Genetic Rule Extraction Optimizing Brier Score

Ulf Johansson
School of Business and Informatics
University of Borås
Borås, Sweden
ulf.johansson@hb.se

Rikard König
School of Business and Informatics
University of Borås
Borås, Sweden
rikard.konig@hb.se

Lars Niklasson
Informatics Research Centre
University of Skövde
Skövde, Sweden
lars.niklasson@his.se

ABSTRACT

Most highly accurate predictive modeling techniques produce opaque models. When comprehensible models are required, rule extraction is sometimes used to generate a transparent model, based on the opaque. Naturally, the extracted model should be as similar as possible to the opaque. This criterion, called fidelity, is therefore a key part of the optimization function in most rule extracting algorithms. To the best of our knowledge, all existing rule extraction algorithms targeting fidelity use 0/1 fidelity, i.e., maximize the number of identical classifications. In this paper, we suggest and evaluate a rule extraction algorithm utilizing a more informed fidelity criterion. More specifically, the novel algorithm, which is based on genetic programming, minimizes the difference in probability estimates between the extracted and the opaque models, by using the generalized Brier score as fitness function. Experimental results from 26 UCI data sets show that the suggested algorithm obtained considerably higher accuracy and significantly better AUC than both the exact same rule extraction algorithm maximizing 0/1 fidelity, and the standard tree inducer J48. Somewhat surprisingly, rule extraction using the more informed fidelity metric normally resulted in less complex models, making sure that the improved predictive performance was not achieved on the expense of comprehensibility.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Concept Learning*

General Terms

Algorithms

Keywords

Rule extraction, Brier score, Genetic programming

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

1. INTRODUCTION

Predictive classification is the task of learning a target function f mapping each instance $\boldsymbol{x}$ to one of the predefined class labels y. The target attribute y (the class label) is a discrete attribute and restricted to values in a predefined set $\{c_1, c_2, \ldots, c_n\}$. Since a predictive classification model should be able to assign every possible input vector to one of the predefined classes, the model must partition the input space; i.e., the classes have to be non-overlapping and exhaustive.

When using machine learning techniques for predictive classification, the model represents patterns in historical data found by the specific algorithm. More technically, the algorithm uses a set of training instances, each consisting of an input vector $\boldsymbol{x}_i$ and a corresponding target value y_i to learn the function $y = f(\mathbf{x}; \theta)$. During training, the parameter values θ are optimized, based on a score function. When sufficiently trained, the predictive model is able to predict a value y, when presented with a novel (test) instance $\boldsymbol{x}$.

For predictive classification, the most important criterion is accuracy, i.e., the number of misclassifications made by the model when applied to novel data should be as few as possible. Unfortunately, most high-accuracy techniques like *artificial neural networks* (ANNs), *ensembles* or *support vector machines*, produce opaque models. Opaque predictive models make it impossible to trace the logic behind a prediction and, on another level, to assess the potential knowledge represented by the model. Clearly, there are applications where this is unacceptable, making transparent models mandatory. So, when models need to be comprehensible, accuracy is often sacrificed by using simpler, but transparent models; most typically decision trees like C4.5/C5.0 [20] or CART [5]. This tradeoff between predictive performance and interpretability is normally referred to as the *accuracy vs. comprehensibility tradeoff*.

Although the most straightforward way of obtaining transparent models is to build decision trees or induce rule sets directly from the training data, another option is to generate a transparent model based on a corresponding opaque predictive model. This process, which is normally called *rule extraction*, has been used mainly on ANN models; for an introduction and a good survey of traditional methods, see [1]. In this paper, we introduce and evaluate a novel rule extraction algorithm which, to the best of our knowledge, is the first such algorithm trying to explicitly mimic the probability estimations of the opaque model, instead of just the predictions.

2. BACKGROUND

Craven and Shavlik [9] coined the term *representation language* for the language used to describe the opaque model. They also used the expression *extraction strategy* for the process of transforming the opaque model into the new representation language. Representation languages typically used when extracting rules from ANNs include (*if-then) inference rules*, *M-of-N rules*, *fuzzy rules*, *decision trees* and *finite-state automata*. Regarding extraction strategies, there are two fundamentally different approaches; *decompositional* (*open-box* or *white-box*) and *pedagogical* (*black-box*).

Decompositional approaches focus on extracting rules at the level of individual units within a trained ANN. A basic requirement for this category of rule extraction is, consequently, that the computed output from each unit must be mapped into a binary outcome, corresponding to a rule consequent. Each unit can be interpreted as a step function, meaning that the problem is reduced to finding a set of incoming links whose summed weights guarantee that the unit's bias is exceeded regardless of other incoming links. When such a combination of links is found, this is readily translated into a rule, where the output of that unit is a consequent of the inputs. The rules extracted at the individual unit level are then aggregated to form the composite rule set for the ANN as a whole. Two classic open-box rule extraction algorithms are RX [21] and Subset [13].

Pedagogical approaches treat the opaque model as a black box. The core idea is to view rule extraction as a learning task, where the target concept is the function originally learned by the opaque model, see Figure 1 below.

