In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 174–179, AAAI Press, Menlo Park, CA, 1995.

Robust Decision Trees: Removing Outliers from Databases

George H. John
Computer Science Dept.
Stanford University
Stanford, CA 94305
gjohn@CS.Stanford.EDU
http://robotics.stanford.edu/~gjohn/

Abstract

Finding and removing outliers is an important problem in data mining. Errors in large databases can be extremely common, so an important property of a data mining algorithm is *robustness* with respect to errors in the database. Most sophisticated methods in machine learning address this problem to some extent, but not fully, and can be improved by addressing the problem more directly. In this paper we examine C4.5, a decision tree algorithm that is already quite robust – few algorithms have been shown to consistently achieve higher accuracy. C4.5 incorporates a pruning scheme that partially addresses the outlier removal problem. In our ROBUST-C4.5 algorithm we extend the pruning method to fully remove the effect of outliers, and this results in improvement on many databases.

Introduction

As Knowledge Discovery in Databases (KDD) becomes more common in practice, users will apply KDD methods and algorithms to more and larger databases. A well-known property of large databases is the frequency of errors. Although theory and practice of database management has advanced rapidly over the past two decades (Ullman 1988), in the end the data stored in a DBMS is no better than the typing skill of the data-entry clerk and the decisions made by experts that are recorded in databases. In order to be successful, database mining systems must be tolerant, or *robust* with respect to errors in databases.

We address the following class of problems in this paper: given a database of some arbitrary number of records and fields, and some distinguished field(s) which we would like to be able to predict given the others, discover some pattern in the database and express this pattern in some language. The field to be predicted should be a character field with relatively few distinct values. The language must be sufficiently expressive that it allows a computer to fill in missing values in the distinguished field in future databases of the same schema. The language should also be understandable so that an expert or data analyst can verify that the discovered patterns make sense. Understandability is also important because a KDD system can uncover important general knowledge which should be understood by its users. In this paper we evaluate discovered patterns by both their predictive ability and understandability. Within this framework we consider the problem of selectively disregarding database records based on a heuristic estimate of whether or not a record is veridical.

Although nearly all machine learning and statistical methods address robustness to errors to some extent in the guise of *overfitting avoidance*, this puts the focus on only part of the system. Most methods for overfitting avoidance involve measuring the accuracy of the learned model on the training database, and measuring the complexity of the learned model. The model is then repeatedly made simpler until some desired balance of accuracy and complexity is reached. While our method also penalizes the model for being too complex, our method as well penalizes the data itself for being too hard to learn, by removing records from the training database. (Alternately, one can think of the algorithm reducing the weights of the suspicious records to zero.) As we shall see, the latter method results in much more understandable patterns with about the same predictive ability.

We discuss robust statistical methods in the next section. The following section reviews the C4.5 decision tree and rule induction algorithm, explaining the pruning method used to avoid overfitting and relating it to ideas in robust statistics. We then describe ROBUST-C4.5, our robust modification of C4.5. The following section presents our experiments, which verify our claims. We then discuss related work in machine learning, neural networks and statistics, and then present our conclusions.

Robust Statistical Methods

All physical sciences and engineering disciplines are grounded on observations and measurements of physical entities and events. Whenever such data is collected into a large database, we almost inevitably have the problem of finding and addressing errors in the database. Since this is such a core issue to all physi-

