G-REX: A Versatile Framework for Evolutionary Data Mining

Rikard Konig[1], Ulf Johansson[1], Lars Niklasson[2]
[1]University of Borås, [2]University of Skövde, Sweden
rikard.konig@hb.se, ulf.johansson@hb.se, lars.niklasson@his.se

Abstract

This paper presents G-REX, a versatile data mining framework based on Genetic Programming. What differs G-REX from other GP frameworks is that it doesn't strive to be a general purpose framework. This allows G-REX to include more functionality specific to data mining like preprocessing, evaluation- and optimization methods, but also a multitude of predefined classification and regression models. Examples of predefined models are decision trees, decision lists, k-NN with attribute weights, hybrid kNN-rules, fuzzy-rules and several different regression models. The main strength is, however, the flexibility, making it easy to modify, extend and combine all of the predefined functionality. G-REX is, in addition, available in a special Weka package adding useful evolutionary functionality to the standard data mining tool Weka.

1. Introduction

Genetic Programming (GP) is a powerful evolutionary computational technique that can be applied to a wide variety of problems. Most GP frameworks are, however, constructed to be as general as possible, and therefore lack most functionality specific to data mining problems; e.g. preprocessing, evaluation techniques and optimization of a set of basic predefined classification and regression models. Two of the more common general purpose GP frameworks, which both lack this basic data mining related functionality, are DGPF [1] and Open BEAGLE [2]. These frameworks could of course be modified and extended with necessary functionality, but due to their aim of being general this would be both undesirable and require a substantial amount of work. We instead, in this paper, present a GP framework designed and built with data mining in mind. Our framework is called G-REX (Genetic-Rule EXtraction) since it was originally used for rule extraction from opaque models like ensembles or neural networks [3] G-REX has, after the introduction, been thoroughly extended and evaluated in several papers presented at both evolutionary and data mining conferences; for a summary see [4]. Lately, G-REX has been substantially modified, with the aim of becoming a general data mining framework, based on GP. In addition to basic data mining functionality, G-REX also offers optimization of several different classification- and regression models, and can easily be extended with new models. G-REX also facilitates the use of different optimization criteria and several performance metrics. All functionality, except the model representation, is readily available from an intuitive graphical user interface or from the command line.

2. Basic technology

G-REX is written in Java and is tested on Windows (XP and Vista) and Linux (Ubuntu). The application is heavily based on design patterns, making the code easy to extend and maintain. Plug-in features allow new nodes and fitness functions to be used without any need of recompilation. New nodes are created by extending a *node* base class and implementing a single method. The definition of a certain model (nodes and syntax) is defined in an external text file using G-REX model definition language, a meta-language similar to standard BNF notation.

A special *experimenter* module allows the planning of larger studies involving several experiments, all with potentially individual settings. Each experiment is added to a list and is then automatically distributed to available processors.

3. The G-REX Application

The following section will describe the main features of G-REX. Staring with available preprocessing functionality followed by evaluation- and optimization methods. Finally the main classification and regression models implemented in G-REX will be presented.

978-0-7695-3503-6/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDMW.2008.117

3.1. Preprocessing

ARFF-files (defined in the Weka-project [5]) are used as the file format for data files. This enables an external way of defining the type of each attribute in the data sets, as well as making comparison and collaboration with Weka easy. At the Weka website, many publicly available machine learning datasets are also available in ARFF format. A special version of Weka, where we have included most G-REX features is also available from the authors homepage[1].

G-REX can handle data in both numerical and string format. Numerical attributes with missing values (defined as "?") is automatically replaced with the mean value of the corresponding attribute. Categorical missing values are, similarly, replaced with the mode value of the variable. If necessary, attributes can be excluded using the graphical user interface.

N-fold cross validation with optional stratification is also implemented to make experiment evaluation easy.

