Characterizing Model Errors and Differences

Stephen D. Bay SBAY@ICS.UCI.EDU
Michael J. Pazzani PAZZANI@ICS.UCI.EDU
Department of Information and Computer Science, University of California, Irvine, CA 92697, USA

Abstract

A critical component of applying machine learning algorithms is evaluating the performance of the models induced and using the evaluation to guide further development. Traditionally the most common evaluation metric is error or loss, however this provides very little information for the designer to use when constructing a system. We argue that an evaluation method should provide detailed feedback on the performance of an algorithm and that this feedback should be in the language of the problem: Our goal is to characterize model errors or the differences between models in the feature space. We provide a framework for this that allows different algorithms to be used as the discovery engine and we consider two approaches: (1) a classification strategy where we use a standard rule learner such as C5; (2) a descriptive paradigm where we use a new discovery algorithm: a contrast set miner. We show that C5 suffers from several problems that make it unsuitable for this task.

1. Introduction

A fundamental problem in machine learning is understanding the conditions for which a learning algorithm works well. Understanding an algorithm's strengths and weaknesses and being able to compare algorithms with each other are necessary for designers to develop (or select) learning algorithms for a specific problem. Generally, one can attempt to analyze and understand algorithms either theoretically or empirically.

Theoretical analyses of machine learning algorithms have usually resulted in weak performance guarantees that are not much use to a practitioner. Algorithms are typically proven to be *asymptotically consistent* (i.e. will achieve the Bayes optimal error rate given enough training examples) or that the algorithm can be used to *PAC* (Probably Approximately Correct) learn a given concept (i.e., the algorithm will achieve bounded error with high probability) (Valiant, 1984). Another approach is to analyze average case behavior under specific distributional assumptions, such as learning conjunctive or m-of-n concepts (Langley & Iba, 1993; Langley & Sage, 1999; Pazzani & Sarrett, 1992). Although these analyses are useful in understanding the general behavior of an algorithm, they are unable to provide guidance to the designer in the form of specific predictions of an algorithm's performance with a given problem. Thus most researchers and practitioners resort to empirical evaluation to understand learning algorithms.

The most common method of empirically evaluating a classifier is to examine its error, or more generally loss, and many comparisons of algorithms use only this metric. Loss can easily be estimated by using a test set or cross-validation. However because loss is a single number, it does not reveal much about the algorithm except gross performance on the domain. If loss was consistent over all areas, this would be sufficient to characterize performance, however this is clearly not the case: some classes are more difficult than others especially in applications such as fraud detection (Fawcett & Provost, 1997). Even within a class, certain instances will be more or less difficult. Other evaluation methods include confusion matrices that cross-tabulate the true classes of instances with the predicted class, ROC curves (Bradley, 1997; Provost, Fawcett, & Kohavi, 1998) which evaluate the algorithm's performance in terms of true and false positive rates, and the Kappa (κ) statistic (Cohen, 1960) which is a summary measure of agreement between two models.

These approaches leave the practitioner with little understanding of how the algorithm and problem domain interact. They cannot answer questions like "On which types of examples is my classifier most and least accurate?", or "What are the differences between these two classifiers given that they have the same accuracy?"

To date, there has been little work on understanding the areas a classifier is performing well or poorly and presenting this information in a comprehensible manner. We address this deficiency by presenting procedures which characterize model errors and differences. We focus on producing results that are interpretable, comprehensible and provide insight into the domain.

Our goal is to automatically characterize model errors or model differences *in the feature space of the problem.* Specifically, we seek conjunctions of attributes and values that represent areas of good or poor performance (or agreement/disagreement between two models). For example, in the Adult Census domain (Blake & Merz, 1998) the goal is to predict if the salary of a person is greater or less than \$50,000 from demographic variables. When we evaluate models produced for this task, we would like to obtain rules such as:

> Classifier MC4 (Kohavi, Sommerfield, & Dougherty, 1997) is 21% less accurate than average on people who are between 45 and 55 years of age, are high school graduates, and are married. This represents 115 misclassified instances.

Alternatively, if we are comparing two classifiers:

> MC4 and naive Bayes are 9% less likely to agree than average on people who have Masters degrees and are married. This represents 50 instances with different predictions.

