To Appear in KDD-98

Targeting Business Users with Decision Table Classifiers

Ron Kohavi and **Daniel Sommerfield**
Data Mining and Visualization
Silicon Graphics, Inc.
2011 N. Shoreline Blvd
Mountain View, CA 94043-1389
{ronnyk,sommda}@engr.sgi.com

Abstract

Business users and analysts commonly use spreadsheets and 2D plots to analyze and understand their data. On-line Analytical Processing (OLAP) provides these users with added flexibility in pivoting data around different attributes and drilling up and down the multi-dimensional cube of aggregations. Machine learning researchers, however, have concentrated on hypothesis spaces that are foreign to most users: hyperplanes (Perceptrons), neural networks, Bayesian networks, decision trees, nearest neighbors, etc. In this paper we advocate the use of decision table classifiers that are easy for line-of-business users to understand. We describe several variants of algorithms for learning decision tables, compare their performance, and describe a visualization mechanism that we have implemented in MineSet. The performance of decision tables is comparable to other known algorithms, such as C4.5/C5.0, yet the resulting classifiers use fewer attributes and are more comprehensible.

Introduction

About two years ago we met with a director in charge of sales analysis and forecasting at our company. We extracted actual data and showed him and his team ("the clients") some data mining and visualization analyses using two classifier models: decision trees and Naive Bayes. The dataset we used was relatively large for common machine learning techniques at the time: about one million records with 70 attributes each. The decision tree visualizer we used was interactive and allowed fly-throughs over the model in order to cope with large trees that were created, commonly with thousands of nodes. The Naive Bayes visualizer also provided a visual display of the classifier, but the model was less complex, containing only a few visual elements for each attribute and its size was independent of the number of records (other than indirectly through discretization of continuous attributes). After several meetings with the clients, we made the following observations:

1. It took longer than we thought to explain to the clients what the models mean.
2. The clients found the Naive Bayes classifier the much more interesting of the two. It was relatively easy for them to understand how an attribute value changes the label probabilities. However, they soon realized the "naive" nature of this model, which assumes conditional independence of the attributes given the label. They still thought of it as a powerful tool, but they themselves were looking at interactions through spreadsheets and OLAP analysis, and those patterns that they were finding "the hard way" could not be seen in the Naive Bayes model.
3. The clients found some interesting patterns in the decision trees, but they did not feel the structure was natural for them. They were looking for those two or three attributes and values (*e.g.*, a combination of geographies and industries) where something "interesting" was happening. In addition, they felt it was too limiting that the nodes in a decision tree represent rules that all start with the same attribute.

In response to the above observations, we considered several possibilities and chose a spreadsheet-like classifier model that can be presented visually. This direction was reinforced when the director mentioned in one of the meetings that he wanted a Karnaugh-like map of his data—a map commonly used to present Boolean functions visually and which has been extended to non-Boolean functions by several authors (Michalski 1978; Wnek & Michalski 1994; LeBlank, Ward, & Wittels 1990) under the names General Logic Diagrams and Dimensional Stacking. At the time, a new On-Line Analytical Processing (OLAP) system was being fielded in our company, and our clients found the projections of multi-dimensional cubes onto spreadsheets very intuitive. Our prior work with decision tables (Kohavi 1995a; 1995b) and General Logic Diagrams (Kohavi, Sommerfield, & Dougherty 1997) was encouraging and we decided to develop the algorithm and the visualizations further. The prior work built decision tables by conducting a wrapper-based attribute selection, starting from the empty set of attributes.
Our contributions in this paper include:

1. A fast attribute selection mechanism based on entropy.

2. A new mechanism for handling instances not found in the decision table that has a desirable property of breaking in favor of a local neighborhood (unlike the global majority rule used in the original work on decision tables).
3. A different way of storing the data using an oblivious decision tree structure (Kohavi & Li 1995), which not only provides an efficient structure for building decision tables, but also provides support for the above mechanism and extends much better to parallel implementations.
4. An empirical evaluation of several variants for selecting attributes.
5. A description of an interactive visualizer for decision table classifiers.

The Decision Table Classifier

Given a training set of labeled instances, an induction algorithm builds a classifier. We describe two variants of decision table classifiers based conceptually on a simple lookup table. The first classifier, called **DTMaj** (Decision Table Majority) returns the majority of the training set if the decision table cell matching the new instance is empty, *i.e.*, it does not contain any training instances. The second classifier, called **DTLoc** (Decision Table Local), is a new variant that searches for a decision table entry with fewer matching attributes (larger cells) if the matching cell is empty. This variant therefore returns an answer from the local neighborhood, which we hypothesized will generalize better for real datasets that tend to be smooth, *i.e.*, small changes in a relevant attribute do not result in changes to the label.