3.2. Evaluation and Optimization

Most machine learning techniques are designed to optimize accuracy (ACC). At the same it is argued 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 since these assumptions rarely are true for real world problems [6]. Instead other metrics such as area under roc curve (AUC) or Brier score (BRI) are often recommended.

G-REX has two main features to handle this; first ACC, AUC and BRI are reported for all experiments. Secondly, G-REX facilitates optimization of each of these metrics; i.e. there are three built-in fitness functions. Experimental results show (not surprisingly) that it is favorable to optimize the metric that are used for evaluation. In practice, this translates to that the data miner should optimize the metric that fits the problem at hand best, for example if a model with a good ability to rank instances is needed, AUC should be optimized.

G-REX fitness functions are, more specifically, based on two properties. The first part is a reward for achieving good result on the optimization metric and the second part is a punishment for each node included in the tree. This size punishment was initially used to keep evolved models fairly small, in order to facilitate comprehensibility when performing rule extraction. For general classification and regression, the size punishment will make the evolution favor smaller models, something that often will produce models that will generalize better, simply by avoiding over-fitting. In this way, G-REX will balance the size and the performance and will not add unnecessary elements to the rule. Previous studies [7, 8] have shown that this size punishment allows G-REX to evolve models that are considerably smaller than the corresponding decision trees created using standard algorithms like C4.5 or CART, while still having a comparable accuracy.

Even if the size punishment is very effective in reducing the size of the evolved models, they can still on occasion (due to the nature of evolution) contain introns. An intron is either a copy of an important part of, for instance, a rule or just meaningless nonsense used by the evolution to protect valuable parts from destruction due to crossover or mutation operations. Introns are, consequently, a crucial part of the evolution but will only add unnecessary complexity to the final rule. G-REX can, automatically, remove introns and other useless parts of a tree with a general simplification process (also based on GP) which can be applied to any type of model that can be represented in G-REX.

3.3. Classification models

The most important classification models implemented in G-REX are presented below. Leaf nodes present the predicted class, a probability estimation *P* (calculated using a *Laplace* esitmate) and the number of training instances *N* that are classified by the node, see Figure 1.

Decision trees. The basic classification model used by G-REX is a typical decision tree with binary splits based on a single condition. Optionally, G-REX also allows complex conditions (for all models containing conditions) where several variables can be combined using conjunctions and disjunctions; for a sample G-REX decision tree see Figure 1 below.

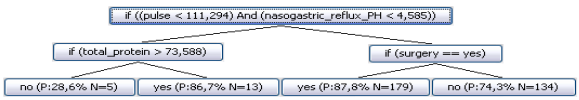

Figure 1 – Tree with complex conditions

Decision List. A decision list is similar to a decision tree with the exception that each node must predict at least one class. Normally, decision lists are created to classify as many instances as possible early in the list. In this way, large parts of the data set will be classified using simple and general expressions, while more complex expression will be used to define the

[1]http://www.hb.se/ida/aim/grex

uncharacteristic instances of the data set. In G-REX, this is done with a special fitness function that tries to maximize the accuracy while maximizing brevity (i.e. minimizing the average number of conditions needed to be checked to classify all instances). For more details see [9].

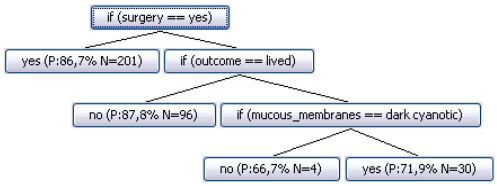

Figure 2 - Basic decision list model

***k*-Nearest Neighbor.** G-REX *k*-NN implementation uses the Euclidian distance metric and can automatically choose *k* using cross-validation, apply distance voting and/or evolve attribute weights. Two hybrid extensions have also been made to the standard *k*-NN model, allowing G-REX to evolve decision trees with *k*-NN nodes. The first extension, named *Global-kNN* is simply a decision tree which allows different values of *k* for different parts of the data set; see Figure 3. The second extension *Local-kNN* is similar to the first extension, but here the *k*-NN nodes only consider the *k* nearest neighbors of the training instances actually in the specific leaf; i.e. the instances that meet all conditions leading to the node.