Knowledge of model errors or differences could be used in many ways to help guide the iterative process of constructing a machine learning system. For example, it could be used to:

1. *Identify areas in the feature space where the model should not be trusted.* Examples which fall into poorly performing areas can be flagged for exception and handled by a human.

2. *Give insight into the domain by identifying hard areas.* For example, knowing that we are less accurate on people who are older than 45 and have a high school education with the naive Bayesian classifier suggests that perhaps education level and age interact and that the independence assumption is not appropriate.

3. *Indicate where the model should be improved or repaired.* For example, if our credit processing application makes many errors on people who are X and Y then we could develop a special purpose model for those applicants, or we could collect more data in those areas to improve performance.

4. *Understand how changes to an algorithm affect its performance in a problem domain.* For example, we could answer "How does relaxing the independence assumption in the naive Bayesian classifier by allowing dependencies affect classification?"

Many classifiers can produce confidence ratings on their predictions. Our work differs from this in that we generate a description of the model's performance, not a confidence value for a specific instance (although we could also do this). Additionally, most methods that generate confidence ratings are tied to the learning algorithm and cannot easily be transfered.

In the next section we discuss related work. In Section 3 we present a framework for solving this problem and in Section 4 we describe two different discovery engines that can plug into the framework. In Section 5 we experiment on data to evaluate these approaches. Finally, we present concluding remarks in Section 6.

2. Related Work

Many researchers have tried to learn when a classification algorithm is appropriate for a problem domain based on characteristics of the data set (Aha, 1992; Brazdil, Gama, & Henery, 1994; Rendell & Cho, 1990; Michie, Spiegelhalter, & Taylor, 1994). The basic idea is to use properties such as the number of features, number of classes, or the number of instances to learn by inspection or through automated analysis when an algorithm is appropriate. For example, Aha (1992) used a rule learner to automatically derive results such as the following.

> If (# training instances < 737) AND (# prototypes per class > 5.5) AND (# relevant attributes > 8.5) Then IB1 will perform better than C4

These works characterize performance in terms of data set properties. In contrast, we take an orthogonal approach where our goal is to characterize performance in terms of the feature space.

Another related area is *multiple models* work (Dietterich, 1997) which examines methods for combining several classifiers to obtain better performance than could be achieved individually. Many multiple model approaches have attempted to learn where a model is appropriate in the feature space and to select the best model for each instance based on its location.

For example, Merz (1995) partitions the feature space according to existing decision regions in the component classifiers. He then attempts to select the best classifier for each region. Wolpert (1992) introduced a framework called *stacked generalization* where the goal was to have a classifier learn which of the base classifiers to listen to on the basis of past predictions and location of the test instance in the feature space. Brodley (1995) partitions the feature space recursively and selects the best classifier for each region.

The main difficulty with using this work to characterize models is that the focus is on prediction not description. Thus often the learning is implicit and cannot be easily shown to the end user. For example, consider trying to understand the decision regions of a neural network used for model selection.

3. Framework for Characterizing Models

The basic approach we use is to augment a test set of data (data not used to train the classifiers) with information concerning whether or not the classifier made a correct prediction, or alternatively, if the models differed from each other. We then use an algorithm, such as C5 or STUCCO (Bay & Pazzani, 1999a, 1999b), to determine the conditions when errors or differences occur. Table 1 describes the MErr procedure for characterizing model errors, and Table 2 describes the MDiff procedure for characterizing model differences.

This process can be viewed as a simple form of meta-learning (Wolpert, 1992) where the models that we seek to characterize are the base learners and the goal is to generate comprehensible descriptions of model errors or differences (instead of accurate predictions).

Table 3 shows an example of MErr's augmented data. The first four columns represent the original learning problem for which salary is the target variable. The last column is the augmentation and indicates if the classifier agreed with the true class labels. For MDiff, we would include extra columns for each classifier's salary prediction.

