Genetic Programming for Automatically Constructing Data Mining Algorithms

Alex A. Freitas
University of Kent, UK

Gisele L. Pappa
University of Kent, UK

INTRODUCTION

At present there is a wide range of data mining algorithms available to researchers and practitioners (Witten & Frank, 2005; Tan et al., 2006). Despite the great diversity of these algorithms, virtually all of them share one feature: they have been *manually* designed. As a result, current data mining algorithms in general incorporate human biases and preconceptions in their designs.

This article proposes an alternative approach to the design of data mining algorithms, namely the automatic creation of data mining algorithms by means of Genetic Programming (GP) (Pappa & Freitas, 2006). In essence, GP is a type of Evolutionary Algorithm – i.e., a search algorithm inspired by the Darwinian process of natural selection – that evolves computer programs or executable structures.

This approach opens new avenues for research, providing the means to design novel data mining algorithms that are less limited by human biases and preconceptions, and so offer the potential to discover new kinds of patterns (or knowledge) to the user. It also offers an interesting opportunity for the automatic creation of data mining algorithms tailored to the data being mined.

BACKGROUND

An Evolutionary Algorithm (EA) is a computational problem-solving method inspired by the process of natural selection. In essence, an EA maintains a population of "individuals", where each individual represents a candidate solution to the target problem. EAs are iterative generate-and-test procedures, where at each "generation" (iteration) a population of individuals is generated and each individual has its "fitness" computed. The fitness of an individual is a measure of the quality of its corresponding candidate solution. The higher the fitness of an individual, the higher the probability that the individual will be selected to be a "parent" individual. Certain operations (often "crossover" and/or "mutation" operations inspired by their natural counterparts) are applied to the selected parent individuals in order to produce "children" individuals. The important point is that, since the children are in general produced from parents selected based on fitness, the children (new candidate solutions) tend to inherit parts of the good solutions of the previous generation, and the population as a whole gradually evolves to fitter and fitter individuals (better and better solutions to the target problem). For a comprehensive review of EAs in general the reader is referred to (De Jong, 2006; Eiben & Smith, 2003), and a comprehensive review of EAs for data mining can be found in (Freitas, 2002).

The basic principle of EAs – i.e., artificial evolution of candidate solutions, where parent solutions are selected based on their fitness in order to create new children solutions – still holds in Genetic Programming (GP). The main distinguishing feature of GP – by comparison with other types of EA – is that the candidate solutions represented by GP individuals are (or at least should be) computer programs or executable structures. GP has been an active research field for about 15 years (Koza, 1992; Banzhaf et al., 1998; Langdon & Poli, 2002; Koza, 2003), and Koza (2006) reports 36 instances where GP discovered a solution infringing or duplicating some patent, which led Koza to claim that GP is an automated invention machine that routinely produces human-competitive results. However, the creation of a GP system for automatically evolving a full data mining algorithm, as proposed in this article, is a new research topic which is just now

starting to be systematically explored, as discussed in the next section.

MAIN FOCUS

The problem of automatically evolving a data mining algorithm to solve a given data mining task is very challenging, because the evolved algorithm should be generic enough to be applicable to virtually any data set that can be used as input to that task. For instance, an evolved algorithm for the classification task should be able to mine any classification data set (i.e., a data set having a set of records, each of them containing a class attribute and a set of predictor attributes – which can have different data types); an evolved clustering algorithm should be able to mine any clustering data set, etc.

In addition, the automatic evolution of a fully-fledged data mining algorithm requires a GP method that not only is aware of the basic structure of the type of data mining algorithm to be evolved, but also is capable of avoiding fatal programming errors – e.g. avoiding infinite loops when coping with "while" or "repeat" statements. Note that most GP systems in the literature do not have difficulties with the latter problem because they cope only with relatively simple operations (typically mathematical and logical operations), rather than more sophisticated programming statement such as "while" or "repeat" loops. To cope with the two aforementioned problems, a promising approach is to use a grammar-based GP system, as discussed next.

Grammar-Based Genetic Programming

Grammar-Based Genetic Programming (GGP) is a particular type of GP where a grammar is used to create individuals (candidate solutions). There are several types of GGP (Wong & Leung, 2000; O'Neill & Ryan, 2003). A type of GGP particularly relevant for this article consists of individuals that directly encode a candidate program in the form of a tree, namely a derivation tree produced by applying a set of derivation steps of the grammar. A derivation step is simply the application of a production rule to some non-terminal symbol in the left-handed side of the rule, producing the (non-terminal or terminal) symbol in the right-handed side of the rule. Hence, each individual is represented by a derivation tree where the leaf nodes are terminal symbols of the grammar and the internal nodes are the non-terminal symbols of the grammar.

The use of such a grammar is important because it not only helps to constrain the search space to valid algorithms but also guides the GP to exploit valuable background knowledge about the basic structure of the type of data mining algorithm being evolved (Pappa & Freitas, 2006; Wong & Leung, 2000).

