Using Genetic Programming to Increase Rule Quality

Rikard König[1], Ulf Johansson[1], Lars Niklasson[2]

[1]School of Business and Informatics, University of Borås, Sweden
[2]School of Humanities and Informatics, University of Skövde, Sweden
rikard.konig@hb.se, ulf.johansson@hb.se, lars.niklasson@his.se

Abstract

Rule extraction is a technique aimed at transforming highly accurate opaque models like neural networks into comprehensible models without losing accuracy. G-REX is a rule extraction technique based on Genetic Programming that previously has performed well in several studies. This study has two objectives, to evaluate two new fitness functions for G-REX and to show how G-REX can be used as a rule inducer.

The fitness functions are designed to optimize two alternative quality measures, area under ROC curves and a new comprehensibility measure called brevity. Rules with good brevity classifies typical instances with few and simple tests and use complex conditions only for atypical examples. Experiments using thirteen publicly available data sets show that the two novel fitness functions succeeded in increasing brevity and area under the ROC curve without sacrificing accuracy. When compared to a standard decision tree algorithm, G-REX achieved slightly higher accuracy, but also added additional quality to the rules by increasing their AUC or brevity significantly.

Introduction

Most high-accuracy techniques for predictive classification produce opaque models like neural networks, ensembles or support vector machines. Opaque predictive models make it impossible for decision-makers to follow and understand the logic behind a prediction, which, in some domains, must be deemed unacceptable. In domains where models need to be interpretable (or even comprehensible) accuracy is often sacrificed for comprehensibility by using simpler but transparent models; most typically decision trees. This tradeoff between predictive performance and interpretability is normally termed *the accuracy vs. comprehensibility tradeoff*. With this tradeoff in mind, several researchers have suggested rule extraction algorithms where the opaque models are transformed into comprehensible models, keeping an acceptable accuracy.

Most significant are the many rule extraction algorithms suggested for extracting rules from trained neural networks; see e.g. RX (Hongjun, Setiono and Huan 1995) and TREPAN (Craven and Shavlik 1996). Several authors have discussed key demands on a reliable rule extraction method; see e.g. (Andrews, Diederich and Tickle 1995) and (Craven and Shavlik 1999). The most common criteria are: accuracy (the ability of extracted representations to make accurate predictions on previously unseen data), comprehensibility (the extent to which extracted representations are humanly comprehensible) and fidelity (the extent to which extracted representations accurately model the opaque model from which they were extracted).

The accuracy of an extracted rule set is evaluated similar to all predictive models; i.e. typically measuring error rates using either a hold-out set or some cross-validation scheme. Comprehensibility, on the other hand is harder to assess. In most studies, comprehensibility is measured as the size of the extracted representation. This is an obvious simplification, chosen with the motivation that it makes comparisons between techniques fairly straightforward. In addition, size could be calculated in a number of ways. Some typical choices (for tree structures) include number of questions, total number of tests, number of nodes and number of interior nodes. Furthermore, comprehensibility is in practice, clearly not linear, making all measures based on size slightly dubious. All in all, though, the choice to evaluate comprehensibility using the size of the model must be regarded as the most accepted.

In this study we use a rule extraction algorithm called G-REX (Genetic Rule EXtraction). G-REX was initially suggested in (Johansson, Konig and Niklasson 2003) but has after that been extended and thoroughly evaluated, for a summary see (Johansson, 2007). G-REX uses an extraction strategy based on Genetic Programming (GP) making it straightforward to use a variety of representation languages, just by varying the function and terminal sets. G-REX also explicitly targets the accuracy vs. comprehensibility tradeoff by using a fitness function where accuracy is balanced against the size of the extracted representation.