Table 1. Finding Model Errors (MErr)

Procedure MErr
Input: D test data, C classifier
Output: R rules describing errors of C
Begin
1. predict classes for all instances in D with C
2. foreach element of D
3. if the prediction matches the true class
4. then augment the instance with $\mathsf{agree} = 1$
5. else augment the instance with $\mathsf{agree} = 0$
6. discover rules (R) which describe when agree is 0 or 1.
End

Table 2. Finding Model Differences (MDiff)

Procedure MDiff
Input: D test data, C_1 classifier 1, C_2 classifier 2
Output: R rules describing when C_1 and C_2 (dis)agree
Begin
1. predict classes for all instances in D with C_1, C_2
2. foreach element of D
3. augment the data vector with class predictions
4. if the predictions are the same
5. then augment the instance with $\mathsf{agree} = 1$
6. else augment the instance with $\mathsf{agree} = 0$
7. discover rules (R) which describe when agree is 0 or 1.
End

Table 3. MErr Augmented Data

Age	Sex	Occupation	Salary	Agree
34	M	Tech-Support	$> \$50K$	0
49	F	Prof-Specialty	$> \$50K$	1
24	F	Exec-Managerial	$\leq \$50K$	0
57	M	Admin-Clerical	$\leq \$50K$	1

4. Discovery Algorithms

In this section we describe two algorithms that could be used for discovery in our framework.

4.1 A Contrast Set Miner: STUCCO

We briefly review the STUCCO algorithm for mining contrast sets. The reader is directed to Bay and Pazzani (1999a, 1999b) for a more detailed description.

STUCCO is a complete mining algorithm that searches for contrast sets, conjunctions of attribute-value pairs, that have substantially different probabilities across several distributions or groups. Formally, the goal is to find contrast sets (cset) where

$$P(\mathsf{cset} \mid G_1) \neq P(\mathsf{cset} \mid G_2) \quad (1)$$

$$|\mathsf{support}(\mathsf{cset}, G_1) - \mathsf{support}(\mathsf{cset}, G_2)| \geq \delta \quad (2)$$

The *support* of a contrast set with respect to a group G is the percentage of examples in G where the contrast set occurs; δ is a user defined threshold called the *minimum support difference.* Equation 1 is a statistical significance criterion; it ensures that the contrast set represents a true difference between the groups and is not caused by fluctuations in random sampling. We test this by using a chi-square test which must reject independence of group membership and the contrast set (i.e. the null hypothesis is $P(\mathsf{cset} \mid G_1) = P(\mathsf{cset} \mid G_2)$). Equation 2 is an effect size criterion; it measures the magnitude of the difference and ensures that the effect is big enough to be important.

These two criteria map directly onto our problem. For both MErr and MDiff the groups are the sets of instances where agree=0 or agree=1. The criteria are equivalent to finding sets where the agreement differs from the average; i.e. if A and N are the set of examples where agree=0 and agree=1, then STUCCO searches for sets with agreement different from $|A|/(|A| + |N|)$.

STUCCO takes a two stage approach to mining. In the first stage, STUCCO searches for all possible contrast sets that meet the criteria. In the second stage, STUCCO summarizes the mined results to present only a small set of rules.

STUCCO organizes the search for contrast sets using set-enumeration trees (Rymon, 1992) to ensure that every node is visited only once or not at all if it can be pruned. Figure 1 shows an example set-enumeration tree for four attribute-value pairs. STUCCO searches this tree using breadth-first search; it starts with the most general terms, i.e. those contrast sets with a single attribute-value pair such as gender = female or occupation = Tech-Support. These sets are the easiest to understand and will have the largest support. It then progresses to more complicated sets that involve conjunctions of terms, for example, gender = female $\wedge$ occupation = Tech-Support.

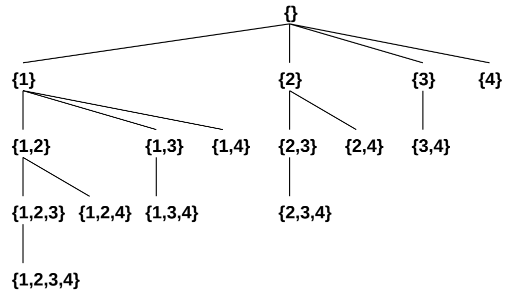

Figure 1. Example search tree for four attribute-values pairs {1,2,3,4}.

At each level, STUCCO scans the database to count the support of all nodes for each group. It examines the counts to determine which nodes meet the criteria and which nodes should be pruned.