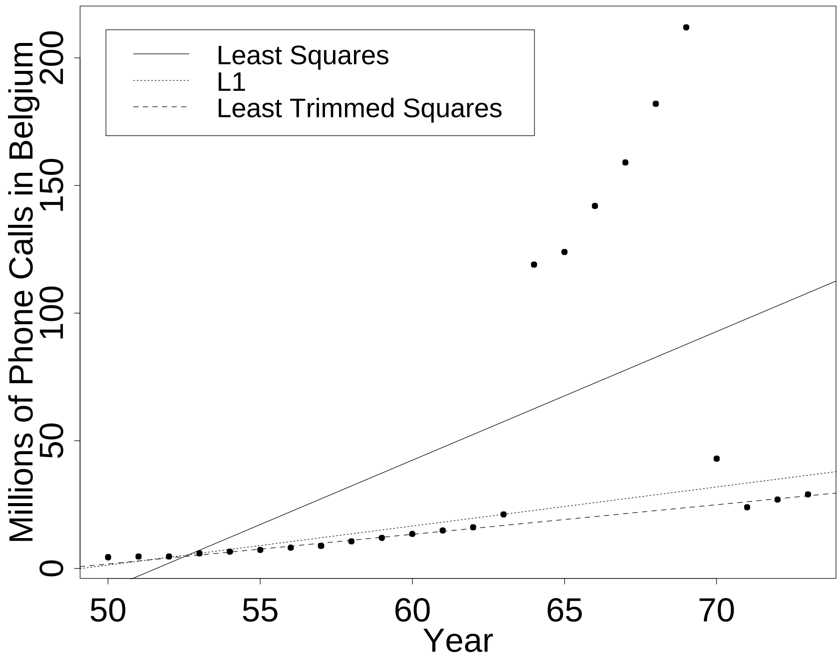

Figure 1: Millions of phone calls in Belgium, 1950–1973, from Rousseeuw & Leroy (1987). Plotted in S-PLUS using code from Venables & Ripley (1994).

cal sciences and engineering disciplines, one might expect that scientists and statisticians have considered the problem for some time. In 1777 while studying astronomical data, Daniel Bernoulli wrote "I see no way of drawing a dividing line between those that are to be utterly rejected and those that are to be wholly retained; it may even happen that the rejected observation is the one that would have supplied the best correction to the others..." (Barnett & Lewis 1993, p. 27).

In statistics, an *outlier* is defined as a "case that does not follow the same model as the rest of the data" (Weisberg 1985). This is broad, including not only erroneous data but also "surprising" veridical data. Most of the work addressing outliers in statistics assumes either a simple estimation task, such as estimating the mean and covariance matrix of a normal distribution, or a linear regression. Such statistical methods are called *robust*, since the estimators can withstand (are only minimally effected by) outliers in data. We will present a high level overview of work on outliers, and then discuss how they relate to robust decision trees.

Figure 1 shows a database of phone calls made in Belgium. Except for the six years from 1964 – 1969 it's a pretty dull dataset. The six outliers were caused by a mistake in recording the data – during the six year period the total number of minutes spent on the phone was recorded rather than the total number of calls. We can see that the effect of the outliers on linear regression is disastrous. Linear regression attempts to fit a line to a set of points so as to minimize the sum-squared residuals (errors):

$$\sum_p (y_p - \hat{y}_p)^2 \ ,$$

where p indexes each training point, and $\hat{y}_p$ indicates the value at x_p predicted by the linear model. A simple algebraic procedure finds the line minimizing this criterion. Outliers are especially problematic because of the squared error objective function.

There are two main ways that statisticians have addressed the outlier problem: accommodation and identification/rejection. We'll first discuss an example of the accommodation method, then the rejection method. Figure 1 shows the result of "L1" regression, which is just like least squares regression except that the objective function is changed to $\sum_p |y_p - \hat{y}_p|$, which de-emphasizes points far away from the fitted line. The accommodation methods, due to Huber (1981), involve modifying the objective function (usually sum-squared error) to down-weight the effect of outliers. Huber's proposal was the M-estimate: $\sum_p \rho(y_p - \hat{y}_p)/\sigma$, where ρ is some function and σ is an unimportant constant scale factor. When ρ is the absolute value function we get L1 regression.

Rather than defining an objective function that lessens the impact of outliers while still allowing them to influence the fitted model, we may instead simply try to identify outliers directly and remove them from consideration. Another way to say this is that we may restrict our attention only to those training records that seem "normal." The least trimmed squares (LTS) method of Rousseeuw & Leroy (1987) uses this idea – rather than minimizing the sum of squared errors, it instead finds the line with minimal trimmed sum of squared errors, which corresponds to minimizing the errors on those points that seem most normal. (When a set of numbers is trimmed, the highest and lowest few numbers are removed.) Figure 1 shows that the LTS regression fit the veridical data best.

Another approach to identification is not through the residuals on each training record but rather on a measure of the effect a record has on the learned model. Points with high *leverage*, points with disproportionately high effect on the fitted model, are identified and removed from training data in order to better fit the remaining points.