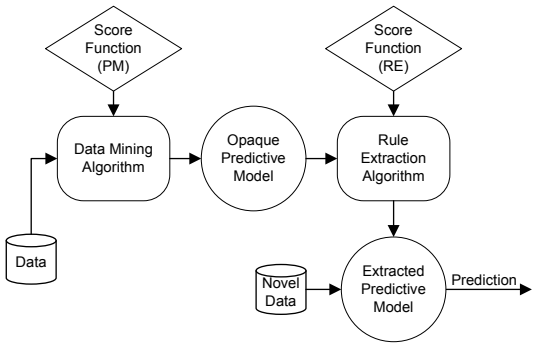

Figure 1: Black-box rule extraction

Black-box rule extraction is, consequently, an instance of predictive modeling, where each input-output pattern consists of the original input vector $\boldsymbol{x}_i$ and the corresponding prediction $f(\boldsymbol{x}_i;\theta)$ from the opaque model. So, black-box rule extraction is simply the task of modeling the function from (original) input attributes to the opaque model predictions. Two typical and well-known black-box rule extraction algorithms are TREPAN [8] and VIA [22].

There are several criteria used for evaluating rule extraction algorithms. In [7] Craven and Shavlik listed five criteria: *accuracy*, *comprehensibility*, *fidelity*, *scalability* and *generality*. Most researchers have evaluated their rule extraction methods using the first three criteria, but according to Craven and Shavlik, scalability and generality have often been ignored. Yet another important criterion, often overlooked but recognized by Towell and Shavlik in [23], is *consistency*. A rule extraction algorithm is consistent if it extracts similar rules every time it is applied to a specific data set. Towell and Shavlik argue that consistency is important since it would be very hard to give any significance to a specific rule set if the extracted rules vary significantly between runs. Since these criteria are central to the evaluation of rule extraction algorithms, they are presented in more detail below.

Accuracy: High accuracy, defined as low error on unseen data, must be considered the primary criterion for all predictive modeling. In the rule extraction domain, this criterion means that an extracted model should have high accuracy when used for prediction. One specific demand is that the extracted model should have higher accuracy than transparent models induced directly from the data set.

Comprehensibility: The comprehensibility of a model is more often than not measured as the size of the extracted representation. This is arguably a simplification, chosen with the motivation that it makes comparisons between techniques fairly straightforward. Strictly speaking, there is not a one-to-one relationship between size and comprehensibility. At the very least, comprehensibility is not linear, making all measures based on size slightly dubious. In practice, comprehensibility is normally regarded as a binary property; a particular model is either comprehensible or not, given the specific situation. All in all, though, the choice to evaluate comprehensibility using the size of the extracted model is the most accepted.

Fidelity: Model fidelity measures, in all existing rule extraction techniques, how similar the output from the extracted model, for each and every instance, is to the output from the opaque model. For classification problems, this metric is consequently an aggregation over all instances, expressing the percentage of instances classified identically to the opaque model by the extracted model. We name this fidelity, which looks only at final predictions, *0/1 fidelity*. It should be noted that low fidelity would indicate that the rule extraction algorithm does not express the relationship found by the opaque model, but instead finds another relationship. Naturally, models produced by rule extraction are normally much simpler than the opaque models they are extracted from, making them incapable of fully representing the complex relationships found by the opaque models. Having said that, extracted models must still have fairly high fidelity and, more importantly, this property must also transfer to test data; i.e., extracted models must still perform similarly to the opaque model, when applied to novel data.

Scalability: Most often scalability measures how the running time of an algorithm varies depending on the input size. For rule extraction, however, the term input size is actually rather complex in itself. For open-box rule extraction algorithms, the most important factor is the size of the ANN. Since black-box techniques do not use the actual architecture, the factors determining the running time are instead typically number of instances, number of attributes and number of classes, but also the parameters of the rule extraction algorithm. It should be noted that scaling is an inherent problem, regarding both running time and comprehensibility, for decompositional methods. The potential size of a rule for a unit with n inputs each having k possible values is k^n, meaning that a straightforward search for possible rules is normally impossible for larger networks. In addition,

the size of rule sets produced by decompositional algorithms tend to be proportional to the network size.

Generality: Clearly, a rule extraction technique is very limited if it is only capable of extracting rules from a specific kind of opaque model, e.g., a single MLP. Ultimately, a technique should instead be able to disregard all factors like underlying architecture and training regime to extract rules from any kind of opaque model. Naturally, such flexibility is impossible to achieve for open-box approaches, while most black-box techniques have this kind of generality, at least implicitly. In addition, it should be possible to extract accurate rules from all kinds of predictive models. This includes, of course, both regression and classification models. Furthermore, the technique must manage both binary classification and multiclass problems. Finally a general rule extraction algorithm should have the capability to extract rules in several different representation languages.

Consistency: The motivation for consistency is that if extracted models differ greatly, it becomes very hard to put faith in one specific model. The most straightforward way of measuring consistency is to ignore the actual rules and instead compare predictions from the rules. This approach makes it rather easy to obtain numerical values, and is also the technique used by Towell and Shavlik in [23]. Another subtlety is the question of whether consistency should be measured between rules extracted from the same opaque model or between entire runs, just using the same data set. Towell and Shavlik used the latter approach, which is reasonable since their rule extraction approach is deterministic in the sense that it will always produce identical rules when rule extracting from a specific trained ANN. Several other techniques, however, may produce different extracted models from one opaque model, making it necessary to also compare models extracted from the same opaque model.