STUCCO's pruning is based on obtaining tight bounds on the maximum support difference in any branch of the tree, maximum value of the χ^2 statistic used to test Equation 1, and the maximum difference between any descendent and its ancestor. During search, STUCCO dynamically reorders attribute-value pairs to maximize the effect of pruning.

STUCCO also explicitly controls the search error to limit false discoveries. STUCCO keeps careful track of the number of statistical tests made to verify Equation 1 and adjusts the α level for individual tests to control the overall Type I error rate.

In the second stage, STUCCO summarizes the mined results by showing the most general contrast sets first, those involving a single term, and then only showing more complicated conjunctions if they are surprising based on the previously shown sets. For example, we might start by showing the contrast sets gender = female, occupation = Tech-Support, and salary > \$50K. STUCCO would then show more complicated sets such as gender = female $\wedge$ occupation = Tech-Support followed by gender = female $\wedge$ occupation = Tech-Support $\wedge$ salary > \$50K. The conjunctions are only shown if their supports could not be predicted from the subsets using a log-linear model (Everitt, 1992).

4.2 A Rule Learner: C5

Another approach to distinguishing two groups from each other is to use a rule learner or decision tree to learn a classification strategy (i.e. the classes are agree=0 and agree=1). In this paper, we use C5 which is an updated version of C4.5 (Quinlan, 1993). It is a workhorse of the machine learning community and is a gold standard to which many new algorithms are compared.

C5 performs greedy heuristic search to develop a decision tree. Starting at the root of the tree, C5 selects an attribute-value test to partition the feature space. Each partition is represented by a child node and is then recursively divided with more tests. The tests are chosen to create child nodes which tend to be mainly of one class. After finding the tree C5 can then convert it to rules and remove unnecessary terms.

Note that if we augment the data with class predictions (Table 2, Line 3), C5 will only obtain rules that say *if the prediction from classifier 1 is different from classifier 2 then agree = 0.* Because of the greedy search strategy, no further rules will be generated since rules of this form can cover the entire data set.

One might think that a quick fix is to perform two runs of C5: once with the first algorithm's prediction and again using only the second algorithm's prediction. While this may solve the immediate problem of obtaining rules that reference class predictions, it does not address the cause: Once C5 finds a rule that covers a set of examples it does not look for alternative coverings. In practice, C5 almost always considered class predictions informative for finding rules where agree=0 and included them as tests, thus we now have the opposite problem of trying to find rules which do not include class predictions. Clearly, we can always remove some variables from the analysis to force C5 to generate rules without them, however this process of running C5 on different subsets of features can become very cumbersome.

While C5 is an effective classification tool, we believe it has several disadvantages in a discovery context: First, rule learners and decision trees are not complete. They use heuristics to prune large portions of the search space and thus they may miss important rules. Second, decision trees and rule learners are unstable classification algorithms (Breiman, 1996). Small changes in the learning set can result in wildly different trees or rule sets. Stability is essential for domain users to accept the discovered knowledge. For example, in a manufacturing domain Turney (1995) found that the engineers "were disturbed when different batches of data from the same process result in radically different decision trees." The engineers lost confidence even though the trees had high predictive accuracy. Finally, it is difficult to specify useful criteria such as minimum support or an acceptable false positive rate in the classification framework.

5. Evaluation

Evaluating descriptive tasks is difficult because it is somewhat subjective and the factors that make a rule set useful may differ from person to person. However, we believe that there are several quantifiable characteristics that are important to any discovered rule set. First, the individual rules should affect a substantial number of instances. Rules that affect only a small number of instances are not interesting because of their limited impact. Second, the rules should be short as short rules are easier to comprehend. Rule length is not the only factor in rule comprehensibility (Pazzani, Mani & Shankle, 1997), but it serves as a useful operational measure (Domingos, 1997). Finally, the rules should be stable to minor variations of the data if users are to accept the results as valid (Turney, 1995).

We evaluated the impact of a rule by examining its *effect size* which we define as the difference in the number of examples on which a model agrees with the true labels (or other model) compared with the average agreement (see the example with MC4 below). *Rule Length* is the number of terms in the conjunction used to identify an area of model agreement or disagreement. We define *stability* as the expected pairwise agreement of rule sets constructed from different data sets drawn from the same distribution where the agreement of two rule sets X and Y is:

$$\mathrm{agreement}(\mathrm{X},\mathrm{Y}) = \frac{|\mathrm{X}\cap\mathrm{Y}|}{|\mathrm{X}\cup\mathrm{Y}|} \tag{3}$$

Agreement is 0 when X and Y have no rules in common and 1 when they are identical.