Commonly, outliers are thought of as shown in Figure 1 – points that are far away from the rest of the data – but this needn't be the case. In a classification task, for example, a point of class A might be an outlier if found in the middle of a dense cluster of points from class B. The identification of outliers in categorical (nominal, unordered) data has not been addressed in statistics. It is this problem of outliers in categorical data that the present paper addresses. The only idea that seems to be easily portable from linear regression to classification trees (or decision trees) is the idea of leverage – we will next explore this idea in the context of C4.5.

C4.5 and Pruning: Model Selection

C4.5 (Quinlan 1993) is a decision tree induction program. A decision tree predicts unknown field values by asking a series of yes/no or multiple-outcome questions. Each question is of the form "Is the value of field f equal to k?" or "Is the value of field f less than x." Each question in the series is chosen based on the answers to previous questions. After some number of questions have been answered, the value of the unknown field is predicted.

One problem with decision trees (and with all non-parametric statistical methods and learning methods) is that they can simply memorize a database given to them as training data. Generally, memorizing the database is not a useful activity (we have the database already!) so instead we want to try to learn patterns that *generalize* well to future data. Endeavors to solve this problem usually rely on the application of Occam's razor, which can be paraphrased as "keep it simple." *I.E.*, if several decision trees are roughly consistent with a database, we should choose the simplest one if we hope to generalize well. There are also statistical arguments phrasing this as the *bias-variance* problem (Breiman, Friedman, Olshen & Stone 1984, Geman, Bienenstock & Doursat 1992), but we will not pursue this further.

After inducing a decision tree from a database we often are left with a complex tree that fits our training database well. We then *prune* it to make it simpler. This usually causes it to perform more poorly on the training data, but better on data held out as a test sample. Given a complex decision tree with n nodes, we need some procedure for choosing one of the many subtrees that we may arrive at by pruning. C4.5's procedure for this involves repeatedly asking "should this node be made into a leaf?" at each node. The question is answered by estimating the error of the node given the current tree, then estimating the error if the node were pruned to a leaf. If the resulting error is smaller, the node is pruned. After repeatedly pruning or not pruning nodes we are left with the final tree.

C4.5's error estimation method implicitly weighs the number of records that would be misclassified if a given current node were pruned against the reduction in tree size resulting from pruning the node. If only a few points will be misclassified, but many nodes can be saved, C4.5 will prune the node. Thus, after pruning, the set of records misclassified by C4.5's tree will approximate the set of points with high leverage – those few points which caused a large tree structure to be created so that they would be classified correctly.

Table 1: The ROBUST-C4.5 algorithm repeatedly runs C4.5 and removes those records it misclassifies from the training database until all records in the reduced training set are correctly classified.

```
ROBUSTC45(TrainingData)
  repeat {
    T <- C45BuildTree(TrainingData)
    T <- C45PruneTree(T)
    foreach record in TrainingData
       if T misclassifies Record then
          remove Record from TrainingData
  } until T correctly classifies all
    Records in TrainingData
```

Robust C4.5: Data Selection

Pruning is a method of local data selection, since records are effectively removed from nodes deep in the tree – pruning a node to a leaf has the same effect as removing all instances that were not in the majority class in the subset of the database used to build that node's subtree. This begs the question: if the records are *locally* un-informative or harmful, then why should we suspect that they helped the algorithm to discover patterns higher in the tree – that is, more globally? We could think of no good answer to this question, and so developed the ROBUST-C4.5 algorithm described in Table 1.

A novelty in our robust approach is to *rebuild* the tree using the reduced training set. The reduced training set is defined to the original training set minus the

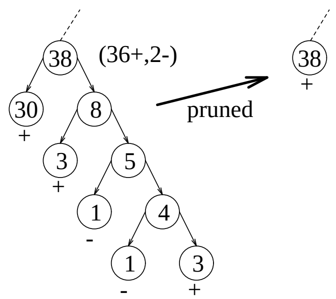

Figure 2: Leverage in a decision tree: the two points caused the subtree to be built. When they are removed, the tree size shrinks by 8 nodes.

instances which the pruned tree now classifies incorrectly (the "confusing" instances). While retraining may seem odd, it is in fact just an extension of the assumptions underlying pruning. By pruning the tree we essentially assume that these confusing instances are locally not useful. Retraining merely takes this assumption a step further by completely removing these instances from the training set. This makes explicit the assumption that locally un-informative or harmful records are globally un-informative as well. The regularization algorithm continues pruning and retraining until no further pruning can be done. Though common in regression in the guise of *robust* (Huber 1977) or *resistant* (Hastie & Tibshirani 1990, Chapter 9) fitting, in the context of classification this appears to be novel.