2.1 Related work

There are several black-box rule extraction algorithms using evolutionary approaches. In essence, a population consists of candidate solutions (i.e. extracted models) and the fitness function reflects the chosen optimization criterion. Most often, the fitness is based on fidelity, but may also include terms explicitly targeting improved comprehensibility or accuracy. Comprehensibility is normally handled by using a length penalty, while generalization ability (accuracy) can be targeted by including (possibly artificial) training instances not used when building the opaque model.

Algorithms based on genetic algorithms (GA) typically have to code and decode rule sets into binary or real-valued vectors, which may be slightly awkward. One GA-based rule extraction algorithm is GEX [18] where several subpopulations are evolved on parallel islands and each subpopulation is specialized in searching for rules for one specific class. To allow for rules of different length, a fairly technical encoding of the rules is used. One other cited drawback of GEX [14] is that one instance can be covered by several rules, while it is not guaranteed that at least one rule will be valid.

If instead genetic programming (GP) is used, the individuals are normally coded as S-expressions, typically representing decision trees, making their interpretation quite straightforward. One clear advantage for GP is the possibility to easily choose the representation language, just by specifying the grammar with corresponding function and terminal sets.

We have previously suggested a GP-based rule extraction algorithm called G-REX (Genetic Rule EXtraction) [16]. Since G-REX is a black-box rule extraction algorithm, it is able to extract rules from arbitrary opaque models, including ensembles. For a summary of the G-REX technique and previous studies, see [15].

One key property of G-REX is the ability to use a variety of different representation languages, simply by changing the function and terminal sets. G-REX has previously been used to extract, for instance, decision trees, regression trees, Boolean rules and fuzzy rules. Previous studies have shown that G-REX often finds very compact representations which are surprisingly accurate. As a matter of fact, G-REX models are often only slightly less accurate than the underlying opaque models. Most importantly, G-REX has, in several studies, outperformed techniques like CART and C4.5 regarding both accuracy and comprehensibility.

The scalability of G-REX, as a black-box method, is only dependent of the size of the data set, not factors concerning the underlying representation. Still, using evolutionary algorithms to produce decision trees is much more computationally intensive, and therefore slower, than using standard techniques. In practice, however, building the opaque model, especially when using ensembles, often takes longer time than the actual rule extraction.

Since G-REX uses GP to optimize the extracted representation, consecutive runs, even using the same opaque model, will normally result in slightly different extracted rules. In other words, G-REX is inherently inconsistent. This is also considered to be the main drawback for G-REX (and other rule extractors based on evolutionary algorithms) in the survey reported by Huysmans, Baesens and Vanthienen; see [14]. We do, however, argue that consistency is a highly overvalued criterion, and showed in a previous study [17] that most G-REX trees extracted on a specific data set have higher accuracy than the corresponding CART tree.

3. METHOD

As mentioned in the introduction, the overall purpose of this study is to introduce and evaluate a novel fitness function, based on a more informed fidelity criterion, for G-REX. More specifically, the key idea is to minimize the differences in probability estimates between the models, instead of just the differences in actual predictions. Although this may seem fairly straightforward, a number of design choices have to be made, most importantly how probability estimates should be calculated for different models, and exactly what metric to use for measuring the difference between two probability estimates.

When using G-REX, the available functions and terminals constitute the literals of the representation language. Functions will typically be logical or relational operators while the terminals could be, for instance, input variables or constants. Naturally more complex functions like *M-of-N* could also be used. Function and terminal sets do not, however, fully specify the representation language since it is not always possible to freely combine all functions and terminals. Technically, this is called *typed GP*; i.e., some nodes require their child nodes to have a specific type. Because of this, representation languages must be presented not only as function and terminal sets, but using as a grammar. In this study, the representation language is similar to standard decision trees, but restricted to binary splits. The represen-

tation language used is presented using Backus-Naur form in Figure 2 below.

```
F = { if, ==, <, > }
T = { i1, i2, ..., in, c1, c2, ..., cm, ℜ }

DTree          :-   (if RExp Dtree Dtree) | Class
RExp           :-   (ROp ConI ConC) | (== CatI CatC)
ROp            :-   < | >
CatI           :-   Categorical input variable
ConI           :-   Continuous input variable
CatC           :-   Categorical attribute value
ConC           :-   ℜ
Class          :-   c1| c2| ...| cm
```

Figure 2: Representation language

In this study, rules will be extracted from two kinds of opaque ensemble models; Random forests [4] and ensembles of bagged ANNs. For the actual experimentation, the Weka data mining workbench [24] is used. Since we decided to use Weka, the probability estimates from the two opaque models are produced in a similar way. For both techniques, the probability estimates for each instance are averaged over all M included models (random trees or ANNs) to produce the overall probability estimate for that instance; see equation (1) below where $p^{c_j}(i)$ is the resulting probability estimate of class c_j for instance i and $p_k^{c_j}(i)$ is the probability estimate of model k for class c_j on instance i.