We evaluated models induced on the Adult Census data set (Blake & Merz, 1998). The classification task is to categorize people with salary greater or less than $50,000 per year based on demographic variables such as age, working class, education, sex, hours worked, capital gain or loss reported, etc. We used the original training set of 32561 instances to develop our classifiers and the test set of 16281 instances to evaluate them.

We used this database because the variables and problem task are understandable and meaningful. Most readers will have some intuition about the types of people who have high salaries and this can provide a sanity check on our discovered rules. We examined three queries to evaluate our system.

1. What types of errors does the decision tree MC4 (Kohavi et al., 1997) commit?
2. What are the differences between the nearest neighbor (1NN) and 5-nearest neighbor (5NN) classifiers?
3. How does relaxing the independence assumption in the naive Bayesian classifier by allowing limited dependencies affect classification? We compare naive Bayes with SuperParent, a Bayesian classifier that allows additional dependencies (Keogh & Pazzani, 1999).

We discretized the data set prior to analysis by examining the variables and their histograms and selecting cutpoints that were meaningful to us.[1] For ease of interpretation, we converted our results for both C5 and STUCCO into English rules with the following format.

> Classifier MC4 is 15% less accurate than average at predicting the salary of people whose occupation is craft-repair and are married. This represents 183 misclassified instances.

By accuracy differences we mean absolute difference: i.e. if the average accuracy was 85% then the above rule identifies a set of people where the accuracy is 85% - 15% = 70%. The 183 misclassified instances is the effect size of the rule and is the number of additional examples that would have been correctly classified if the accuracy was the same as the average.

In our initial experiments with C5, we found that the default options produced poor rule sets: C5 would produce many rules that had very small effect sizes and occasionally would fail to find any rules at all (i.e. C5 would just predict `agree=1` as a default class). In an attempt to mitigate the small effect sizes, we set the `-m` parameter, which sets the minimum number of cases that must follow at least two branches after a split, to maximize the effect size of the resulting rules. We tried values ranging from 2 (the default) to 640 (2, 5, 10, 20,

[1] The naive Bayesian classifier requires that continuous variables are discretized or that a probability model, such as a Gaussian, is fit to the marginal distributions.

40, 80, 160, 320, 640) and we reported the results for the value that produced rules with the highest median effect size without a degenerate rule set (e.g., no rules for agree=0 or agree=1). We also found it necessary to set the misclassification costs to balance the different class sizes. We set the cost of misclassifying an instance inversely proportional to its class frequency. For STUCCO, we set $\delta = 2\%$ to focus on rules with large effect sizes.

Table 4 summarizes the experimental results for our three queries. For stability, we created 20 random samples of size 5000 from the test data and reported the average agreement between all pairs of rule sets. For MDiff we reported the results for STUCCO rules which did not reference class predictions to make the results comparable to C5.

Table 4. Summary of C5 and Stucco (S) rule sets characterizing MC4 errors, comparing 1NN with 5NN and comparing Naive Bayes (NB) with SuperParent (SP).

	MC4 Errors		1NN vs. 5NN		NB vs. SP	
	C5	S	C5	S	C5	S
number of rules	52	143	685	123	46	192
median effect size	44	148	2	64	38	115
average rule size	2.9	2.0	3.8	2.1	2.6	2.6
stability	0.25	0.54	0.05	0.48	0.24	0.40

5.1 Characterizing Decision Tree Errors

We used MErr to characterize the errors that the decision tree, MC4, would make on the Adult data set. MC4 correctly classified 13811 cases and incorrectly classified 2470 cases. Appendix A shows a subset of the discovered rules. The results highlight a number of problems with using C5 as a discovery algorithm.

First, C5 suffers from fragmentation problems caused by the divide and conquer approach to induction. C5 found many rules that affected only a small number of examples. For example,

> MC4 is 6% more accurate than average on people who have a Bachelors degree, are married, work in a professional specialty, reported a capital gain of \$0, and have a salary > \$50K. This represents 13 correctly classified instances.

