CHAPTER 1

Generating Automatic Projections by Means of Genetic Programming

C. ESTÉBANEZ and R. ALER

Universidad Carlos III de Madrid, Spain

1.1 INTRODUCTION

The aim of inductive machine learning (ML) is to generate models that can make predictions from analysis of data sets. These data sets consist of a number of instances or examples, each example described by a set of attributes. It is known that the quality or relevance of the attributes of a data set is a key issue when trying to obtain models with a satisfactory level of generalization. There are many techniques of feature extraction, construction, and selection [1] that try to improve the representation of data sets, thus increasing the prediction capabilities of traditional ML algorithms. These techniques work by filtering nonrelevant attributes or by recombining the original attributes into higher-quality ones. Some of these techniques were created in an automatic way by means of genetic programming (GP).

GP is an evolutionary technique for evolving symbolic programs [2]. Most research has focused on evolving functional expressions, but the use of loops and recursion has also been considered [3]. Evolving circuits are also among the successes of GP [4]. In this work we present a method for attribute generation based on GP called the GPPE (genetic programming projection engine). Our aim is to evolve symbolic mathematical expressions that are able to transform data sets by representing data on a new space, with a new set of attributes created by GP. The goal of the transformation is to be able to obtain higher accuracy in the target space than in the original space. The dimensions of the new data space can be equal to, larger, or smaller than those of the original. Thus, we also intend that GPPE be used as a dimension reduction technique as

Optimization Techniques for Solving Complex Problems, Edited by Enrique Alba, Christian Blum, Pedro Isasi, Coromoto León, and Juan Antonio Gómez

well as creating highly predictive attributes. Although GPPE can either increase or reduce dimensionality, the work presented in this chapter focuses on reducing the number of dimensions dramatically while attempting to improve, or at least maintain, the accuracy obtained using the original data.

In the case of dimension reduction, the newly created attributes should contain all the information present in the original attributes, but in a more compact way. To force the creation of a few attributes with a high information content, we have established that the data in the projected space must follow a nearly linear path. To test GPPE for dimensionality reduction, we have applied it to two types of data mining domains: classification and regression. In classification, linear behavior will be measured by a fast classification algorithm based on selecting the nearest class centroid. In regression, linear behavior will be determined by simple linear regression in the projected space.

GP is very suitable for generating feature extractors, and some work has been done in this field. In the following section we overview briefly some approaches proposed in the literature. Then, in Section 1.4 we focus on GPPE, which can be used in both the classification and regression domains, and we show some experimental results in Section 1.5. We finish with our conclusions and some suggestions for future work.

1.2 BACKGROUND

There are many different constructive induction algorithms, using a wide variety of approaches. Liu et al. [1] provide a good starting point for the exploration of research into feature extraction, construction, and selection. Their book compiles contributions from researchers in this field and offers a very interesting general view. Here we discuss only works that use GP or any other evolutionary strategy, and we focus on those that are among the most interesting for us because they bear some resemblance to GPPE.

Otero et al. [5] use typed GP for building feature extractors. The functions are arithmetic and relational operators, and the terminals are the original (continuous) attributes of the original data set. Each individual is an attribute, and the fitness function uses the information gain ratio. Testing results using C4.5 show some improvements in some UCI domains. In Krawiec's work [6], each individual contains several subtrees, one per feature. C4.5 is used to classify in feature space. Their work allows us to cross over subtrees from different features.

Shafti and Pérez [7] discuss the importance of applying GA as a global search strategy for constructive induction (CI) methods and the advantages of using these strategies instead of using classic greedy methods. They also present MFE2/GA, a CI method that uses GA to search through the space of different combination of attribute subsets and functions defined over them. MFE2/GA uses a nonalgebraic form of representation to extract complex interactions between the original attributes of the problem.

Kuscu [8] introduced the GCI system. GCI is a CI method based on GP. It is similar to GPPE in the sense that it uses basic arithmetic operators and the fitness is computed measuring the performance of an ML algorithm (a quick-prop net) using the attributes generated. However, each individual represents a new attribute instead of a new attribute set. In this way, GCI can only generate new attributes that are added to the original ones, thus increasing the dimensionality of the problem. The possibility of reducing the number of attributes of the problem is mentioned only as possible and very interesting future work.