$$p^{c_j}(i) = \frac{1}{M}\sum_{k=1}^{M} p_k^{c_j}(i) \tag{1}$$

In this study, all ANNs use a localist coding; i.e., the number of output units is equal to the number of classes. When predicting, the output unit with the highest output value determines the class associated with that instance. For ANN base classifiers using localist coding, a probability estimate is readily produced by just normalizing all outputs so they sum to unity; see equation (2) below, where $p_k^{c_j}(i)$ is the probability estimate of ANN_k for class c_j on instance i, o_j is the output value for output neuron j representing class c_j and O is the number of output units (classes).

$$p_k^{c_j}(i) = \frac{o_j}{\sum_{n=1}^{O} o_n} \tag{2}$$

For random tree base classifiers, the easiest way to obtain a class probability is to use the *relative frequency*; i.e., the proportion of training instances corresponding to a specific class in each leaf. In equation (3) below, the probability estimate $p_k^{c_j}(i)$, based on relative frequencies, is defined as

$$p_k^{c_j}(i) = \frac{g(i,j,k)}{\sum_{l=1}^{C} g(i,l,k)} \tag{3}$$

where $g(i,j,k)$ gives the number of instances belonging to class j that falls in the same leaf as instance i in tree k and C is the number of classes.

Normally, however, a more refined technique, called *Laplace estimate* or *Laplace correction* is applied. The main reason is that the basic maximum-likelihood estimate does not consider the number of training instances reaching a specific leaf. Intuitively, a leaf containing many training instances is a better estimator of class membership probabilities. With this in mind, the Laplace estimator calculates the estimated probability as:

$$p_k^{c_j}(i) = \frac{1 + g(i,j,k)}{C + \sum_{l=1}^{C} g(i,l,k)} \tag{4}$$

It should be noted that the Laplace estimator in fact introduces a prior uniform probability for each class; i.e., before any instances have reached the leaf, the probability for each class is $1/C$. In Weka, all tree probability estimates are calculated using the Laplace correction.

For evolutionary based rule extraction methods, the chosen fitness function must correspond to the optimization criterion in use. Normally, the fitness function simply maximizes the fidelity to the opaque model, using a 0/1 measure, i.e., the extracted model should make as many identical classifications as possible (on training or validation data) to the opaque model. Sometimes a small length penalty is added to the fitness function to prioritize more compact extracted models. Here, however, we suggest the use of a more informed optimization criterion, based on probability estimates instead of just final predictions. Simply put, the idea is to minimize the difference in probability estimates between the opaque model and G-REX. For this, we will employ a fitness function based on the *Brier score* [6], which measures the quality of probability estimates using the mean squared error.

For two-class problems, let y_i denote the response variable (class) of instance i, where $y_i = 0$ or 1. Denote the probability estimate that instance i belongs to class 1, by p_i. The Brier score (BS) is then defined as

$$BS = \sum_{i=1}^{N}(y_i - p_i)^2 \tag{5}$$

which is exactly the sum of squares of the difference between the true class and the predicted probability over all instances. For the multi-class case, the generalized Brier score can be used. Let p_{ij} denote the probability estimate that instance i belongs to class j and y_{ij} be an indicator variable such that $y_{ij} = 1$ if $y_i = j$ and 0 otherwise. Then, the generalized Brier score is defined as

$$GBS = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{C}(y_{ij} - p_{ij})^2 \tag{6}$$

The generalized Brier score is of course reduced to the Brier score when $C = 2$. A high (generalized) Brier score indicates poor predictive performance. As mentioned above, y_{ij} is normally restricted to the values 0 and 1. If, however, both y_{ij} and p_{ij} instead denote probability estimates, the Brier score can still be used, but now to measure the difference between two probability estimates. With this in mind, we will in this study use the following fitness function[1] when extracting G-REX trees

$$fitness = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{C}(p_G^{c_j}(i) - p_E^{c_j}(i))^2 \tag{7}$$

$p_G^{c_j}(i)$ is the G-REX probability estimate of class c_j for instance i and $p_E^{c_j}(i)$ is the corresponding probability estimate of the opaque ensemble.

[1] A very small length penalty is also included in the actual fitness function in order to encourage smaller models.

Finally, we must define how G-REX probability estimates should be calculated. Since G-REX will evolve trees, it must be noted that it is really the leaf that a specific instance falls into (i.e. all other instances also reaching that leaf) that will determine the probability estimate. Because of this, we will evaluate two slightly different strategies for obtaining probability estimates, one working at the instance level and one at the leaf level. The first strategy (*GR-i*) is the simple method using relative frequencies described above; see equation (3). This strategy, consequently, tries to minimize the difference in probability estimates between the extracted and the opaque model, on an instance-for-instance level. The second strategy (*GR-l*), on the other hand, tries to optimize the G-REX tree by considering all instances that fall in a specific leaf. Simply put, the probability estimate for each instance falling in a certain leaf is calculated as the average probability estimate from the *opaque* model of these instances. More specifically, let $w_{iq} = 1$ if instances i and q fall in the same leaf and 0 otherwise. Then

$$p_k^{c_j}(i) = \frac{\sum_{q=1}^{N} w_{iq} p_E^{c_j}(q)}{\sum_{q=1}^{N} w_{iq}} \quad (8)$$

where $p_E^{c_j}(q)$ is the probability estimate of the opaque, ensemble model for class c_j for instance q.