This occurs because in C5 the tree structure will tend to divide the data into very small sets with high concentrations of instances with agree = 0 or 1.

Second, C5 also missed many obvious rules found by STUCCO. For example, C5 did not find the following two very simple rules:

> MC4 is 26% less accurate than average on people who have a salary > \$50K. This represents 1013 misclassified instances.

> MC4 is 13% less accurate than average on people who are married. This represents 925 misclassified instances.

This occurs because of the heuristic and greedy search to obtain very high purity subspaces. C5 will typically ignore attribute-value sets with smaller accuracy differences but large effect sizes.

STUCCO was more stable than C5 on this query. In Figure 2 we plotted the agreement between all pairs of rule sets for C5 and STUCCO. The x-axis measures agreement, and the y-axis measures $|X \cup Y|$ which is the size of the union. This shows that STUCCO was more stable even though it produced larger rule sets.

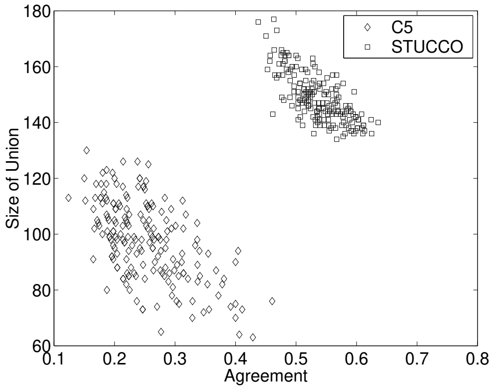

Figure 2. Stability of C5 and STUCCO

5.2 Differences Between 1NN and 5NN

We used MDiff to compare the 1NN and 5NN classifiers. We chose this task because we believed it would be extremely difficult as the two classifiers are very similar: The 5NN classifier can be viewed as a smoothed version of the 1NN classifier. The voting process over the k nearest neighbors serves as a variance reduction technique (Bay, 1999).

There were 14996 test cases where the 1NN and 5NN classifiers agreed and 1285 cases where they disagreed. As we can see from the statistics in Table 2, the fragmentation problem is much worse for C5 on this query. *The median effect size for C5 was only 2, whereas for STUCCO it was 64.* C5 was also very unstable on this dataset producing many long rules (average size 3.8) which did not re-occur with alternative test data sets drawn from the same distribution.

5.3 Relaxing the Naive Bayes Independence Assumption

We used MDiff to analyze how the naive Bayesian classifier changes when we add additional dependencies between features. We compared naive Bayes with SuperParent (Keogh & Pazzani, 1999), an efficient search algorithm for adding dependencies to maximize classification performance. Naive Bayes and SuperParent agreed on 86% of the examples.

As before, C5 still suffers from fragmentation problems and finds rules that affect only a small number of examples. Limited space prevents us from displaying all our results, thus we present a small subset of STUCCO results in Table 5 which shows the differences between NB and SP by occupation.

Table 5. Occupation and Model Differences

occupation	model agreement	effect size
craft-repair	-15%	-307
other-service	+12%	+192
exec-managerial	+3%	+51
prof-specialty	+4%	+75
handlers-cleaners	+11%	+79
machine-op-inspect	+4%	+44
admin-clerical	+5%	+82
farming-fishing	-8%	-38
transport-moving	-18%	-136

From the table we can clearly see how adding dependencies affected occupation categories. In contrast, a typical C5 rule involving occupation is:

> Naive Bayes and SuperParent are 21% less likely to agree than average on people who have some college education, are married, work in transport-moving and reported a capital-gain of $0. This represents 14 instances with different predictions.

C5 does not build its rule set hierarchically; thus we never see the main effects as with STUCCO.

6. Conclusions and Future Work

We presented MErr and MDiff which are simple procedures for characterizing the errors a model makes or the difference between two models in the feature space of the domain. They produce comprehensible descriptions and allow a practitioner insight into the domain using the language of the problem.