Relating this to the previous discussion on robust statistics, in the context of decision trees we may identify a set of points with high leverage by examining the difference in number of nodes between a tree built with and without the set of points (Figure 2). This difference is estimated by starting with a pure tree built using all the points and then pruning. The training instances that it now classifies incorrectly are the high leverage points that were removed. However, the removal of the points by pruning was only approximate—the obvious step is to remove the points from the training set and retrain.

Experiments

Because of the arguments above, we hypothesized that Robust-C4.5 would be a worthy competitor to C4.5 always resulting in smaller trees and increasing accuracy on domains with many outliers. Also, since the definition of a robust statistic requires small changes in the statistic for small changes in the data, the standard deviation of the accuracy of Robust-C4.5 should be smaller than C4.5. Below we first describe the experimental methodology and then present and discuss the results.

Method

To test the hypotheses, we compared Robust-C4.5 and C4.5 on 21 databases selected from the UCI and Statlog database repositories (Murphy & Aha 1994, Michie, Spiegelhalter & Taylor 1994). The databases are all stored and processed as a single relation (table). The databases were selected with no particular agenda in mind other than convenience. To compare Robust-C4.5 and C4.5 on a given database we used ten-fold cross-validation. At a computational expense, this method gives one a more reliable estimate of the accuracy of a learning algorithm than a single run on a held-out test set. Cross-validation involves randomly partitioning the database into ten disjoint databases, then providing each algorithm with nine as training data and using the remaining records as test cases. This process is repeated ten times using the different possible test sets.

For purposes of testing the hypotheses, the important statistics to gather for each run of each algorithm on each database are: the mean of the accuracies from the ten runs, the mean of the tree sizes, and the standard deviation of the accuracy among the ten cross-validation runs.

Results

Table 2 presents the results from the experiments described above. For each dataset and algorithm, we report the average of the ten cross-validation accuracies, their standard deviation, and the average of the ten tree sizes. At the bottom, each column's average is reported.

There are several interesting things to point out in the table. Note that Robust-C4.5's average accuracy is slightly higher than C4.5, which weakly verifies our assumption that Robust-C4.5 will have higher accuracy than C4.5. Note that since Robust-C4.5 specializes in the removal of outliers, it might perform badly on complex datasets where there is no noise, but some patterns are under-represented. In such settings we might expect Robust-C4.5 to degrade performance by throwing out perfectly good data (the problem that worried Bernoulli over two hundred years ago), and this was the case. Using a paired t test, we found that the accuracies of Robust-C4.5 and C4.5 were statistically indistinguishable (at the 95% level) on all datasets but four. Robust-C4.5 performed better than C4.5 on "labor-neg" (Canadian labor negotiation), and worse on Tic-Tac-Toe and "segment" (an image segmentation problem). Although we cannot make any statements about the probability of human error in manually classifying the image segments for the "segment" database, the Tic-Tac-Toe database is noise-free and fairly difficult for decision trees to learn (note the average tree size of 133), so it fits our "hard-but-error-free" profile perfectly.

Another interesting pattern in the data is the average tree size. Robust-C4.5 manages to produce

Table 2: Accuracies for C4.5 and RC4.5 on several databases.