3.1 Experiments

This study contains two experiments. In the first experiment, G-REX extracted trees from Random forest ensembles consisting of 500 random trees. In the second experiment, ensembles of 15 bagged MLPs were used as opaque models. For these techniques, default settings in Weka were used. In the experimentation, the performance of the two suggested G-REX strategies *GR-i* and *GR-l* was compared to the standard technique C4.5 (implemented as J48 and using default settings in Weka) and standard G-REX (*GR-o*), i.e., using a fitness function based on 0/1 fidelity.

In the experiments, both accuracy and area under the ROC curve (AUC) were used for evaluation. While accuracy is based only on the final classification, AUC measures the ability to rank instances according to how likely they are to belong to a certain class; see e.g. [11]. AUC can be interpreted as the probability of ranking a true positive instance ahead of a false positive; see [3].

For the evaluation, standard 4-fold cross-validation was used. The reported accuracies were therefore averaged over the four folds. Following the standard procedure for AUC evaluation, only one ROC curve based on all test set instances from each fold was produced per data set. In addition, to evaluate comprehensibility, the tree size (measured as total number of nodes) was calculated for J48 and all three versions of G-REX.

Regarding GP settings for G-REX, previous studies have showed that these should ideally be optimized for each data set. In this, study, however, we settled for using almost identical parameter values over all data sets. The only difference was that since multiclass problems were expected to require more complex models, G-REX was allowed to start with larger trees (creation depth 14 instead of 10) on these data sets. Table 1 below shows the GP parameter settings used.

Table 1: GP parameters

Parameter	Value
Crossover rate	0.8
Mutation rate	0.01
Population size	1000
Generations	100
Creation depth	10/14
Creation method	Ramped half-and-half
Selection	Roulette wheel
Elitism	Yes

The 26 data sets used are all well-known and publicly available from the UCI Repository [2]. For both G-REX and the opaque models, missing values for all numerical attributes were handled by replacing the missing value with the mean value of that attribute. For nominal attributes, missing values were replaced with the mode value. For J48, the internal handling of missing values was used, since this is normally more effective than the general approach, described above.

4. RESULTS

Table 2 below shows the accuracy results for Experiment 1. Considering the mean ranks, there is a clear ordering, showing that the two suggested approaches outperformed GR-o, which in turn outperformed J48.

Table 2: Experiment 1 - Accuracy

Data sets	RF	J48	GR-o	GR-i	GR-l
Balance S	.813	.787	.790	.790	**.798**
Bcancer	.678	.699	.727	**.731**	.717
Cmc	.512	.506	**.547**	.532	.534
Colic	.840	**.856**	.845	.840	.848
Credit-A	.864	.855	.841	.855	**.859**
Credit-G	.769	**.730**	.724	.718	.727
Diabetes	.760	.733	.734	**.738**	.736
Ecoli	.816	.786	.807	**.813**	.789
Glass	.785	.668	.659	**.673**	**.673**
Haberman	.647	.739	**.761**	.745	.739
Heart-C	.828	.759	.776	.779	**.782**
Heart-H	.813	.779	**.810**	.789	.793
Heart-S	.829	.770	.759	**.774**	**.774**
Hepatitis	.839	.807	.813	.819	**.826**
Iono	.926	**.900**	.846	.855	.866
Iris	.947	.953	.960	**.967**	**.967**
Labor	.930	.737	.825	**.877**	.842
Lcancer	.531	.406	**.438**	**.438**	**.438**
Liver	.739	.620	.620	.632	**.652**
Lymph	.845	.730	**.824**	.777	.791
Sonar	.846	.755	.736	**.760**	.726
Tae	.629	**.550**	**.550**	.523	.523
Vote	.956	.956	.949	**.963**	**.963**
WBC	.966	.956	**.961**	.957	.956
Wine	.972	.916	.910	**.933**	.916
Zoo	.951	.931	.911	.911	**.941**
Mean	**.809**	**.765**	**.774**	**.776**	**.776**
Mean rank		**3.10**	**2.71**	**2.12**	**2.08**

Looking at individual data sets, the picture is quite clear. J48 obtained the worst accuracy on 14 of 26 data sets, while GR-o had rank 3 or worse on 16 data sets. GR-i, on the other hand, had a rank of 2 or better on 16 data sets, while the corresponding result for GR-l was 19 data sets with a rank of 2 or better. To determine if there are any statisti-

cally significant differences, we use the statistical tests recommended by Demšar [10] for comparing several classifiers over a number of data sets; i.e., a Friedman test [12], followed by a Nemenyi post-hoc test [19]. With four classifiers and 26 data sets, the critical distance (for $\alpha = 0.05$) is 0.92, so based on these tests, GR-i and GR-l both obtained significantly higher accuracies than J48. Furthermore, the two novel approaches also performed much better than GR-o, even if the differences are not statistically significant. Turning to AUC results, Table 3 below shows that the suggested approaches were very successful. As a matter of fact, the mean ranks show that the AUCs obtained by GR-i and GR-l were significantly higher than both J48 and GR-o.