Both procedures use a plug-in algorithm such as a rule learner or contrast set miner to generate descriptions of a model's performance. We evaluated MErr and MDiff using C5, a classification algorithm, and STUCCO, a discovery algorithm for finding contrast sets. We found that C5 suffers from several problems that make it unsuitable for this task such as fragmentation, incompleteness, and instability. STUCCO did not suffer from these problems. We believe that MErr and MDiff in conjunction with STUCCO will be effective diagnostic tools for building machine learning systems.

Finally, we see two main areas for future work. First, we believe that it is important to try other algorithms as rule generators. In particular, Maximal Parsimonious Discrimination (Valdes-Perez & Pericliev, 1997) appears promising as it was designed to generate comprehensible class descriptions. Second, there has been little research into understanding how humans interpret rules and rule sets and our work would greatly benefit from further investigation here.

Acknowledgements

This research was funded in part by the National Science Foundation grant IRI-9713990. We thank Eamonn Keogh for providing naive Bayes and SuperParent experimental results, and the anonymous reviewers for many comments that improved this paper.

References

Aha, D. W. (1992). Generalizing from case studies: A case study. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 1–10). San Francisco: Morgan Kaufmann.

Bay, S. D. (1999). Nearest neighbor classification from multiple feature subsets. *Intelligent Data Analysis*, *3*, 191–209.

Bay, S. D. & Pazzani, M. J. (1999a). Detecting change in categorical data: Mining contrast sets. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: The Association for Computing Machinery.

Bay, S. D. & Pazzani, M. J. (1999b). Detecting group differences: Mining contrast sets. Information and Computer Science, University of California, Irvine.

Blake, C. & Merz, C. J. (1998). UCI repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/].

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*, 1145–1159.

Brazdil, P., Gama, J., & Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. In *Proceedings of the Sixth European Conference on Machine Learning*. Berlin: Springer Verlag

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Brodley, C. E. (1995). Recursive automatic bias selection for classifier construction. *Machine Learning*, *20*, 63–94.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*, 37–46.

Dietterich, T. G. (1997). Machine learning research: Four current directions. *AI Magazine*, *18*, 97–136.

Domingos, P. (1997). Knowledge acquisition from examples via multiple models. In *Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

Table 6. STUCCO and C5 Results for MC4 Errors.

STUCCO	Acc. Diff.	Effect Size
salary > $50K ∧ cap-gain = $0	-35%	-1059
Married ∧ Male	-13%	-836
salary > $50K ∧ Husband	-21%	-623
age = (25,35] ∧ salary > $50K	-37%	-290
hours-per-week = (45,60] ∧ Husband	-14%	-244
age = (45,55] ∧ salary > $50K	-21%	-222
Some-college ∧ salary > $50K ∧ cap-loss = $0	-34%	-205
Craft-repair ∧ Husband	-15%	-182
Exec-managerial ∧ Husband	-15%	-154
Bachelors	-5%	-134
age > 55 ∧ Husband	-10%	-123
hours-per-week = (45,60] ∧ salary ≤ $50K ∧ Married	-14%	-105
race = White	-1%	-97
Admin-clerical ∧ Married	-17%	-89
cap-gain = ($3500,$7500] ∧ salary ≤ $50K	-36%	-71
Bachelors ∧ salary ≤ $50K	-4%	-62
Masters ∧ Married	-9%	-50
Never-married ∧ salary ≤ $50K	+15%	+787
Female	+8%	+423
Divorced ∧ salary ≤ $50K	+15%	+301
Divorced	+8%	+176
Admin-clerical	+5%	+84
Masters ∧ salary > $50K ∧ Married	+15%	+59
Machine-op-inspct	+5%	+46