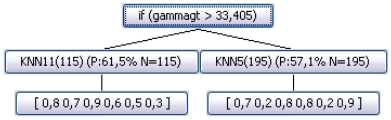

Figure 3 - Hybrid kNN with attribute weights

Fuzzy Logic. When evolving fuzzy logic rules, G-REX uses a fuzzification method and a tree based model presented [10]. Here continuous attributes are fuzzyfied automatically using a clustering technique based on the K-Means algorithm, while categorical attributes are treated in a normal Boolean fashion. For a sample fuzzy classification tree; see Figure 4 below.

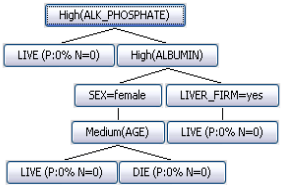

Figure 4 - Fuzzy tree model

A fuzzy set is represented as a node with two children, where the left node is related to the value of being a member of the fuzzy set while the right child is assigned the value of not being a member. A path of fuzzy membership values from the root to a leaf is multiplied and all paths leading to the same class are summarized. An instance is classified as the class which gets the highest membership value.
The right child (i.e. not being a member) of categorical attributes are omitted as these always would result in a 0 value and thus give the whole path a value of 0.

3.4. Regression models

For regression problems, G-REX has three types of regression models which are discussed briefly below, for more details see [11]. Time series prediction can also be performed, if attribute delays are present in the data file.

Leaf nodes present a prediction, the absolute percentage error *P* and the number of instances *N* reaching the node. Predictions, real values and the absolute percentage error can also be shown graphically in a diagram which is linked to the evolved model. A mouse click on a leaf node of the model will highlight all predictions in the diagram made by that node.

Simple regression. Similar to most regression trees, G-REX simplest regression model consist of a normal decision tree with constant values as leaves. This model can also use complex conditions and optionally allows comparisons between attributes. For a sample model, see Figure 5 below.

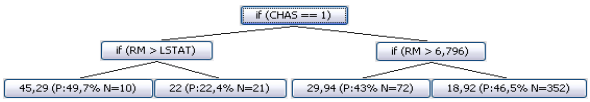

Figure 5 - Simple regression model

***k*-NN regression.** All *k*-NN models presented in the classification section can also be applied to regression problems. When used for regression, the *k*-NN node predicts the average value of the *k* nearest neighbors. Distance and attributes weights can also be applied.

Hybrid linear regression. The hybrid linear regression models is similar to simple regression, with the exception of that linear regression are used as leaves. Each linear regression ($y=k*x+m$) contains a single random attribute x and is optimized using least square for the training instances reaching that node. The least square optimization is done to guide the GP process and thus decreases the evolution time.

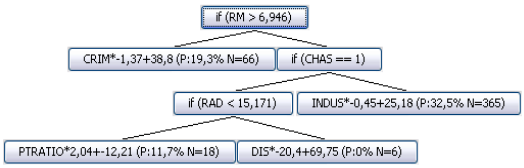

Figure 6 - Hybrid linear regression model

Time series. If tap delays of appropriate attributes are included in the data file, G-REX can also be used for time series prediction. All of the previously presented regression models are also applicable for time series regression. G-REX, in addition, includes a special time series model which allows arithmetic expression including attributes to be used as leaves; see Figure 7 below.

