Performance (no ERP)	-3.5088%	1.7544%	-1.7544%	1.7543%
CPU-Performance (Discrete ERP)	-5.3571%	-3.5714%	-5.3572%	-1.7858%
CPU-Performance (Continuous ERP)	2.1739%	0.0000%	-4.3478%	-4.3478%
Spambase	-0.2669%	0.3559%	0.1779%	0.0889%
Soybean	-16.9811%	-8.1761%	-13.8365%	-10.6918%
Glass *	4.4776%	4.4776%	2.9831%	2.9831%
Glass (No ID)	1.4925%	1.4925%	1.4925%	1.4925%

Table 5: The percent difference between the combined algorithm and the Instance-based algorithms. A positive number indicates that the combined algorithm improved performance.

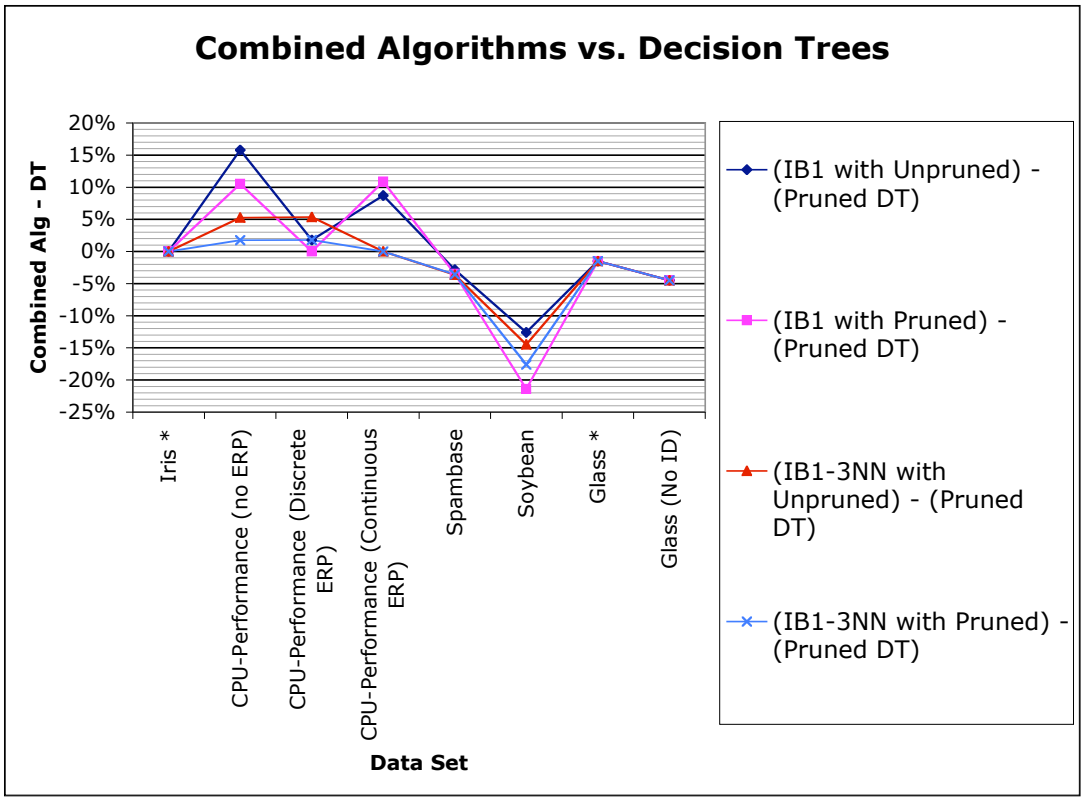

Chart 6: Plot of (the Percent Correct of the combined algorithm) – (the percent correct by the pruned decision tree).