C5	Acc. Diff.	Effect Size
cap-gain = $0 ∧ salary > $50K	-35%	-1059
Bachelors ∧ Married ∧ cap-gain = $0 ∧ salary ≤ $50K	-53%	-223
Never-married ∧ cap-gain = $0 ∧ salary > $50K	-85%	-145
Masters ∧ Married ∧ cap-gain = $0 ∧ salary ≤ $50K	-85%	-107
age = (45,55] ∧ Some-college ∧ Married ∧ salary ≤ $50K	-45%	-62
cap-gain = ($0, $3500] ∧ salary > $50K	-85%	-44
Widowed ∧ cap-gain = $0 ∧ salary > $50K	-85%	-25
Doctorate ∧ Married ∧ cap-gain = $0 ∧ salary ≤ $50K	-85%	-18
Married (Armed Forces spouse) ∧ salary ≤ $50K	-55%	-6
Doctorate ∧ salary > $50K	-2%	-2
HSgrad ∧ cap-gain = $0 ∧ salary ≤ $50K	+15%	+623
age ≤ 25 ∧ Some-college ∧ salary ≤ $50K	+15%	+166
age = (25,35] ∧ Some-college ∧ cap-gain = $0 ∧ salary ≤ $50K	+14%	+96
Grade11 ∧ cap-gain = $0 ∧ salary ≤ $50K	+15%	+89
Separated ∧ salary ≤ $50K	+15%	+72
Grade10 ∧ cap-gain = $0 ∧ salary ≤ $50K	+14%	+59
Farming-fishing ∧ cap-gain = $0 ∧ salary ≤ $50K	+13%	+54
Grade7-8 ∧ cap-gain = $0 ∧ salary ≤ $50K	+15%	+41
Grade9 ∧ cap-gain = $0 ∧ salary ≤ $50K	+15%	+33
Bachelors ∧ Married ∧ Exec-managerial ∧ cap-gain = $0 ∧ salary > $50K	+13%	+31
Married (spouse absent) ∧ salary ≤ $50K	+15%	+27
cap-gain = ($7500,$10000] ∧ salary > $50K	+15%	+25
cap-gain = ($3500,$7500] ∧ salary > $50K	+5%	+11
Bachelors ∧ Other-service ∧ salary ≤ $50K	+15%	+11
Prof-school ∧ salary > $50K	+3%	+5

Everitt, B. S. (1992). *The analysis of contingency tables.* London: Chapman and Hall, second edition.

Fawcett, T. & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery, 1*, 291–316.

Keogh, E. & Pazzani, M. J. (1999). Learning augmented Bayesian classifiers: A comparison of distribution based and classification based approaches. In *Proceedings of the Seventh International Workshop on AI and Statistics* (pp. 225–230). San Francisco: Morgan Kaufmann.

Kohavi, R., Sommerfield, D., & Dougherty, J. (1997). Data mining using MLC++, a machine learning library in C++. *Journal of Artificial Intelligence Tools, 6*, 537–566.

Langley, P. & Iba, W. (1993). Average-case analysis of a nearest neighbor algorithm. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 889–894). San Francisco: Morgan Kaufmann.

Langley, P. & Sage, S. (1999). Tractable average-case analysis of naive Bayesian classifiers. In *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 220–228). San Francisco: Morgan Kaufmann.

Merz, C. J. (1995). Dynamical selection of learning algorithms. In D. Fisher & H. Lenz (Eds.), *Learning from data: artificial intelligence and statistics.* Berlin: Springer Verlag.

Michie, D., Spiegelhalter, D. J., & Taylor, C. (Eds.) (1994). *Machine learning, neural and statistical learning.* West Sussex, England: Ellis Horwood.

Pazzani, M. J., Mani, S., & Shankle, W. R. (1997). Beyond concise and colorful: Learning intelligible rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining.* Menlo Park, CA: AAAI Press.

Pazzani, M. J. & Sarrett, W. (1992). A framework for average case analysis of conjunctive learning algorithms. *Machine Learning, 9*, 349–372.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning.* San Francisco: Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: programs for machine learning.* San Mateo, CA: Morgan Kaufmann.

Rendell, L. & Cho, H. H. (1990). Empirical learning as a function of concept character. *Machine Learning, 5*, 267–298.

Rymon, R. (1992). Search through systematic set enumeration. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning.* San Francisco: Morgan Kaufmann.

Turney, P. (1995). Technical note: Bias and the quantification of stability. *Machine Learning, 20*, 23–33.

Valdes-Perez, R. E., & Pericliev, V. (1997) Maximally parsimonious discrimination: A generic task from linguistic discovery. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence.* Menlo Park, CA: AAAI Press.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM, 27*, 1134–1142.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*, 241–259.

Appendix A: Example Rules

Table 6 shows a subset of rules found by STUCCO and C5 on MC4 errors. After sorting the rules by effect size, we selected every sixth and second rule respectively.