Dataset	Accuracy Mean C4.5	Accuracy Mean RC4.5	Accuracy StdDev C4.5	Accuracy StdDev RC4.5	Tree Size C4.5	Tree Size RC4.5
australian	85.53	85.81	3.8	3.8	48.1	32.9
breast-cancer	74.49	74.13	5.2	5.3	10.8	11.5
breast	94.42	94.28	2.8	2.7	20.2	12.6
chess	99.45	99.45	0.4	0.4	55.0	55.0
cleve	73.54	74.54	7.1	6.5	43.5	28.4
crx	84.80	85.10	4.5	4.7	53.3	32.6
diabetes	72.76	74.71	6.1	5.9	127.8	38.0
german	73.10	75.10	3.6	3.6	149.6	30.8
glass	67.78	70.10	10.8	10.2	51.6	37.8
heart	74.10	73.36	10.5	6.0	35.8	21.8
hepatitis	76.20	77.95	7.6	6.6	15.4	10.4
horse-colic	84.51	85.59	4.7	4.5	18.4	16.2
hypothyroid	99.10	99.50	0.7	0.6	15.0	9.6
iris	94.66	92.67	5.3	7.3	7.2	7.2
labor-neg	77.68	84.67	17.5	14.7	5.9	5.6
lymphography	76.91	75.57	11.3	10.9	26.6	20.1
segment	96.35	95.87	1.3	0.8	79.8	71.6
sick-euthyroid	97.67	97.60	0.8	0.9	30.0	23.8
tic-tac-toe	86.23	83.20	4.6	4.4	133.0	109
vote	94.73	94.73	2.1	2.1	14.5	14.5
vote1	88.71	88.71	5.9	5.9	28.6	26.8
AVERAGE	84.42	84.88	5.5	5.1	46.2	32.9

trees that are 29% smaller than C4.5's trees, while maintaining the same level of accuracy. In many cases Robust-C4.5 can dramatically reduce tree size — note the "diabetes" (Pima Indian diabetes) and "german" (accept/reject loan applications) datasets, where tree size was reduced by 70% and 79% respectively. The "german" dataset probably resembles most closely the types of tasks common in KDD: given features such as salary, savings, type of loan and other personal information, predict whether the loan should be granted or denied. Since one of the goals of KDD is to present understandable patterns to a data analyst, Robust-C4.5 should be a worthwhile additional tool.

Regarding the robustness of Robust-C4.5 note that the average standard deviation is indeed smaller than C4.5 by about .4. Looking at each database, the standard deviation of Robust-C4.5 was smaller than C4.5 on eleven, the same on six, and higher on four. This supports the last hypothesis and indicates that by removing outliers, the Robust-C4.5 algorithm is able to learn more stable representations of the hidden patterns in the data than C4.5.

Hastie (1994) suggested that the success of Robust-C4.5 was simply due to the fact that C4.5 does not prune enough, therefore the advantage from Robust-C4.5 is simply due to the fact that after rebuilding the tree, the pruning algorithm gets another shot at reducing the size further. To test this we ran two other full experiments like the one reported above with different settings for C4.5's pruning parameter, and found qualitatively the same results as in our original experiment. Thus the superior performance of Robust-C4.5 does not seem to be due solely to under-pruning on C4.5's part.

Related Work

Robust-C4.5 is based on our earlier work in decision trees (John 1995), which also rejected misclassified records and retrained. Regarding robust methods and outlier rejection, Huber (1977) states "I am inclined to ... prefer technical expertise to any 'statistical' criterion for straight outlier rejection." We are guilty of this sin in our work, but Guyon, Boser & Vapnik (1993) have proposed an interesting method for making use of human expertise in outlier removal to "clean" a dataset. Similarly, rather than rejecting suspected outliers outright, Robust-C4.5 could as well interview an expert, presenting the suspicious database records and asking for an opinion as to whether the record should be removed or down-weighted. Barnett & Lewis (1993) give an interesting and entertaining survey of work in outlier removal in statistics. Venables & Ripley (1994, ch. 8) present a concise review of robust statistics, accompanied by S-PLUS code.

Another way of looking at robust methods (and the identification approach in particular) is as a data selection method. The selection of data for training is an ubiquitous problem in KDD. Many methods are *passive*, accepting data from a training sample in some random order or all at once as a set. Other methods

(Aha 1991) actively accept or reject training records from a temporal sequence presented to them. The difference between outlier removal and active learning (as commonly presented) is somewhat akin to the difference between forward and backward feature subset selection (John, Kohavi & Pfleger 1994) or forward and backward (construction/pruning) search methods over neural net or decision tree architectures. In general, unless time is a very limited resource, the best results are achieved by starting "big" and then shrinking (Breiman et al. 1984). This should apply to pattern selection as well, and thus we suspect backward pattern selection (*i.e.*, outlier removal) will give greater performance than forward selection. Cohn (1994) discusses yet more "active" data selection methods that might perform even better, but they require the availability of a domain expert.