Hu [9] introduced another CI method based on GP: GPCI. As in GCI, in GPCI each individual represents a newly generated attribute. The fitness of an individual is evaluated by combining two functions: an absolute measure and a relative measure. The absolute measure evaluates the quality of a new attribute using a gain ratio. The relative measure evaluates the improvement of the attribute over its parents. A function set is formed by two Boolean operators: AND and NOT. GPCI is applied to 12 UCI domains and compared with two other CI methods, achieving some competitive results.

Howley and Madden [10] used GP to evolve kernels for support vector machines. Both scalar and vector operations are used in the function set. Fitness is computed from SVM performance using a GP-evolved kernel. The hyperplane margin is used as a tiebreaker to avoid overfitting. Although evolved kernels are not forced by the fitness function to satisfy standard properties (such as Mercer's property) and therefore the evolved individuals are not proper kernels, results in the testing data sets are very good compared to those of standard kernels. We believe that evolving proper distance functions or kernels is difficult because some properties (such as transitivity or Mercer's property) are not easy to impose on the fitness computation.

Eads et al. [11] used GP to construct features to classify time series. Individuals were made of several subtrees returning scalars (one per feature). The function set contained typical signal-processing primitives (e.g., convolution), together with statistical and arithmetic operations. SVM was then used for classification in feature space. Cross-validation on training data was used as a fitness function. The system did not outperform the SVM, but managed to reduce dimensionality. This means that it constructed good features to classify time series. However, only some specific time series domains have been tested. Similarly, Harvey et al. [12] and Szymanski et al. [13] assemble image-processing primitives (e.g., edge detectors) to extract multiple features from the same scene to classify terrains containing objects of interest (i.e., golf courses, forests, etc.). Linear fixed-length representations are used for the GP trees. A Fisher linear discriminant is used for fitness computation. Results are quite encouraging but are restricted to image-processing domains.

Results from the literature show that, in general, the GP projection approach has merit and obtains reasonable results, but that more research is needed. New variations of the idea and more domains should be tested. Regression problems are not considered in any of the works reviewed, and we believe that a lot more research on this topic is also needed.

1.3 DOMAINS

In this chapter we are interested in applying GPPE to two classical prediction tasks: classification and regression. We have used bankruptcy prediction as the classification domain and IPO underpricing prediction as the regression domain.

1.3.1 Bankruptcy Prediction

In general terms, the bankruptcy prediction problem attempts to determine the financial health of a company, and whether or not it will soon collapse. In this chapter we use a data set provided and described by Vieira et al. [14]. This data set studies the influence of several financial and economical variables on the financial health of a company. It includes data on 1158 companies, half of which are in a bankruptcy situation (class 0) and the rest of which have good financial health (class 1). Companies are characterized by 40 numerical attributes [14]. For validation purposes we have divided the data set into a training set and a test set, containing 766 (64%) and 400 (36%) instances, respectively.

1.3.2 IPO Underpricing Prediction

IPO underpricing is an interesting and important phenomenon in the stock market. The academic literature has long documented the existence of important price gains in the first trading day of initial public offerings (IPOs). That is, there is usually a big difference between the offering price and the closing price at the end of the first trading day. In this chapter we have used a data set composed of 1000 companies entering the U.S. stock market for the first time, between April 1996 and November 1999 [15]. Each company is characterized by seven explicative variables: underwriter prestige, price range width, price adjustment, offer price, retained stock, offer size, and relation to the tech sector. The target variable is a real number which measures the profits that could be obtained by purchasing the shares at the offering price and selling them soon after dealing begins. For validation purposes we have divided the data set into a training set and a test set, containing 800 (80%) and 200 (20%) instances, respectively.

1.4 ALGORITHMIC PROPOSAL

In this section we describe the genetic programming projection engine (GPPE). GPPE is based on GP. Only a brief summary of GP is provided here. The reader is encouraged to consult Koza's book [2] for more information.