In this study we use G-REX as a rule inducer instead of as a rule extraction algorithm. The only difference to the normal use of G-REX is that the target of the GP is the actual target value instead of predictions of an opaque model. There are two reasons for this approach, first the aim of this study is to evaluate two new fitness functions and for this an opaque model is an extra step that will only add unnecessary complexity to the study. The second, more important reason, is that this approach also allows us to demonstrate the advantages of a rule inducer based on GP. Most rule inducers producing comprehensible models only optimize accuracy. This is a fundamental design choice that is built into the algorithms, making it hard or impossible to change the optimization criteria. GP on the other hand is very flexible and can optimize arbitrary functions. G-REX is an excellent GP technique to use for this evaluation as it allows fitness function to be added and combined as buildings block for more complex optimization criterions. Another strength of G-REX is that it can use arbitrary representation languages facilitating a tailored solution consisting of an optimal representation language and an optimal fitness function for each problem.

Method

The purpose of this study is to evaluate the effect of different fitness functions when using GP for rule induction. Although accuracy is the standard criterion when evaluating predictive models, there are several other criteria that could be used. Here, area under the ROC curve (AUC), and a special fitness function called brevity is evaluated. Brevity is aimed at producing compact rule sets, thereby increasing the comprehensibility. During experimentation, comparisons are made against both the standard G-REX fitness function, primarily optimizing accuracy and against MatLabs decision tree algorithm Treefit.

G-REX fitness functions

One of G-REX main strengths is that an arbitrary fitness function can be used to optimize one or more criteria. In previous G-REX studies the fitness function here called ACCFitness has been used. In this study two new fitness functions AUCFitness and BREFitness are introduced. All three fitness functions are described in detail below.

ACCFitness

In previous studies the fitness function ACCFitness was used to optimize accuracy while keeping evolved G-REX rules fairly compact. The (training or validation) accuracy is simply calculated by dividing the number of correctly classified instances with the total number of instances. To encourage smaller rule sets, a punishment is given in relation to the length of the rule. Rule length is calculated as the number of elements (functions and terminals) in the rule.

$$ACCFitness = 100 * \frac{\# correct_i}{\# instances} - length_i * p \qquad (1)$$

A parameter p is used to balance the size of the length punishment against accuracy. It should be noted that this use of accuracy in the fitness function ensures that the length punishment will affect the evolved rules in the same way regardless of the number of instances in the data set.

AUCFitness

Provost, Fawcett and Kohavi (1998) argues that using accuracy for evaluating classifiers has two serious flaws; i.e. it is presumed that the true class distribution is known and that the misclassification cost is equal for all classes. This is regarded as a problem as these assumptions rarely are true for real world problems. Instead the Receiver Operating Characteristic (ROC) curve is suggested as a more appropriate metric for comparing classification performance. Specifically, ROC curves are insensitive to changes in class distributions. ROC graphs are commonly used in medical decision making, and have in recent years been used increasingly in machine learning and data mining research; see e.g.(Fawcett 2006). ROC curves measure the relationship between hit rate and false alarm rate and are based on estimations of the probability that an instance belongs to a certain class. For a detailed description of how ROC curves are constructed see (Fawcett 2006). To compare classifiers using ROC curves, the curve has to be converted into a single scalar value, normally the area under the ROC curve (AUC). Since ROC curves are projected onto the unit square, AUC will always be bounded by 0 and 1. An AUC of 1 means that the predictive model is perfect, while random guessing corresponds to an AUC of 0.5.

Since G-REX uses GP to evolve rules, it does not, just like standard decision trees, provide explicit probability distributions for the classifications. The number of correct and incorrect training classifications are, however, recorded for each leaf in the tree, making it possible to produce probability distributions if needed.