For the rest of this paper, we assume the attributes are discrete. In our experiments, we pre-discretized the data as described in Fayyad & Irani (1993) and Kohavi & Sahami (1996), where the discretization thresholds are determined based only on the training set and then applied to the test set.

The Functional Definition

A decision table has two components:

1. A **schema**, which is a list of attributes[1].
2. A **body**, which is a multiset of labeled instances. Each instance consists of a value for each of the attributes in the schema and a value for the label. The set of instances with the same values for the schema attributes is called a cell.

Given an unlabeled instance, $\vec{x}$, the label assigned to the instance by a decision table classifier is computed as follows. Let $\mathcal{I}$ be the set of labeled instances in the cell that exactly matches the given instance $\vec{x}$, where only the attributes in the schema are required to match and all other attributes are ignored. If $\mathcal{I} \neq \emptyset$, return the majority class in $\mathcal{I}$, breaking ties arbitrarily. Otherwise ($\mathcal{I} = \emptyset$), the behavior depends on the type of decision table used:

1. A DTMaj returns the majority class in the decision table.
2. A DTLoc removes attributes from the end of the list in the schema and tries to find matches based on fewer attributes until one or more matches are found and their majority label is returned. This increases the cell coverage until training instances match $\vec{x}$.

Unknown values are treated as distinct values in the matching process.

Given a dataset and a list of attributes for the schema, a decision table is well defined functionally. Before we describe how to choose the list of attributes (the induction step), we show how a DTLoc can be implemented efficiently, *i.e.*, we show how one can quickly find the exact set of instances matching the largest prefix of attributes in the schema.

The Implementation of the Decision Table Classifier

A decision table can be stored in different data structures. A DTMaj can be implemented in a universal hash table. Such a hash table provides very efficient insert-instance and classify-instance operations, but a DTLoc is harder to implement this way efficiently.

A DTLoc requires that we use smaller subsets of the list if an instance is not found. The data structure we used to implement this is an oblivious decision tree, a structure similar to a decision tree except that at a given level, every node tests the same attribute. As in decision trees, leaves are labeled with the majority class. Given an instance, a DTLoc classifier traces the path from the root to a leaf, branching according to the attribute values, and predicting the label at the leaf. Because empty nodes are never created, this data structure provides an efficient implementation of the DTLoc. The leaf represents the majority class for the instances matching the longest prefix of the schema attributes.

Inducing Decision Tables

The task of the inducer is to find the optimal list of attributes for the schema, *i.e.*, the list of attributes such that the decision table created from this list will have the lowest possible error on the population from which the training set was sampled.

Unlike previous work, which used the wrapper approach (Kohavi 1995a; Kohavi & John 1997), we chose a much faster entropy-based attribute selection. This methods finds the attribute that maximizes the mutual information for each level of the oblivious tree as described in Kohavi & Li (1995). Splitting continues until the leaves are pure or until more than 200,000 leaves are required. The error for each oblivious tree is estimated using cross-validation or holdout depending on the size of the training set: below 5000 records 10-fold cross-validation is used and above 5000 records a

[1] We require a list here and not a set as in Kohavi (1995b) to support DTLoc.

1/3 holdout is used. The oblivious tree with the lowest estimated error is chosen.

Because entropy is known to favor multi-way splits (Cover & Thomas 1991; Quinlan 1993), we tested a related variant, **normalized mutual information**, which penalizes multi-way splits by dividing the gain from the root by $\log \ell$, where ℓ is the number of leaves.

During our initial experiments, we observed large differences in error rates between the two approaches, yet neither was significantly better on most datasets. We therefore decided to run both methods and choose the one that has lower estimated error on the training set (the test set is never used). We found that this worked well and usually selected the version that actually had the lower estimated error on the test set.

Empirical Evaluation

We conducted an empirical evaluation of the induction algorithms on several datasets from the UCI repository (Merz & Murphy 1998). We chose the larger datasets available, mostly natural but also a few artificial ones (m-of-n and the monk problems). The artificial datasets were tested on the given training and test sets. The natural datasets were evaluated using 10-fold cross-validation if the file size was less than 3,000 records (to ensure a small standard deviation of the mean estimated error) and using 1/3 holdout if the file size was over 3,000 records. We compared the following algorithms:

C4.5 This is the well-known decision tree algorithm by Quinlan (1993). We used release 8, the latest available. We also ran C5.0, which is a commercial successor to C4.5, but the differences were very minor and we could not easily parse the C5.0 trees in their binary format to determine the number of attributes used.