GP has three main elements:

1. A population of individuals, in this case, computer programs
2. A fitness function, used to measure the goodness of the computer program represented by the individual

3. A set of genetic operators, the basic operators being reproduction, mutation, and crossover

The GP algorithm enters into a cycle of fitness evaluation and genetic operator application, producing consecutive generations of populations of computer programs, until a stopping criterion is satisfied. Usually, GP is stopped when an optimal individual is found or when a time limit is reached. Every genetic operator has the probability of being applied each time we need to generate an offspring individual for the next generation. Also, GP has many other parameters, the most important ones being the size of the population (M) and the maximum number of generations (G).

GPPE is a GP-based system for computing data projections with the aim of improving prediction accuracy in prediction tasks (classification and regression). The training data E belong to an N-dimensional space U, which will be projected into a new P-dimensional space V. In classification problems the goal of the projection is that classification becomes easier in the new space V. By "easier" we mean that data in the new space V are as close to linear separability as possible. Similarly, in regression problems the aim is to project data so that they can be approximated by a linear regression. To include both cases, we will talk about linear behavior. P can be larger, equal to, or smaller than N. In the latter case, in addition to improving prediction, we would also reduce the number of dimensions.

GPPE uses standard GP to evolve individuals made of P subtrees (as many as the number of dimensions of the projected space V). Fitness of individuals is computed by measuring the degree of linear behavior of data in the space projected by using the individual as a projection function from U to V. The system stops if 100% linear behavior has been achieved (i.e., a 100% classification rate or 0.0 error in regression) or if the maximum number of generations is reached. Otherwise, the system outputs the highest-fitness individual (i.e., the most accurate individual on the training data). Algorithm 1.1 displays pseudocode for GPPE operation. For the implementation of our application, we have used Lilgp 1.1, a software package for GP developed at Michigan State University by Douglas Zongker and Bill Punch.

Next, we describe the main GPPE elements found in all GP-based systems: the terminal and function set for the GP individuals, the structure of the GP individuals themselves, and the fitness function.

1.4.1 Terminal and Function Set

The terminal and function set is composed of the variables, constants, and functions required to represent mathematical projections of data. For instance, $(v_0, v_1) = (3 * u_0 + u_1, u_2^2)$ is a projection from three to two dimensions comprising the following functions and terminals:

$$\text{functions} = \{+, *, ^2\}$$

$$\text{terminals} = \{3, u_0, u_1, u_2, 2\}$$

Algorithm 1.1 Pseudocode for GPPE Operation

```
P = Initial population made of random projections;
generation = 0; stopCondition = FALSE;
while (generation < maxGenerations) AND (NOT stopCondition) do
 for each projection p ∈ P do
  f_p = fitness(p);
  if (f_p = perfectFitness) then stopCondition = TRUE;
 end for
 P' = geneticOperatorsApplication(P, f);
 P = P';
 generation = generation + 1;
end while
```

The set of functions and terminals is not easy to determine: It must be sufficient to express the problem solution. But if the set is too large, it will increase the search space unnecessarily. In practice, different domains will require different function and terminal sets, which have to be chosen by the programmer. We consider this to be an advantage of GP over other methods with fixed primitives because it allows us to insert some domain knowledge into the learning process. At this point in our research, we have tested some generic sets appropriate for numerical attributes. In the future we will analyze the domains to determine which terminals and functions are more suitable.

This generic terminal set contains the attributes of the problem expressed in coordinates of U $(u_0, u_1, \ldots, u_N)$, and the ephemeral random constants (ERCs). An ERC is equivalent to an infinite terminal set of real numbers. The generic functions we have used in GPPE so far are the basic arithmetical functions $+$, $-$, *, and /, and the square and square root. We have judged them to be sufficient to represent numerical projections, and experimental data have shown good results.