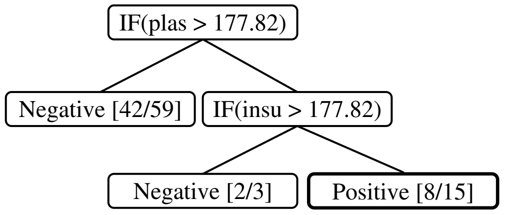

Figure 1 –Correct classifications/instance reaching node

The relative frequencies are normally not used directly for probability estimations since they do not consider the number of training instances supporting a classification. The Laplace estimate is a common technique to produce probability distributions based on the support (Margineantu and Dietterich 2002). Equation 2 below shows how probability estimates p are calculated when using Laplace. N is the total number of instances, C is the number of classes and k is the number of training instances supporting the predicted class A.

$$p_{classA} = \frac{k+1}{N+C} \tag{2}$$

In the example above, the probability estimate for the lower right node would be calculated as *8/15* without Laplace and *9/17 ((8+1)/(15+2))* using Laplace.

When employing *AUCFitness*, G-REX uses Laplace estimates for ranking the instances during the ROC analysis. Overall, *AUCFitness* balances the AUC and the length of the rule in the same way that *ACCFitness* balances accuracy and length.

$$AUCFitness = 100 * AUC_i - length_i * p \tag{3}$$

BREFitness

The fitness function *BREFitness* is aimed at optimizing rule brevity. Instead of just using the length of the rules as a base for punishment *BREFitness* evaluates how the rule is actually used when classifying instances. The idea is that a tree must not only be relatively small but also should classify as many instances as possible using only a few tests i.e. the rule brevity should be high. The overall principle is that a typical instances should be classified using a few simple tests, while more complex rules should be used for more atypical patterns only. Although brevity could be measured using different metrics, we suggest that brevity should be calculated as the *average number of tests needed to be checked in order to classify all instances in the data set*, i.e. the brevity for a rule set *r* when classifying the instances *k=1..n* is:

$$BRE_r = \sum_{k=1}^{k=n} \frac{\# conditions_k}{\# instances} \tag{4}$$

If the numbers in the leaves in Figure 1 show test set instances, the corresponding brevity is calculated as: *(59*1+3*2+15*2)/(59+3+15)=1.1234*. Note that using this definition, brevity does not consider accuracy at all. A rule with the best (lowest) possible brevity (1.0) could classify all instances incorrectly. Brevity should therefore only be used to evaluate rules with comparable accuracy. Because of this, *BREFitness* functions uses accuracy as a reward and brevity as a punishment (a lower *BRE* means that the rule has better brevity). The size of the punishment is adjusted, again using a parameter *p* .

$$BREFitness = 100 * \frac{\# correct_i}{\# instances} - BRE_i * p \tag{5}$$

Treefit

The *Treefit* algorithm in MatLab is a decision tree algorithm for regression and classification which is based on CART(Breiman et al. 1983). *Treefit* produces a binary tree with splits based on the input variables. For continuous variables the relational operators < and > are used. Categorical variables are split using the = operator (*variable = category list*) which is true if the current variable is one of the listed categories. Gini diversity index (GDI) is used as splitting criterion and each node is set to contain at least five instances. GDI measures the class impurity in the node and is calculated by equation 6. Given a node t with estimated class probabilities $p^2(j|t)$, j=1,…,J (for J classes) the GDI is given by:

$$GDI = 1 - \sum_j p^2\left(j \mid t\right) \tag{6}$$

When pruned, the tree is first evaluated on the training data using 10-fold cross validation and then pruned to the minimal subtree that is within one standard error of the minimal cross validation error.

Experiments

The following section describes the details of the experiments.

Data sets

This study uses 13 data sets publicly available from the UCI – repository (Blake and Merz, 1998) All data sets are binary problems; i.e. two-class problems. Table 1 below presents the characteristics of each data set. *Ins.* is the number of instances; *Con.* is the number of continuous inputs and *Cat.* is the number of categorical input variables.