Table 3: Experiment 1 - AUC

Data sets	RF	J48	GR-o	GR-i	GR-l
Balance S	.947	.834	.853	.880	**.890**
Bcancer	.621	.596	.557	**.694**	.612
Cmc	.674	.654	.679	.696	**.702**
Colic	.886	.829	.813	.829	**.840**
Credit-A	.932	.871	.888	**.904**	.903
Credit-G	.799	.667	.627	**.728**	.718
Diabetes	.820	.726	.713	**.779**	.766
Ecoli	.963	.892	.912	**.930**	**.930**
Glass	.929	.775	.814	**.832**	.823
Haberman	.655	.538	.654	.673	**.680**
Heart-C	.894	.763	.803	.795	**.829**
Heart-H	.876	.740	.779	.834	**.844**
Heart-S	.888	.751	.755	.826	**.838**
Hepatitis	.858	.609	.767	.721	**.820**
Iono	.981	**.871**	.822	.831	**.871**
Iris	.995	.968	.968	.972	**.982**
Labor	.974	.720	.819	.859	**.882**
Lcancer	.648	.489	.565	.566	**.649**
Liver	.767	.631	.565	.612	**.661**
Lymph	.925	.792	.831	.836	**.883**
Sonar	.944	.757	.757	.792	**.829**
Tae	.765	.698	**.708**	.707	.688
Vote	.990	**.977**	.921	.951	.967
WBC	.990	.956	.967	.969	**.970**
Wine	.999	.937	.922	.951	**.968**
Zoo	.998	.976	.949	.979	**.996**
Mean	**.874**	**.770**	**.785**	**.813**	**.829**
Mean rank		**3.36**	**3.24**	**2.00**	**1.40**

So, when using the AUC metric for the evaluation, the more informed optimization criterion led to significantly better extracted models. Another interesting observation is that GR-l clearly outperformed GR-i with regard to AUC. Counting the number of wins, GR-l actually managed to win 19 of the 26 data sets.

Table 4 below shows the accuracy results for Experiment 2, i.e., when extracting from bagged ANNs. Although the overall picture is quite similar to Experiment 1, the differences are smaller here. Consequently, the only statistically significant result is that GR-i obtained higher accuracy than J48. The main difference, compared to Experiment 1, is that GR-o here performed quite well, actually winning as many as 11 data sets. Nevertheless it should be noted that on those data sets (possibly with the exception of the Diabetes data set) GR-i and GR-l most often obtained accuracies comparable to GR-o. This is, however, not true the other way around. On several data sets (especially Iono and Hepatitis, but also Haberman, Heart-S and Wine) GR-i and GR-l had considerably higher accuracies than GR-o.

Table 4: Experiment 2 - Accuracy

Data sets	ANN	J48	GR-o	GR-i	GR-l
Balance S	.928	.787	**.803**	.795	**.803**
Bcancer	.699	.699	.741	**.752**	.738
Cmc	.541	.506	**.547**	.519	.546
Colic	.839	**.856**	.848	.845	.821
Credit-A	.861	.855	**.857**	.852	.846
Credit-G	.754	**.730**	.711	.720	.721
Diabetes	.771	.733	**.758**	.740	.737
Ecoli	.860	.786	.792	**.795**	.792
Glass	.696	.668	.654	**.673**	.636
Haberman	.719	.739	.748	.761	**.768**
Heart-C	.825	.759	.795	**.809**	**.809**
Heart-H	.816	.779	**.799**	.796	.793
Heart-S	.807	.770	.756	**.793**	.770
Hepatitis	.865	.807	.781	.807	**.832**
Iono	.906	**.900**	.821	.883	.869
Iris	.953	.953	.953	**.967**	.960
Labor	.930	.737	**.842**	**.842**	.825
Lcancer	.469	.406	.438	.438	**.469**
Liver	.701	.620	**.626**	**.626**	**.626**
Lymph	.851	.730	**.764**	**.764**	.757
Sonar	.851	.755	.745	**.760**	.750
Tae	.589	.550	**.556**	.497	.523
Vote	.949	.956	**.972**	.959	.963
WBC	.964	.956	.954	.960	**.963**
Wine	.972	.916	.921	.933	**.938**
Zoo	.951	.931	**.960**	.941	.951
Mean	**.810**	**.765**	**.775**	**.778**	**.777**
Mean rank		**3.19**	**2.38**	**2.10**	**2.33**

As seen in Table 5 below, the AUC results for Experiment 2 are very similar to the results in Experiment 1.