1.4.2 Individuals

Projections can be expressed as

$$(v_0 \cdots v_P) = (f_1(u_0 \cdots u_N), \ldots, f_P(u_0 \cdots u_N))$$

To represent them, individuals are made of P subtrees. Every subtree number i represents function f_i, which corresponds to coordinate v_i in the target space. All subtrees use the same set of functions and terminals.

1.4.3 Fitness Function

The fitness function evaluates an individual by projecting the original data and determining the degree of linear behavior in the target space. For this task we

designed two different fitness functions: one for classification problems and one for regression problems.

Classification The classification fitness function uses a centroid-based classifier. This classifier takes the projected data and calculates a centroid for each class. Centroids are the centers of mass (baricenters) of the examples belonging to each class. Therefore, there will be as many centroids as classes. The centroid-based classifier assigns to every instance the class of the nearest centroid.

This function tries to exert a selective pressure on the projections that forces every instance belonging to the same class to get close. The great advantage of this function is that it is fast and simple. We call this function CENTROIDS.

To avoid overfitting, fitness is actually computed using cross-validation on the training data. That is, in every cross-validation cycle the centroids-based classifier is trained on some of the training data and tested on the remaining training data. Training the centroids means computing the baricenters, and testing them means using them to classify the part of the training data reserved for testing. The final fitness is the average of all the cross-validation testing results.

Regression In this case, linear behavior is defined as data fitting a hyperplane, so in this case, the goal is to adjust projected data to a hyperplane as closely as possible. For the regression tasks, a simple linear regression algorithm is used to compute fitness. More precisely, the fitness is the error produced by the linear regression on the projected data. This error is measured by the normalized mean-square error (NMSE).

1.5 EXPERIMENTAL ANALYSIS

In this section we test GPPE in the classification and regression domains, described in Section 1.3. We want to show that GPPE can help in both tasks. We have executed 100 GP runs in both the classification domain (bankruptcy prediction) and the regression domain (IPO underpricing prediction), each with the parameters displayed in Table 1.1. They were found empirically by running GPPE five times on each domain. The fitness function CENTROIDS was used in the classification domain, and REGRESSION in the regression domain. Tenfold cross-validation was used to compute the training fitness, to avoid overfitting the training data.

Every GP run involves running GPPE on the same problem but with a different random seed. However, typical use of GP involves running the GP engine several times and selecting the best individual according to its training fitness. We have simulated this procedure by using bootstrapping [16] on these 100 samples, as follows:

- Repeat B times:
  - Select R samples from the 100 elements data set.

TABLE 1.1 Common GPPE Parameters for Both Experimentation Domains

Parameter	Value
Max. generations (G)	100
Population size (M)	500 (bankruptcy)/5000 (IPO)
Max. nodes per tree	75
Initialization method	Half and half
Initialization depth	2–6
Number of genetic operators	3
Genetic operator 1	Reproduction rate = 15% Selection = fitness proportion
Genetic operator 2	Crossover rate = 80% Selection = tournament Tournament size = 4
Genetic operator 3	Mutation selection = tournament Tournament size = 2

- From the set of R samples, select the best sample i, according to its training fitness.
- Return the accuracy of i, according to the test set.

In this case, $B = 100$ and $R = 5$. This means that we have simulated 100 times the procedure of running GPPE five times and selecting the best individual according to its training fitness. In both the classification and regression domains, the resulting bootstrapping distributions follow a normal distribution, according to a Kolmogorov–Smirnov test, with $p < 0.01$. Table 1.2 displays the average, standard deviation, and median values for the bootstrapping distribution on the test set.