Data set	Ins.	Con.	Cat.
Breast-w	699	9	0
Colic	368	7	15
Credit-a	690	6	9
Credit-g	1000	7	13
Diabets	768	8	0
Heart-cleveland	303	6	8
Heart-statlog	270	13	0
Hepatitis	155	6	13
Ionosphere	351	24	0
Labor	57	8	8
Liver-disorders	345	6	0
Sonar	208	60	0
Vote	435	0	16

Table 1 - Characteristics of data sets

Preprocessing

Missing values are, for continuous input variables, replaced with the mean value of all none missing values for the variable. Categorical missing values are, similarly, replaced with the type value of the variable. When missing values have been handled, each data set is divided into ten folds which are stratified to ensure that each fold will have a representative class distribution. For evaluation, standard 10-fold cross validation is used in the experiments.

G-REX

In this study G-REX executes in a batch of ten runs for each fold. The rule with the highest fitness is selected as the winner and is then evaluated on the test set. All fitness functions use *p = 0.05* , which should produce short and comprehensible rules. G-REX used the GP settings presented in Table 2.

Number of generations	100
Population size	1000
Crossover probabillity	0.8
Mutation probabillity	0.001
Creation type	Ramped half and half
Maximum creation depth	6
Length punishment (*p*)	0.05

Table 2 - G-REX settings used in all experiments

G-REX can use arbitrary representations languages but in this study the simple *if-else* representation languages presented using Backus-Naur form in Figure 2 was deemed to be sufficient.