Table 5: Experiment 2 - AUC

Data sets	ANN	J48	GR-o	GR-i	GR-l
Balance S	.988	.834	.867	**.871**	.853
Bcancer	.690	.596	.628	.647	**.680**
Cmc	.715	.654	.666	.694	**.699**
Colic	.868	.829	.818	**.850**	.849
Credit-A	.916	.871	.885	**.909**	.897
Credit-G	.783	.667	.628	.717	**.722**
Diabetes	.826	.726	.759	.773	**.779**
Ecoli	.964	.892	.876	**.942**	**.942**
Glass	.882	.775	.797	.841	**.891**
Haberman	.722	.538	.621	**.655**	.641
Heart-C	.896	.763	.819	.833	**.846**
Heart-H	.894	.740	.756	**.832**	.829
Heart-S	.882	.751	.746	.795	**.818**
Hepatitis	.858	.609	.612	.707	**.742**
Iono	.950	**.871**	.784	.842	.824
Iris	.996	.968	.959	.970	**.994**
Labor	.961	.720	.832	**.867**	.862
Lcancer	.619	.489	.565	.592	**.748**
Liver	.740	.631	.624	**.658**	.642
Lymph	.927	.792	.800	.805	**.835**
Sonar	.920	.757	.742	.786	**.787**
Tae	.722	**.698**	.689	.645	.682
Vote	.989	**.977**	.967	.945	.950
WBC	.989	.956	.953	.968	**.975**
Wine	.999	.937	.945	.946	**.970**
Zoo	.998	.976	.976	.968	**.983**
Mean	**.873**	**.770**	**.781**	**.810**	**.825**
Mean rank		**3.26**	**3.26**	**1.98**	**1.50**

Again, both GR-i and GR-l obtained significantly higher AUCs than standard G-REX and J48. In addition, GR-l is again clearly better than GR-i, winning 17 data sets and

tying one, in a direct pairwise comparison. It is also interesting to note that the difference in AUC between standard G-REX and J48 is again fairly small. In this experiment, the mean ranks are even identical. In Table 6 below, the average sizes, measured as total number of nodes in the tree, are given for the three rule extraction algorithms and J48. For simplicity, J48 multi-splits were counted as only one node.

Table 6: Model sizes

Data sets	J48	GR-o		GR-i		GR-l	
		RF	NN	RF	NN	RF	NN
Balance S	76	51	51	45	54	46	53
Bcancer	16	36	29	21	28	32	30
Cmc	245	29	33	17	16	22	23
Colic	19	25	19	18	15	15	18
Credit-A	24	15	15	11	13	11	11
Credit-G	98	21	22	11	16	13	12
Diabetes	38	17	20	16	11	21	19
Ecoli	30	33	28	24	18	14	20
Glass	42	31	44	19	18	27	19
Haberman	2	23	20	21	11	17	16
Heart-C	35	42	30	23	24	32	25
Heart-H	14	20	23	11	14	16	13
Heart-S	34	32	27	32	34	31	33
Hepatitis	12	22	23	23	22	25	19
Iono	22	21	20	20	20	29	21
Iris	8	11	13	13	9	13	13
Labor	7	14	12	11	15	11	14
Liver	33	24	25	17	14	18	16
LCancer	14	23	32	26	26	33	32
Lymph	25	27	38	24	29	24	29
Sonar	23	31	31	28	34	32	34
Tae	40	35	35	27	11	45	19
Vote	9	15	11	14	14	15	17
WBC	20	13	15	16	15	15	14
Wine	10	22	24	18	26	27	21
Zoo	14	23	23	16	21	18	19
Mean	**35**	**25**	**25**	**20**	**20**	**23**	**21**

Although the main result is that all techniques normally produce quite comprehensible models, the use of J48 does, however, occasionally result in extremely bushy trees that can not be deemed comprehensible. In the experimentation, all J48 trees on the CMC and Credit-G data sets, and most on the Balance scale data set must be considered incomprehensible. All models extracted by the three G-REX versions were, on the other hand, comprehensible, even if some trees from the Balance scale data set probably should be regarded as borderline cases. Somewhat surprisingly, trees evolved using the more complex fitness functions were normally smaller than standard G-REX trees. Furthermore, the kind of opaque model used for the rule extraction seems to have very little influence on model sizes. Although the differences in both AUC and size between GR-i and GR-l are fairly small, it is no surprise that the strategy producing slightly larger (more detailed) models, also obtains the best AUC. Still, the most important result is that the use of the more informed fidelity criterion, as fitness function, did not result in more complex trees, compared to standard G-REX. All in all, a huge majority of all extracted trees are very compact, often around 10 nodes. Trees of this size are of course easy to inspect and understand, see the sample tree of size 13 in Figure 3 below. It must be noted, of course, that the probability estimates are only presented in the final tree, they are not actually part of the evolving trees.

```
if plasma > 142.01
|T: [23.4% 76.6%]
|F: if pedigree < 1.26
|   |T: if bmi < 26.97
|   |   |T: [93.3% 6.7%]
|   |   |F: if age > 25.90
|   |   |   |T: if pedigree > 0.74
|   |   |   |   |T: [41.9% 58.1%]
|   |   |   |   |F: [69.1% 30.9%]
|   |   |   |F: [89.3% 10.7%]
|   |F: if age > 29.0
|   |   |T: [28.0% 72.0%]
|   |   |F: [77.9% 22.1%]
```

Figure 3: Sample evolved tree. Diabetes data set

Summarizing the experimental results, the two algorithms using the more informed fidelity criterion obtained significantly higher AUCs than standard G-REX and J48 in both experiments. The fact that the differences in AUC between the suggested algorithms and standard G-REX were statistically significant is obviously quite reassuring, showing that the use of the more informed fidelity criterion leads to extracted models with a better ranking ability. In addition, the AUC results show that the strategy operating on the leaf level (GR-l) clearly outperformed the strategy operating on individual instances (GR-i).