DTMaj-O Decision tables with majority predictions for unknown cells. The "O" indicates the automated "optimal" choice (based on the training set only) between the set of attributes suggested by mutual information and normalized mutual information.

DTLoc-O Same as DTMaj-O, except that the majority predictions are replaced by local predictions.

Figure 1 shows a comparison of the above algorithms. Between the two decision table variants, statistically significant differences at the 95% confidence include[2]: DTLoc is superior for m-of-n, segment, shuttle; DTMaj is superior for monk1 and monk2.

Comparing C4.5 and DTLoc, statistically significant differences occur include: C4.5 is superior for breast-cancer (Ljubljana), letter, nursery, satimage; DTLoc is superior for breast (Wisconsin), cleve, german, m-of-n, monk1, and monk2-local.

We hypothesize that decision tables will generally be inferior for multi-class problems because the need to chose a split across a level forces compromises. Letter has 26 classes, nursery has five, satimage has seven, segment has seven. We further hypothesize that decision tables will generally be superior in noisy domains where using the whole dataset for each choice of attribute ensures stability.

Decision tables use dramatically fewer attributes than C4.5 in most datasets. In fact, it is quite surprising to see how well decision tables perform with five or fewer attributes. Out of 16 natural datasets, only DNA and german used more than five attributes (chess is semi-natural) with DTMaj, and only DNA and letter used more than seven attributes with DTLoc. Such relatively small classifiers can be understood by humans, especially if coupled with a good visualizer.

Running times for DTMaj and DTLoc were similar. The longest runs were for Adult and Letter, which took about 12 minutes each on a 195Mhz R10K Silicon Graphics Origin.

In addition to the above experiments, we also tried forward search using the wrapper approach and a variant that starts the wrapper search from the attributes selected using entropy. The error rates were slightly lower, but perhaps not enough to warrant the runtime difference. Because these completely different approaches achieved very similar error rates, we believe that the induction algorithms are finding nearly optimal attribute subsets.

Visualizing Decision Tables

Decision table classifiers can have hundreds or thousands of cells on medium to large datasets. If the table has 5 attributes, each with 4 values, then the table might have $4^5 = 1024$ cells[3]. It is unrealistic to expect that users will want to look at such a table at this level of detail without seeing coarser views of the data first.

Figure 2 shows two snapshots of an interactive decision table visualizer. On the left, the visualizer shows the top-level coarse split based on the first two attributes in the schema. For each intersection of the attribute values, a cake is shown with slice sizes representing the class distribution for those two attributes. In addition to the slices, the cake has a height, which is relative to the number of records for that intersection. The user can click on any set of cakes and the display will show the next pair of attributes, providing drill-downs to the area of interest as shown on the right.

Intersections that do not have data are shown in the background color in order to show where data is missing and help identify correlations and functional dependencies. The ability to see contiguous areas with similar cake distributions allows users to quickly generalize (*e.g.*, similar cakes across three ranges of TSH for FTI between 48.5 and 64.5 shown in Figure 2 on the left).

[2]The standard deviations are not given here due to lack of space, but they were computed.

[3]This is an upper bound. In practice, the number is usually significantly lower due to correlations in the data and lack of sufficient data.