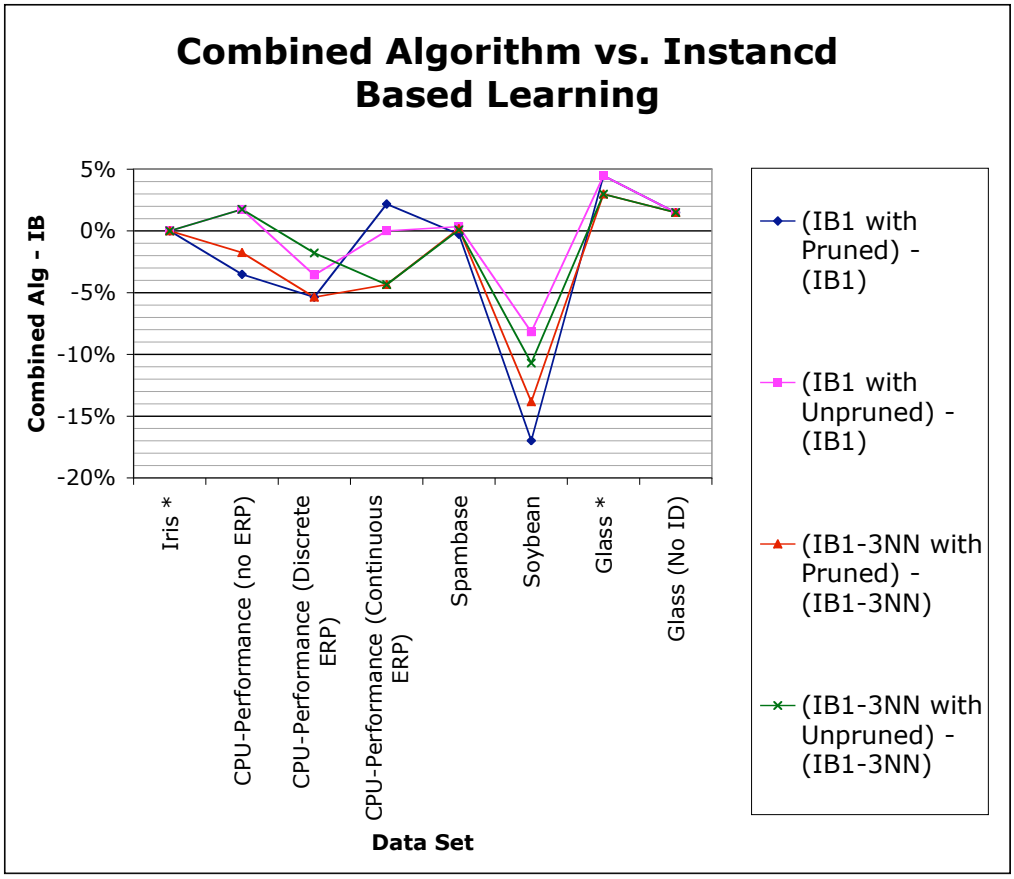

Chart 7: Plot of (the Percent Correct of the combined algorithm) – (the percent correct by the instance-based algorithm).

From this data, I can conclude that this method of combining decision trees and instance-based learning does not result in significantly better results. While I understand that if these results were good, it may require more data sets to be able to show significant results, this amount of data is adequate to show that there was no significant improvement. I discuss possible causes of this lack of improvement in the Conclusions section of this paper.

Conclusions

Based on the data, I conclude that this method of combining decision trees and Instance-based learning does not produce a significantly better algorithm. I can make this conclusion since the combined algorithms gave fewer correct answers on a significant percentage of the data sets. The closest improvement is IB1 with the Unpruned Tree compared to IB1. However, it only improved on 5 out of the 7 data sets and the gain is slight (usually only 1-2%). Based on the analysis, the soybean data may be a fluke or extremely different data set, since performance was far worse for the combined algorithms on this data set. However, since the number of data sets used is small, I am unable to conclude this. Even if the soybean data set is ignored, the gain of the combined algorithms is very slight compared to the instance-based learners and to the pruned decision trees on the data sets that do show an improvement. Many of the data sets resulted in the combined algorithm performing worse.

Also, while the combined algorithm sometimes shows significant gains when compared to the Decision trees, much of this difference is due to the instance-based learners doing better than decision trees on these algorithms.

Therefore, I reject my hypothesis that combining a decision tree and instance-based learning in this method (by using the decision tree to determine the relevant attributes for the instance-based learner). One possible reason for the lack of performance improvement is that decision trees use attributes to distinguish instances from each other while instance-based learning uses attributes to determine how similar instances are from each other. This may pose a problem, since attributes that are good at differentiating instances may not be good indicators of similar instances, and vice versa.

Another reason is that the attributes that one algorithm considers relevant may be different from the attributes that are relevant for the other algorithm. This is likely, since the two algorithms represent the data different and interact with the data differently. Instance-based learning takes instances and looks for similarities between the instances. On the other hand, decision trees often look at the attributes and distinguish the instances based on what the differences between the instances are. These two different approaches may result in what is relevant for one algorithm to be not very relevant for the other algorithm.

While this method of combining the two algorithms did not produce better results, it may be that these algorithms can produce better classification results when combined in some other way. That is future work.

Acknowledgements

I thank Dr. James Reggia for his advice and helpful discussions on this research and this paper.

References

[1] Aha, David.W. (1998) Feature weighting for lazy learning algorithms. In: H. Liu and H. Motoda (Eds.) Feature Extraction, Construction and Selection: A Data Mining Perspective. Norwell MA: Kluwer, 1998.

[2] Aha, David, Dennis Kibler and Mark K. Albert. Instance-Based Learning Algorithms, *Machine Learning*, 6, 1991, 37-66.

[3] Asuncion, A & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science. Last Accessed December 2, 2007.

[4] Domingos, P. (1996). Unifying Instance-Based and Rule-Based Induction. *Machine Learning*, 24:141-168.

[5] Friedman, J.H., R. Kohavi and Y. Yun. Lazy decision trees. *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*. AAAI Press, 1996, 717-724.

[6] Ian H. Witten and Eibe Frank (2005) Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005. (Source of WEKA)

[7] Mitchell, Tom. Machine Learning. McGraw Hill, 1997.

[8] Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning.* 1, 1 (Mar. 1986), 81-10

[9] WEKA Software. The University of Waikato. [http://www.cs.waikato.ac.nz/ml/weka/]. Last Accessed December 2, 2007. (Where WEKA was obtained from).