Regarding accuracy, the picture is slightly different, even if the two suggested algorithms also obtained the best accuracies overall. It should, however, be noted that the gain in accuracy, when compared to standard G-REX, was not significant for any of the four setups. Still, comparing GR-i and GR-l to J48, the extracted trees were significantly more accurate in three of the four setups evaluated.

If mean ranks are used to compare model sizes, the results show very small differences. On most data sets, all techniques produce clearly comprehensible models of similar size. The only exception is when J48, on a couple of data sets, returns huge trees, which obviously are incomprehensible.

5. CONCLUSIONS

We have in this paper introduced and evaluated a novel algorithm for rule extraction from opaque models. The suggested algorithm is, to the best of our knowledge, the first rule extraction algorithm maximizing fidelity at the probability estimation level, instead of just final classifications. More specifically, the algorithm minimizes the generalized Brier score, comparing the probability estimates from the opaque and the extracted model. Two different strategies for calculating the probability estimates for the extracted model, one working at the instance level and one at the leaf level, were evaluated. Since the extraction strategy is based on genetic programming, the optimization used a population of initially randomized candidate trees, each representing an extracted model. Using the generalized Brier score as fitness function, the evolution searched for trees having probability estimates as similar as possible to the original, opaque, model.

In the experimentation, both strategies (i.e. ways of calculating the probability estimates) were evaluated when the novel algorithm was tested on 26 UCI data sets. The al-

gorithm was also applied to two kinds of ensemble models, Random forests and bagged ANNs, leading to four different setups in all. For comparison, we used the well-known tree inducer J48, but also standard G-REX, i.e., the exact same rule extraction algorithm, but with a fitness function maximizing 0/1 fidelity. From the results, it is obvious that the more informed fidelity criterion led to more accurate extracted models. As a matter of fact, all four setups produced models with significantly higher AUC than both J48 and standard G-REX. When extracting models from Random forests, both strategies had significantly higher accuracy than J48, and were considerably more accurate than standard G-REX. When extracting from bagged ANNs, both strategies clearly outperformed J48, but only the strategy working on the instance level achieved a statistically significant difference.

Looking at the comprehensibility criterion, almost all trees extracted in the experimentation were clearly comprehensible. Most importantly, the increase in predictive performance, caused by the use of the more informed fidelity criterion, did not come at the expense of reduced comprehensibility.

6. REFERENCES

[1] R. Andrews, J. Diederich, and A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowl.-Based Syst.*, 8(6):373–389, 1995.

[2] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.

[3] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[5] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, January 1984.

[6] G. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

[7] M. Craven and J. Shavlik. Rule extraction: Where do we go from here? University of Wisconsin Machine Learning Research Group working Paper 99-1.

[8] M. W. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems*, pages 24–30. MIT Press, 1996.

[9] M. W. Craven and J. W. Shavlik. Using neural networks for data mining. *Future Gener. Comput. Syst.*, 13(2-3):211–229, 1997.

[10] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[11] T. Fawcett. Using rule sets to maximize roc performance. In *Proceedings of the 2001 IEEE International Conference on Data Mining, ICDM'01*, pages 131–138, Washington, DC, USA, 2001. IEEE Computer Society.

[12] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of American Statistical Association*, 32:675–701, 1937.

[13] L. Fu. Rule learning by searching on adapted nets. In *AAAI*, pages 590–595, 1991.

[14] J. Huysmans, B. Baesens, and J. Vanthienen. Using rule extraction to improve the comprehensibility of predictive models. FETEW Research Report KBI 0612, K. U. Leuven, 2006.

[15] U. Johansson. *Obtaining Accurate and Comprehensible Data Mining Models: An Evolutionary Approach. PhD-thesis*. Institute of Technology, Linköping University, 2007.

[16] U. Johansson, R. König, and L. Niklasson. Rule extraction from trained neural networks using genetic programming. In *13th International Conference on Artificial Neural Networks, supplementary proceedings*, pages 13–16, 2003.

[17] U. Johansson, R. König, and L. Niklasson. Inconsistency - friend or foe. In *IJCNN*, pages 1383–1388, 2007.

[18] U. Markowska-Kaczmar and M. Chumieja. Discovering the mysteries of neural networks. *Int. J. Hybrid Intell. Syst.*, 1(3,4):153–163, 2004.

[19] P. B. Nemenyi. *Distribution-free multiple comparisons. PhD-thesis*. Princeton University, 1963.

[20] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[21] H. L. Rudy, H. Lu, R. Setiono, and H. Liu. Neurorule: A connectionist approach to data mining. pages 478–489, 1995.

[22] S. Thrun, G. Tesauro, D. Touretzky, and T. Leen. Extracting rules from artificial neural networks with distributed representations. In *Advances in Neural Information Processing Systems 7*, pages 505–512. MIT Press, 1995.

[23] G. G. Towell and J. W. Shavlik. The extraction of refined rules from knowledge-based neural networks. In *Machine Learning*, pages 71–101, 1993.

[24] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, June 2005.