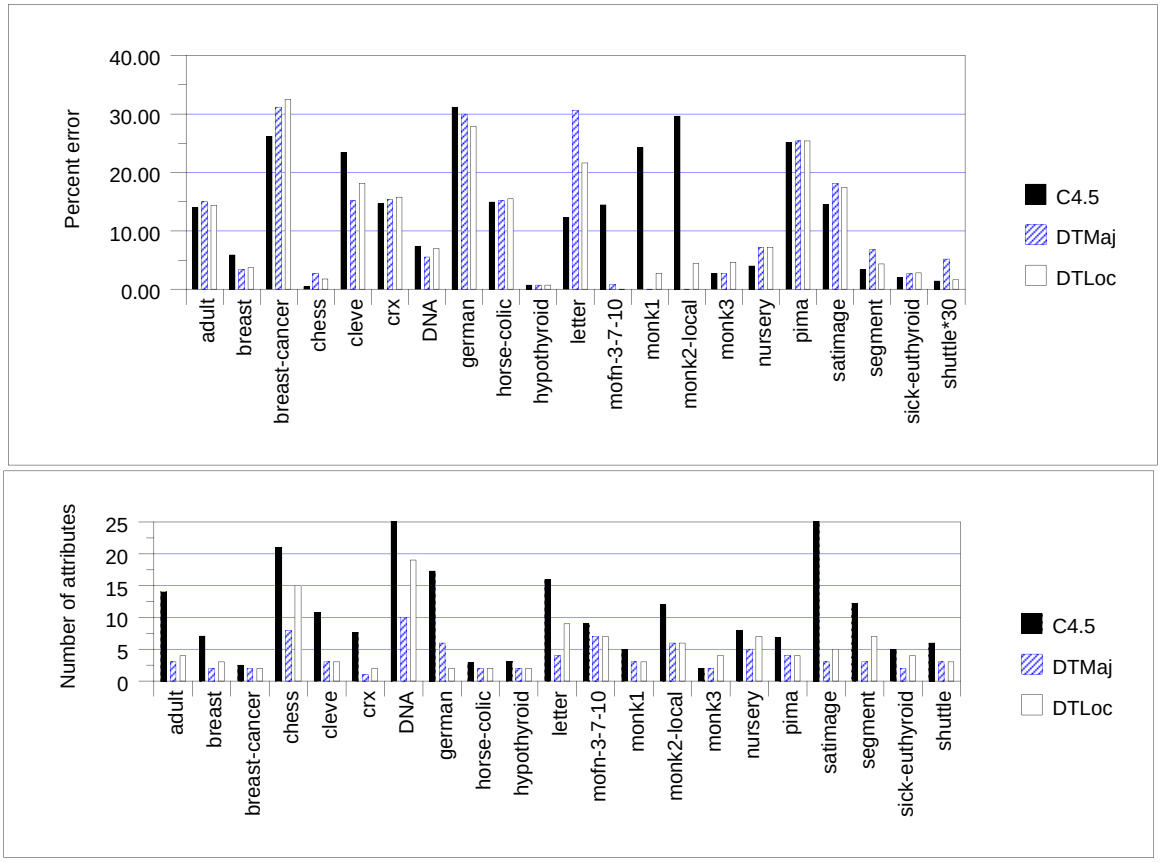

Figure 1: The percent error and number of attributes used in the model for C4.5 and several variants of decision tables (lower is better in both figures). Shuttle*30 has error multiplied by 30 for the appropriate scale. C4.5 used 46 attributes for DNA and 36 for Satimage (both extend beyond the axis).

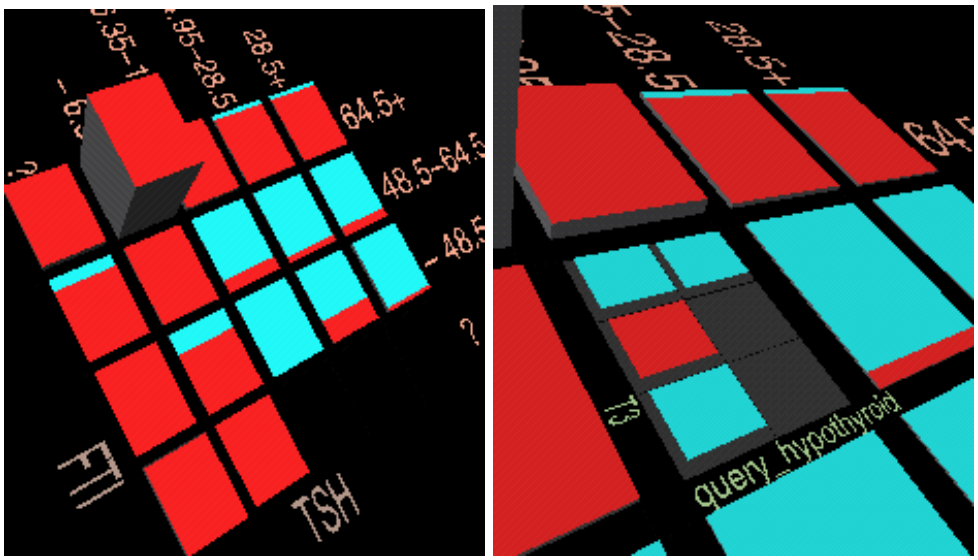

Figure 2: Visualizations of a decision table for the hypothyroid dataset. The left figure shows the top-level view with two attributes: FTI and TSH. Users can see that several intersections are empty: high TSH values imply unknown FTI (probably not measured), and that most data is in the low range of TSH (below 6.3). High values for FTI (above 64.5) are negative hypothyroid with high probability (dark gray). The interesting intersections are for low FTI and high TSH. Drilling down on the middle cake yields the figure on the right, which shows pure intersections once two more variables are taken into account (query_hypothyroid and t3). This is a relatively easy model to comprehend.