Since Robust-C4.5 just runs C4.5 many times on similar databases, the approach could benefit from an incremental tree-updating scheme such as that proposed in Utgoff (1994) – rather than removing the outliers from the database and retraining on the entire remainder, we could instead simply incrementally remove the outliers from the tree itself, never having to fully retrain.

Conclusion

We have presented a learning algorithm based on ideas drawn from robust statistics. Rather than solely being concerned with the structure, complexity and accuracy of the patterns it has learned from the database, our method also actively screens the training database in an attempt to ignore erroneous records. Such robust methods seem well-suited to learning patterns in databases which are likely to have many errors – by ignoring the errors the produced patterns can be more accurate and far simpler than patterns produced by comparable algorithms which are forced to pay attention to all records. Our experiments on 21 databases confirm this claim, showing that on datasets with characteristics typical of KDD tasks, our Robust-C4.5 can learn trees with slightly higher accuracy and up to 70% fewer nodes C4.5's trees.

Acknowledgements

This research was supported under an NSF Graduate Research Fellowship. We would like to thank Jerry Friedman, Nils Nilsson and Pat Langley for helpful discussions.

References

Aha, D. W. (1991), "Instance-based learning algorithms", *Machine Learning* **6**(1), 37–66.

Barnett, V. & Lewis, T. (1993), *Outliers in Statistical Data*, third edition, John Wiley & Sons, New York.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984), *Classification and Regression Trees*, Chapman & Hall, New York.

Cohn, D. (1994), "Improving generalization with active learning", *Machine Learning* **15**(2), 201–221.

Geman, S., Bienenstock, E. & Doursat, R. (1992), "Neural networks and the bias/variance dilemma", *Neural Computation* **4**, 1–58.

Guyon, I., Boser, B. & Vapnik, V. (1993), Automatic capacity tuning of very large VC-dimension classifiers, *in* S. J. Hanson, J. Cowan & C. L. Giles, eds, "Advances in Neural Information Processing Systems", Vol. 5, Morgan Kaufmann, pp. 147–154.

Hastie, T. (1994), Personal Communication.

Hastie, T. J. & Tibshirani, R. J. (1990), *Generalized Additive Models*, Chapman and Hall.

Hirsh, H. & Cohen, W., eds (1994), *Machine Learning: Proceedings of the Eleventh International Conference*, Morgan Kaufmann.

Huber, P. (1981), *Robust Statistics*, Wiley, New York.

Huber, P. J. (1977), *Robust Statistical Procedures*, Society for Industrial and Applied Mathematics, Pittsburgh, PA.

John, G. H. (1995), Robust linear discriminant trees, *in* "Fifth International Workshop on Artificial Intelligence and Statistics", Ft. Lauderdale, FL, pp. 285–291.

John, G., Kohavi, R. & Pfleger, K. (1994), Irrelevant features and the subset selection problem, *in* Hirsh & Cohen (1994), pp. 121–129.

Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (1994), *Machine Learning, Neural and Statistical Classification*, Prentice Hall.

Murphy, P. M. & Aha, D. W. (1994), "UCI repository of machine learning databases", Available by anonymous ftp to `ics.uci.edu` in the `pub/machine-learning-databases` directory.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Rousseeuw, P. J. & Leroy, A. M. (1987), *Robust Regression and Outlier Detection*, John Wiley and Sons, New York.

Ullman, J. D. (1988), *Principles of Database and Knowledge-Base Systems: Volume 1: Classical Database Systems*, Addison-Wesley.

Utgoff, P. (1994), An improved algorithm for incremental induction of decision trees, *in* Hirsh & Cohen (1994), pp. 318–326.

Venables, W. N. & Ripley, B. D. (1994), *Modern Applied Statistics with S-Plus*, Springer-Verlag, New York.

Weisberg, S. (1985), *Applied Linear Regression*, John Wiley & Sons.