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

DTree     :-  (if RExp Dtree Dtree) | Class
RExp      :-  (ConInp ROp ConConst) |
              (CatInp = CatConst)
ROp       :-  < | >
CatInp    :-  Categorical input variable
ConInp    :-  Continuous input variable
Class     :-  Class label
CatConst  :-  Categorical attribute value
ConConst  :-  □
```

Figure 2 - Used representation language

The tree displayed in Figure 1 is an example of this representation; if an *if-statment* is true, the left sub- tree is evaluated, if not the right. An instance is classified as the value of the leaf it reaches, following the *if-statements.*

Evaluation

For actual evaluation, all G-REX fitness function and *Treefit* are evaluated using accuracy, AUC and brevity. These evaluations are done regardless of optimization criterion to investigate whether *AUCFitness* or *BREFitness* perform better than the standard accuracy fitness function. The results are averaged over all folds and reported individually for each dataset. Overall performance of the different techniques are compared using Wilcoxon signed-ranks tests (Wilcoxon, 1945). The signed ranks test is a (two-sided) test of the hypothesis that the difference between the matched samples in the vectors X and Y comes from a distribution with median zero. The tests are performed at the 0.05 significance level, so a p-values larger than 0.05 indicates that the null hypothesis (median is zero) cannot be rejected at the 5% level; a p-value smaller than 0.05 indicates that the results are significantly different.

Results

In the following section the results of the experiments are presented. Each table shows a different metric for the same rule set; i.e. accuracy, AUC, and brevity are calculated for one rule set per technique and dataset, but the results are presented in three different tables. All results are averaged values for ten stratified folds. The result for the best technique on each data set is presented in bold letters. To allow comparisons of the techniques the average result for all dataset are presented on the last row of each table. Wilcoxon tests are used to judge if the differences between the techniques are significant. *p*-values that are significant are presented in bold letters.

Table 3 below shows the achieved accuracy for the different techniques on each dataset.

Data set	**Treefit**	**ACCFit.**	**BREFit.**	**AUCFit.**
Breast-w	0.943	**0.963**	0.943	0.941
Colic	**0.853**	0.824	0.788	0.848
Credit-a	**0.855**	0.846	**0.855**	0.852
Credit-g	**0.740**	0.715	0.707	0.707
Diabets	**0.756**	0.750	0.737	0.737
Heart-cleveland	0.758	**0.788**	0.725	0.749
Heart-statlog	0.774	**0.804**	**0.804**	0.726
Hepatitis	0.769	**0.819**	0.814	0.787
Ionosphere	0.897	**0.932**	0.880	0.877
Labor	0.747	0.783	0.803	**0.817**
Liver-disorders	0.661	0.655	**0.678**	0.626
Sonar	**0.725**	0.711	0.721	0.687
Vote	0.954	0.949	**0.956**	**0.956**
Average	0.802	**0.811**	0.801	0.793

Table 3 - Average accuracy over 10-folds

All techniques produce comparable results. *ACCFitness* achieves slightly higher accuracy than the other techniques, but as seen in Table 4, none of the techniques are significantly better or worse than another.

	ACCFit.	**BREFit.**	**AUCFit.**
Treefit	0.2986	0.7832	0.0835
ACCFit.		0.3911	0.0803
BREFit.			0.3613

Table 4 - p-values from Wilcoxon tests for accuracy

Table 5 shows the AUC values for the same rule sets that were used to calculate the accuracy in Table 3. *AUCFitness* clearly outperforms the other techniques using this metric, achieving the highest AUC on 9 of 13 data sets. The second best technique is *ACCFitness* followed by *Treefit* and *BREFitness*.

Data set	**Treefit**	**ACCFit.**	**BREFit.**	**AUCFit.**
Breast-w	0.933	0.962	0.947	**0.972**
Colic	0.853	0.819	0.777	**0.883**
Credit-a	0.862	0.891	0.862	**0.922**
Credit-g	**0.719**	0.669	0.534	0.705
Diabets	0.720	0.724	0.681	**0.798**
Heart-cleveland	0.780	0.837	0.724	**0.838**
Heart-statlog	0.771	**0.844**	0.762	0.834
Hepatitis	0.496	0.755	0.567	**0.781**
Ionosphere	0.902	**0.920**	0.876	0.918
Labor	0.740	0.806	0.781	**0.833**
Liver-disorders	0.642	**0.653**	0.652	0.640
Sonar	0.725	0.745	0.737	**0.791**
Vote	0.950	0.968	0.959	**0.977**
Average	0.776	0.815	0.758	**0.838**

Table 5 - Average AUC over 10-folds

If the performance is compared using a Wilcoxon test, it is clear that *AUCFitness* is significantly better than all other techniques. This is no surprise since *AUCFitness* is the only techniques explicitly optimizing AUC. However, it is still a strong result for *AUCFitness* since the same rule sets also obtained accuracies comparable to the other fitness functions.

	ACCFit.	**BREFit.**	**AUCFit.**
Treefit	**0.0457**	0.583	**0.0012**
ACCFit.		**2.44E-04**	**0.0178**
BREFit.			**4.88E-04**

Table 6 - p-values from Wilcoxon test for AUC

When evaluating the rule sets against brevity, the results for *Treefit* is not reported as the representation languages is slightly different.

Data set	**ACCFit.**	**BREFit.**	**AUCFit.**
Breast-w	3.04	**1.54**	2.83
Colic	7.51	**3.09**	9.61
Credit-a	6.56	**3.00**	7.03
Credit-g	6.60	**1.06**	12.55
Diabets	2.04	**1.11**	3.95