To determine whether GPPE projections generate descriptive attributes and improve results over those of the original data set, we have used several well-known machine learning algorithms (MLAs) for both domains before and after projecting the data. We have chosen the best-performing GPPE individuals according to their training fitness (individuals e95 from the classification domain and e22 from the regression domain). e95 obtained a training fitness of 81.33% and a test fitness of 80.00%; e22 obtained a training NMSE of 0.59868 and a

TABLE 1.2 Average and Median Values of the Bootstrapping Distribution for B = 100 and R = 5

	Average	Median
Classification domain	79.05 ± 1.01	78.88
Regression domain	0.864084 ± 0.02	0.861467

TABLE 1.3 Comparison of MLA Performance Using Original and Projected Data

	Classification Domain			
	MLP	SMO	Simple Logistics	RBF Network
Original	78.50	79.25	61.75	72.75
Proj. by e95	**80.75**	**80.25**	**80.25**	72.75
PCA var. = 0.95	76.25	**80.25**	79.75	**73.25**
	Regression Domain			
	Linear Reg.	Simp. Lin. Reg.	Least Med. Sq.	Additive Reg.
Original	0.878816	0.932780	1.056546	0.884140
Proj. by e22	**0.838715**	**0.837349**	**1.012531**	**0.851886**
PCA var. = 0.95	0.904745	0.932780	1.061460	0.899076

test NMSE of 0.844891. Multilayer perceptron [17], support vector machine (SMO) [18], simple logistics [19], and a radial basis function network [20] were applied to the projected and unprojected data in the classification domain. The same process was carried out for the regression data, with linear regression, simple linear regression [19], least median square [21], and additive regression [19]. Table 1.3 displays these results, which were computed using the training and test sets described in Section 1.3 (they are the same sets on which GPPE was run). To compare with a commonly used projection algorithm, Table 1.3 also shows the accuracy of the same algorithms with data preprocessed by principal component analysis (PCA) with a variance of 0.95.

Table 1.3 highlights those values where projecting the data yields better results than for the same algorithm working on unprojected data. In both domains, classification and regression accuracy improve by projecting the data, even when PCA is applied. Unfortunately, the differences in Table 1.3 are not statistically significant (according to a t-test with 0.05 confidence). Even so, it must be noted that GPPE reduced the number of attributes of the problem from 40 to only 3 in the classification domain, and from 8 to 3 in the regression domain. PCA generated 26 and 6 attributes on the classification and regression domains, respectively.

1.6 CONCLUSIONS

Inductive ML algorithms are able to learn from a data set of past cases of a problem. They try to extract the patterns that identify similar instances and use this experience to forecast unseen cases. It is clear that the prediction rate is affected enormously by the quality of the attributes of the data set. To obtain higher-quality attributes, researchers and practitioners often use feature extraction

methods, which can filter, wrap, or transform the original attributes, facilitating the learning process and improving the prediction rate.

Some work has been dedicated to implementing a GP system that can produce ad hoc feature extractors for each problem, as explained in Section 1.2. In this chapter we present our own contribution: GPPE, a genetic programming–based method to project a data set into a new data space, with the aim of performing better in data mining tasks. More specifically, in this work we focus on reducing dimensionality while trying to maintain or increase prediction accuracy. GPPE has been tested in the classification and regression domains. Results show that:

- GPPE is able to reduce dimensionality dramatically (from 40 to 3 in one of the domains).
- Attributes created by GPPE enforce nearly linear behavior on the transformed space and therefore facilitate classification and regression in the new space.
- Different learning algorithms can also benefit from the new attributes generated by GPPE.

In the future we would like to test GPPE on more complex domains with a large number of low-level attributes. For instance, image recognition tasks where attributes are individual pixels would be very appropriate. Classifying biological data involving multiple time series, including, for example, heart rate, EEG, and galvanic skin response, may also be a candidate domain. Such specialized domains might require specialized primitives (e.g., averages, summations, derivatives, filters). GP has the advantage that any primitive whatsoever can be utilized by adding it into the function set. We believe that in these domains, our projection strategy will be able to improve accuracy in addition to reducing dimensionality.

We have also compared the results obtained by PCA with those obtained by GPPE, but these two techniques are in no way in competition. On the contrary, in the future we want to use PCA to filter irrelevant attributes, thus reducing the GPPE search space and possibly improving its results.

Automatically defined functions (ADFs) are considered as a good way to improve the performance and optimization capabilities of GP. They are independent subtrees that can be called from the main subtree of a GP individual, just like any other function. The main subtree plays the same part in a GP individual as in the main function of a C program. ADFs evolve separately and work as subroutines that main subtrees can call during execution of the individual. It could be very interesting in the future to study the impact of ADFs in GPPE performance. We think that this could improve the generalization power of our method.