In the context of the problem of evolving data mining algorithms the grammar incorporates background knowledge about the type of data mining algorithm being evolved by the GP. Hence, broadly speaking, the non-terminal symbols of the grammar represent high-level descriptions of the major steps in the pseudo-code of a data mining algorithm, whilst the terminal symbols represent a lower-level implementation of those steps.

A New Grammar-Based GP System for Automatically Evolving Rule Induction Algorithms

The previously discussed ideas about Grammar-Based Genetic Programming (GGP) were used to create a GGP system that automatically evolves a rule induction algorithm, guided by a grammar representing background knowledge about the basic structure of rule induction algorithms (Pappa & Freitas, 2006), (Pappa 2007). More precisely, the grammar contains two types of elements, namely:

a. general programming instructions – e.g. *if-then* statements, *for/while* loops; and
b. procedures specifying major operations of rule induction algorithms – e.g., procedures for initializing a classification rule, refining a rule by adding or removing conditions to/from it, selecting a (set of) rule(s) from a number of candidate rules, pruning a rule, etc.

Hence, in this GGP each individual represents a candidate rule induction algorithm, obtained by applying a set of derivation steps from the rule induction grammar. The terminals of the grammar correspond to modular blocks of Java programming code, so each individual is actually a Java program implementing a full rule induction algorithm.

This work can be considered a major "case study" or "proof of concept" for the ambitious idea of automati-

cally evolving a data mining algorithm with genetic programming, and it is further described below. In any case, it should be noted that the proposed idea is generic enough to be applicable to other types of data mining algorithms – i.e., not only rule induction algorithms – with proper modifications in the GGP. The authors' motivation for focusing on rule induction algorithms was that this type of algorithm usually has the advantage of discovering comprehensible knowledge, represented by IF-THEN classification rules (i.e., rules of the form IF (conditions) THEN (predicted class)) that are intuitively interpretable by the user (Witten & Frank, 2005). This is in contrast to some "black box" approaches such as support vector machines or neural networks. Note that the comprehensibility of discovered knowledge is usually recognized as an important criterion in evaluating that knowledge, in the context of data mining (Fayyad et al., 1996; Freitas, 2006).

In order to train the GGP for evolving rule induction algorithms, the authors used a "meta-training set" consisting of six different data sets. The term meta-training set here refers to a set of data sets, all of which are using simultaneously for the training of the GGP – since the goal is to evolve a generic rule induction algorithm. This is in contrast with the normal use of a GP system for rule induction, where the GP is trained with a single data set – since the goal is to evolve just a rule set (rather than algorithm) specific to the given data set. The fitness of each individual – i.e., each candidate rule induction algorithm – is, broadly speaking, an aggregated measure of the classification accuracy of the rule induction algorithm over all the six data sets in the meta-training set. After the evolution is over, the best rule induction algorithm generated by the GGP was then evaluated on a "meta-test set" consisting of five data sets, none of which were used during the training of the GGP. The term meta-test set here refers to a set of data sets, all of which are used to evaluate the generalization ability of a rule induction algorithm evolved by the GGP. Overall, cross-validation results showed that the rule induction algorithms evolved by the GGP system were competitive with several well-known rule induction algorithms – more precisely, CN2, Ripper and C4.5Rules (Witten & Frank, 2005) – which were manually designed and refined over decades of research.

Related Work on GP Systems for Evolving Rule Induction Algorithms

The authors are aware of only two directly relevant related works, as follows. Suyama et al. (1998) proposed a GP system to evolve a classification algorithm. However, the GP system used an ontology consisting of coarse-grained building blocks, where a leaf node of the ontology is a full classification algorithm, and most nodes in the ontology refer to changes in the dataset being mined. Hence, this work can be considered as evolving a good combination of a modified dataset and a classification algorithm selected (out of predefined algorithms) for that modified dataset. By contrast, the methods proposed in (Pappa & Freitas, 2006) are based on a grammar combining finer-grained components of classification algorithms, i.e., the building blocks include programming constructs such as "while" and "if" statements, search strategies, evaluation procedures, etc., which were not used in (Suyama et al., 1998).

In addition, Wong (1998) proposed a grammar-based GP to evolve a classification algorithm adapted to the data being mined. However, that work focused on just one existing rule induction algorithm (viz. FOIL) of which the GP evolved only one component, namely its evaluation function. By contrast, the method proposed in (Pappa & Freitas, 2006) had access to virtually all the components of a classification algorithm, which led to the construction of new rule induction algorithms quite different from existing ones in some runs of the GP.

FUTURE TRENDS

The discussion on the previous section focused on the automatic evolution of rule induction algorithms for classification, but of course this is by no means the only possibility. In principle other types of data mining algorithms can also be automatically evolved by using the previously described approach, as long as a suitable grammar is created to guide the search of the Genetic Programming system. Since this kind of research can be considered pioneering, there are innumerous possibilities for further research varying the type of data mining algorithm to be evolved – i.e., one could evolve other types of classification algorithms, clustering algorithms, etc.