Heart-cleveland	7.65	**1.18**	10.73
Heart-statlog	2.91	**1.89**	3.69
Hepatitis	5.44	**1.14**	9.77
Ionosphere	4.37	**1.77**	4.82
Labor	7.67	**2.23**	6.54
Liver-disorders	2.48	**1.58**	4.00
Sonar	3.06	**1.62**	3.84
Vote	5.87	**3.00**	9.77
Average	5.02	**1.86**	6.86

Table 7 - Average brevity over 10-folds

In Table 7 it is clear that *BREFitness* produces rules with much lower brevity. Again, it is worth noting that the same rule sets also performed rather well when evaluated using accuracy. It is natural that *BREFitness* does not perform as well when evaluated against AUC since it produces very short rule sets leading to quite rough probability estimations.

A Wilcoxon test also shows that *BREfitness* produces significantly smaller rule sets than the other techniques. Another interesting result is that *AUCFitness* is significantly worse than all other techniques when compared on brevity. *AUCFitness* shortcomings is probably explained using the same argument as above; i.e. to achieve a high AUC it is necessary to have good probability estimations, which can only be produced by more complex rule sets.

	ACCFit.	**AUCFit.**
BREFit.	**0.000244**	**0.000244**
ACCFit		**0.0225**

Table 8 - p-values from Wilcoxon tests for brevity

Conclusions

This study had two objectives, to evaluate two new fitness functions for the G-REX rule extraction algorithm and to show the advantages of using GP as a rule inducer.

Both fitness functions perform well as they retain the same accuracy as the original fitness function while optimizing yet another other criteria. *AUCFitness* produces rules that have high accuracy and significantly higher AUC than *Treefit* and the other evaluated fitness functions. *BREFitness* uses the proposed comprehensibility measure *brevity* to create rules which classifies the typical instances with simple conditions and only uses complex condition for atypical instances. The fitness function outperforms the *AUCFitness* and *ACCFitness* regarding brevity without losing significantly in accuracy.

The experiments show that when G-REX is used as a rule inducer it is clearly a better alternative than *Treefit* and similar decision trees algorithms. G-REX achieves the same or slightly higher accuracy than *Treefit*, but adds additional quality to the same rules by increasing their AUC or brevity. A decision maker could certainly benefit from this extra rule quality, especially by directing G-REX to produce rules optimized on the quality measure best suited for the problem at hand. Another strength of a GP based rule inducer is that the representation language can be tailored for the problem and thereby possibly increase comprehensibility, accuracy or both.

Overall the experiments show that G-REX is well suited for classification tasks and should certainly be considered when a high quality transparent model is needed.

Acknowledgement

This work was supported by the Information Fusion Research Program (University of Skövde, Sweden) in partnership with the Swedish Knowledge Foundation under grant 2003/0104 (URL: http://www.infofusion.se).

References

Andrews, R., Diederich, J. and Tickle, A. B. 1995. "Survey and critique of techniques for extracting rules from trained artificial neural networks." *Knowledge-Based Systems* 8(6):373-389.

Blake, C. L. and Merz, C. J. 1998. "UCI Repository of machine learning databases." University of California, Department of Information and Computer Science.

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone., C. J. 1983. *Classification and Regression Trees*: Wadsworth.

Craven, M. and Shavlik, J. 1999. "Rule Extraction: Where Do We Go from Here?",Technical Report, University of Wisonsin, Machine Learning Research Group.

Craven, M. W. and Shavlik, J. W. 1996. "Extracting Tree-Structured Representations of Trained Networks." *Advances in Neural Information Processing Systems* 8:24-30.

Fawcett, T. 2006. "An introduction to ROC analysis." *Pattern Recognition Letters* 27(8):861-874.

Hongjun, L., Setiono, R. and Huan, L. 1995. "NeuroRule: a connectionist approach to data mining." *In proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland: Morgan Kaufmann Publishers

Johansson, U. 2007. Obtaining accurate and comprehensible data mining models : an evolutionary approach., Department of Computer and Information Science, Linköpings universitet.

Johansson, U., Konig, R. and Niklasson, L. 2003. "Rule Extraction from Trained Neural Networks using Genetic Programming." *In Joint 13th International Conference on Artificial Neural Networks and 10th International Confernce on Neural Information Processing*, ICANN/ICONIP 2003. Istanbul, Turkey.

Margineantu, D. and Dietterich, T. 2002. "Improved Class Probability Estimates from Decision Tree Models." Nonlinear Estimation and Classification; *Lecture Notes in Statistics* 171:169-184.

Provost, F., Fawcett, T. and Kohavi, R. 1998. "The case against accuracy estimation for comparing induction algorithms.", *Machine Learning. Proceedings of the Fifteenth International Conference (ICML'98) Madison*, WI, USA: Morgan Kaufmann Publishers.

Wilcoxon, F. 1945. "Individual comparisons by ranking models." *Biometrics(1)*:80-83.