Acknowledgments

The authors are partially supported by the Ministry of Science and Technology and FEDER under contract TIN2006-08818-C04-02 (the OPLINK project).

REFERENCES

1. H. Liu et al., eds. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic, Norwell, MA, 1998.
2. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.
3. J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.
4. J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human–Competitive Machine Intelligence*. Kluwer Academic, Norwell, MA, 2003.
5. F. E. B. Otero, M. M. S. Silva, A. A. Freitas, and J. C. Nievola. Genetic programming for attribute construction in data mining. In C. Ryan et al., eds., *Genetic Programming* (*Proceedings of EuroGP'03*), vol. 2610 of Lecture Notes in Computer Science. Springer-Verlag, New York, 2003, pp. 389–398.
6. K. Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.
7. L. S. Shafti and E. Pérez. Constructive induction and genetic algorithms for learning concepts with complex interaction. In H.-G. Beyer et al., eds., *Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO'05*), Washington, DC. ACM Press, New York, 2005, pp. 1811–1818.
8. I. Kuscu. A genetic constructive induction model. In P. J. Angeline et al., eds., *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 1. IEEE Press, Piscataway, NJ, 1999, pp. 212–217.
9. Y.-J. Hu. A genetic programming approach to constructive induction. In J. R. Koza et al., eds., *Genetic Programming 1998* (*Proceedings of the 3rd Annual Conference*), Madison, WI. Morgan Kaufmann, San Francisco, CA, 1998, pp. 146–151.
10. T. Howley and M. G. Madden. The genetic kernel support vector machine: description and evaluation. *Artificial Intelligence Review*, 24(3–4):379–395, 2005.
11. D. Eads, D. Hill, S. Davis, S. Perkins, J. Ma, R. Porter, and J. Theiler. Genetic algorithms and support vector machines for time series classification. In *Proceedings of SPIE 4787 Conference on Visualization and Data Analysis*, 2002, pp. 74–85.
12. N. R. Harvey, J. Theiler, S. P. Brumby, S. Perkins, J. J. Szymanski, J. J. Bloch, R. B. Porter, M. Galassi, and A. C. Young. Comparison of GENIE and conventional supervised classifiers for multispectral image feature extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 40(2):393–404, 2002.
13. J. J. Szymanski, S. P. Brumby, P. Pope, D. Eads, D. Esch-Mosher, M. Galassi, N. R. Harvey, H. D. W. McCulloch, S. J. Perkins, R. Porter, J. Theiler, A. C. Young, J. J. Bloch, and N. David. Feature extraction from multiple data sources using genetic programming. In S. S. Shen et al., eds., *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery VIII*, vol. 4725 of *SPIE*, 2002, pp. 338–345.
14. A. Vieira, B. Ribeiro, and J. C. Neves. A method to improve generalization of neural networks: application to the problem of bankruptcy prediction. In *Proceedings of the 7th International Conference on Adaptive and Natural Computing Algorithms*

(*ICANNGA'05*), Coimbra, Portugal, vol. 1. *Springer-Verlag Series on Adaptative and Natural Computing Algorithms*. Springer-Verlag, New York, 2005, pp. 417.

15. D. Quintana, C. Luque, and P. Isasi. Evolutionary rule-based system for IPO underpricing prediction. In H.-G. Beyer et al., eds,. *Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO'05*), New York, vol. 1. ACM SIGEVO (formerly ISGEC). ACM Press, New York, 2005, pp. 983–989.
16. B. Efron and R. J. Tsibirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1994.
17. C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.
18. C. J. Stone. Additive regression and other nonparametric models. *Annals of Statistics*, 13(2):689–705, 1985.
19. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, San Francisco, CA, 2005.
20. M. Buhmann and M. Albowitz. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, New York, 2003.
21. P. Rousseeuw. Least median squares of regression. *Journal of the American Statistical Association*, 49:871–880, 1984.