CONCLUSION

This article proposed the idea of designing a Grammar-based Genetic Programming (GGP) system to automatically evolve a data mining algorithm, and briefly discussed, as a practical example of the effectiveness of this approach, a new GGP system that automatically creates rule induction algorithms. In this GGP system the evaluation of candidate rule induction algorithms was performed by using a "meta-training set" consisting of six datasets, in order to generate rule induction algorithms robust across a number of different datasets. This robustness was indeed obtained, since the automatically-evolved rule induction algorithms were competitive with several well-known manually-designed rule induction algorithms in five datasets used in the "meta-test set", representing datasets unseen during the evolution of the GGP.

An alternative research direction, whose experiments are ongoing, is to use a meta-training and a meta-test set having just one dataset. In this approach a subset of the single data set being mined is used in the meta-training set during the GGP evolution, and the remaining subset of the data (i.e., the set of records or data instances not used in the meta-training set) is used in the meta-test set, to evaluate the generalization ability of the rule induction algorithm output by the GGP. In this case the GGP works in a way that actually evolves a rule induction algorithm tailored to the data being mined, which offers the potential to automatically construct data mining algorithms customized to any dataset provided by the user. Note that this represents a new level of automation in data mining. The potentially significant benefit of this automation is clear because the number of datasets to be mined (as well as the number of users) is much greater than the number of (human) data mining algorithm designers, and so one can hardly expect that human designers would have time to manually design algorithms customized to the data being mined.

REFERENCES

Banzhaf, W., Nordin, P., Keller, R.E. and Francone, F.D. (1998) *Genetic Programming – an Introduction: on the automatic evolution of computer programs and its applications*. Palo Alto, CA: Morgan Kaufmann.

De Jong, K. (2006). *Evolutionary Computation: a unified approach.* MIT Press.

Eiben, E.A. & Smith, J.E. (2003). *Introduction to Evolutionary Computation*. Berlin: Springer.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery: an overview. In U.M Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 1-34). Palo Alto, CA: AAAI/MIT.

Freitas, A.A. (2002) *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Berlin: Springer.

Freitas, A.A. (2006) Are we really discovering "interesting" knowledge from data? *Expert Update, 9(1), Autumn 2006*, 41-47.

Koza, J.R. (1992). *Genetic Programming: on the programming of computers by means of natural selection*. Boston, MA: MIT Press.

Koza, J.R. (2003). *Genetic Programming IV: routine human-competitive machine intelligence*. Berlin: Springer.

Koza, J.R. (2007) http://www.genetic-programming.org/, visited in May 2007.

Langdon, W.B. & Poli, R. (2002). *Foundations of Genetic Programming*. Berlin: Springer.

O'Neill, M. & Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in Arbitrary Language*. Amsterdam: Kluwer.

Pappa, G.L. & Freitas, A.A. (2006). Automatically evolving rule induction algorithms. In *Proc. 17th European Conf. on Machine Learning (ECML-2006)*, *Lecture Notes in Artificial Intelligence, No. 4212* (pp. 341-352). Berlin: Springer.

Pappa, G.L. (2007) Automatically evolving rule induction algorithms with grammar-based genetic programming. *PhD Thesis*. Computing Laboratory, University of Kent, UK.

Suyama, A., Negishi, N. and Yamaguchi, T. (1998). CAMLET: a platform for automatic composition of inductive learning systems using ontologies. In *Proc. Pacific Rim Int. Conf. on AI*, 205-215.

Tan, P.-N., Steinbach, M. and Kumar, V. (2006). *Introduction to Data Mining*. Addison-Wesley.

Witten, I.H. & Frank, E. (2005). *Data Mining: practical machine learning tools and techniques*, 2nd Ed. San Francisco, CA: Morgan Kaufmann.

Wong, M.L. (1998). An adaptive knowledge-acquisition system using genetic programming. *Expert Systems with Applications 15 (1998)*, 47-58.

Wong, M.L. & Leung, K.S. (2000). *Data Mining Using Grammar-Based Genetic Programming and Applications*. Amsterdam: Kluwer.

KEY TERMS

Classification: A type of data mining task where the goal is essentially to predict the class of a record (data instance) based on the values of the predictor attributes for that example. The algorithm builds a classifier from the training set, and its performance is evaluated on a test set (see the definition of training set and test set below).

Classification Rule: A rule of the form IF (conditions) THEN (predicted class), with the meaning that, if a record (data instance) satisfies the conditions in the antecedent of the rule, the rule assigns to that record the class in the consequent of the rule.

Evolutionary Algorithm: An iterative computational problem-solving method that evolves good solutions to a well-defined problem, inspired by the process of natural selection.

Genetic Programming: A type of evolutionary algorithm where the solutions being evolved by the algorithm represent computer programs or executable structures.

Rule Induction Algorithm: A type of data mining algorithm that discovers rules in data.

Test Set: A set of records whose class is unknown by a classification algorithm, used to measure the predictive accuracy of the algorithm after it is trained with records in the training set.

Training Set: A set of records whose class is known, used to train a classification algorithm.