Future Work

Our work can be naturally extended in several areas. The discretization is done only once at the beginning, but could be done at each level in order to capture interactions better. Our handling of unknowns (as a value) is simple and more complex algorithms, such as those employed in C4.5/C5.0 can be used.

We used decision tables for classification, but regression is a possible extension. The visualizer naturally extends to regression, but the entropy-based attribute selection needs to be replaced by another criteria since it relies on discrete labels. Common regression-tree measures (Breiman *et al.* 1984) might work well.

Techniques for multiple model generation, such as Bagging Boosting, or Option Trees might be helpful, especially if they yield different top-level attributes and users can select between them.

Summary

Early work on decision tables (Kohavi 1995a) was promising but was too slow for common use because it was based on a forward selection of attributes using the wrapper model. Our research showed that entropy-based methods, which are much faster, are practically indistinguishable from their more expensive predecessors for error minimization.

We showed that for many of the larger datasets from UCI, a very small number of attributes suffices to achieve comparable accuracy to C4.5, our benchmark algorithm. This observation reinforces claims by Holte (1993) that on natural datasets, a few attributes usually suffice for accurate classifications. When few attributes were used, few test instances ended in empty cells, thus performance differences between global and local handling were small.

Coupled with a good visualizer, decision table classifiers are easy for business users to understand and provide a natural complement to spreadsheets and online analytical processing tools that are commonly used in business settings. Decision tables move us a step forward in satisfying one of the main goals of data mining—improved time to insight.

Acknowledgments We would like to thank Barry Becker for his design and implementation of the decision-table visualizer and Jay Desouza for his initial work on writing the oblivious-decision-tree based decision tables. Jay Desouza was funded by Silicon Graphics for his work. We wish to thank Rick Kufrin from NCSA for his initial help. The work for this paper was done using $\mathcal{MLC}$++ (Kohavi, Sommerfield, & Dougherty 1997).

References

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees.* Wadsworth International Group.

Cover, T. M., and Thomas, J. A. 1991. *Elements of Information Theory.* John Wiley & Sons.

Fayyad, U. M., and Irani, K. B. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022–1027. Morgan Kaufmann Publishers, Inc.

Holte, R. C. 1993. Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 11:63–90.

Kohavi, R., and John, G. H. 1997. Wrappers for feature subset selection. *Artificial Intelligence* 97(1-2):273–324.

Kohavi, R., and Li, C.-H. 1995. Oblivious decision trees, graphs, and top-down pruning. In Mellish, C. S., ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1071–1077. Morgan Kaufmann.

Kohavi, R., and Sahami, M. 1996. Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 114–119.

Kohavi, R.; Sommerfield, D.; and Dougherty, J. 1997. Data mining using $\mathcal{MLC}$++: A machine learning library in C++. *International Journal on Artificial Intelligence Tools* 6(4):537–566.
`http://www.sgi.com/Technology/mlc`.

Kohavi, R. 1995a. The power of decision tables. In Lavrac, N., and Wrobel, S., eds., *Proceedings of the European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence 914, 174–189. Berlin, Heidelberg, New York: Springer Verlag.
`http://robotics.stanford.edu/~ronnyk`.

Kohavi, R. 1995b. *Wrappers for Performance Enhancement and Oblivious Decision Graphs.* Ph.D. Dissertation, Stanford University, Computer Science department. STAN-CS-TR-95-1560,
http://robotics.Stanford.EDU/~ronnyk/teza.ps.Z.

LeBlank, J.; Ward, M.; and Wittels, N. 1990. Exploring n-dimensional databases. In *Proceedings of Visualization*, 230–237.

Merz, C., and Murphy, P. 1998. UCI repository of machine learning databases.

Michalski, R. S. 1978. A planar geometric model for representing multidimensional discrete spaces and multiple-valued logic functions. Technical Report UIUCDCS-R-78-897, University of Illinois at Urbaba-Champaign.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning.* San Mateo, California: Morgan Kaufmann.

Wnek, J., and Michalski, R. S. 1994. Hypothesis-driven constructive induction in AQ17-HCI : A method and experiments. *Machine Learning* 14(2):139–168.