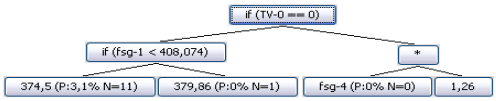

Figure 7 - Time series model

3.5. Defining models

All nodes described above can be combined into an arbitrary model using G-REX model definition languages (GDL). A GDL file is built using nodes defined by their java class name and four constructions: head function (HF), function (F), terminal function (TF), and terminal (T). HF defines the starting node of the model and a TF is a function that can have terminals in both leaves (information needed for creating trees of a certain depth). F:s and TF:s are defined by a name and a semicolon separated argument list. Each argument is in itself a comma separated list of allowed nodes. Terminals are defined by their name and a comma separated list of which nodes they are allowed to be switched with during evolution. The GDL file for the decision list model is presented in Figure 2, where it is also shown how the model could be extended to also allow kNN nodes.

```
1 HF;If;
2 F;If;Less,Greater,Equals;Pred;If,Pred;
3 TF;Less;Var;Val;
4 TF;Greater;Var;Val;
5 TF;Equals;Var;Val;
6 T;Var;Var;
7 T;Val;Val;
8 T;Pred;Pred;

Changes needed to allow KNN nodes
2 F;If;Less,Greater,Equals;Pred;If,Pred,KNN;
8 T;Pred;Pred,KNN;
9 T;KNN;KNN,Pred;
```

Figure 8 - Model definition for decision lists

4. Contribution

We have presented G-REX, a new versatile data mining framework based on Genetic Programming. G-REX includes much functionality specific to data mining such as preprocessing, evaluation- and optimization methods, a multitude of predefined classification and regression models. The main strength is however, the ease of how all the predefined functionality (optimization functions, nodes and models) can be modified, extended and combined with minimal effort. G-REX is also available in a special Weka package adding much useful evolutionary functionality. Both applications and a demo video can be downloaded from www.hb.se/ida/aim/grex

5. References

1. DGPF-project, *Distributed Genetic Programming Framework* 2008, SourceForge.net, Available from: dgpf.sourceforge.net.
2. Gagné, C. and Parizeau, M., *Open BEAGLE*. 2008, Available from: beagle.gel.ulaval.ca/.
3. Johansson, U., Konig, R., and Niklasson, L., *Rule Extraction from Trained Neural Networks using Genetic Programming*, in *Joint 13th International Conference on Artificial Neural Networks,* 2003: Istanbul, Turkey.
4. Johansson, U., *Obtaining accurate and comprehensible data mining models : an evolutionary approach*. Linköping studies in science and technology. Dissertations, 1086. 2007, Linköping: Department of Computer and Information Science, Linköpings universitet. xii, 254 s.
5. WEKA-project, *WEKA*. 2008, University of Waikato, Available from: www.cs.waikato.ac.nz/~ml/weka/.
6. Provost, F., Fawcett, T., and Kohavi, R. *The Case Against Accuracy Estimation for Comparing Induction Algorithms*. 15th Int. Conf. on Machine Learning. 2008 p. 445-453
7. Johansson, U., Konig, R., and Niklasson, L. *Automatically balancing accuracy and comprehensibility in predictive modeling*. in 8th International Conference on Information Fusion (FUSION). 2005 p. 7 pp.
8. Konig, R., Johansson, U., and Niklasson, L. *Using Genetic Programming to Increase Rule Quality*. in Proceedings of the Twenty-First International FLAIRS Conference. 2008. Coconut Grove, Florida, USA: AAAI Press p. 288-293, ISBN: 978-1-57735-365-2.
9. Konig, R., Johansson, U., and Niklasson, L., Increasing rule extraction comprehensibility. *International Journal of Information Technology and Intelligent Computing*, 2006. **1**(2): p. 303-314.
10. Eggermont, J. *Evolving Fuzzy Decision Trees with Genetic Programming and Clustering* in Proceedings of the Fifth European Conference on Genetic Programming (EuroGP'02). 2002: LNCS p. 204-237.
11. Konig, R., Johansson, U., and Niklasson, L. *Genetic Programming – a Tool for Flexible Rule Extraction*. in IEEE Congress on Evolutionary Computation , CEC. 2007 p. 1304-1310, ISBN: 1-4244-1340-0.