Appl Intell (2008) 29: 174–185
DOI 10.1007/s10489-007-0048-0

GPPE: a method to generate ad-hoc feature extractors for prediction in financial domains

César Estébanez · José M. Valls · Ricardo Aler

Published online: 6 June 2007

Abstract When dealing with classification and regression problems, there is a strong need for high-quality attributes. This is a capital issue not only in financial problems, but in many Data Mining domains. Constructive Induction methods help to overcome this problem by mapping the original representation into a new one, where prediction becomes easier. In this work we present GPPE: a GP-based method that projects data from an original data space into another one where data approaches linear behavior (linear separability or linear regression). Also, GPPE is able to reduce the dimensionality of the problem by recombining related attributes and discarding irrelevant ones. We have applied GPPE to two financial domains: Bankruptcy prediction and IPO Underpricing prediction. In both cases GPPE automatically generated a new data representation that obtained competitive prediction rates and drastically reduced the dimensionality of the problem.

Keywords Genetic programming · Projections · Attribute construction · Dimensionality reduction

1 Introduction

The problem of the representation of data is a key issue in the Machine Learning (ML) field. In inductive ML, examples or instances are usually represented as tuples, by means of attributes. Unfortunately, in many classification problems, the original attributes are not the most suitable for prediction and the representation can be improved. Very often attributes have complex interactions between them. In these situations, the relationship between an attribute and a class (or concept) depends in turn on the value of other attributes [1].

Regression problems are also a very interesting family of financial forecasting. In this case, regression methods try to approximate examples by means of surfaces and hyperplanes. Again, a bad representation of data can distort the spatial configuration of examples and its relationships with the target value, thus distracting regression methods and making impossible to do accurate forecasts. In summary, it is known that inappropriate representations of the data can limit the performance of ML.

Attribute Construction methods or Constructive Induction (CI) methods appear in this context. They are part of the wider framework of Feature Extraction, Construction and Selection Methods [2]. The objective of CIs is to generate new attributes from the original ones, transforming the representation of the dataset in such a way that the regularities of the problem become more apparent and easier to detect. This way, CIs improve the prediction capabilities of classical ML forecasting methods.

Principal Component Analysis (PCA) [3] is among the most widely used and accepted techniques in the preprocessing of financial forecasting problems. PCA is a statistical tool that constructs a set of attributes (called factors or principal components) by linearly combining the original attributes. The idea is to generate orthogonal factors that cover most of the total variance of the original attributes. This way, PCA is able to eliminate correlated and irrelevant information, reducing the dimensionality of the problem while retaining the maximum amount of information.

C. Estébanez (✉) · J.M. Valls · R. Aler
Universidad Carlos III de Madrid, Av. De la Universidad 30, 28911, Leganés, Madrid, Spain
e-mail: cesteban@inf.uc3m.es

J.M. Valls
e-mail: jvalls@inf.uc3m.es

R. Aler
e-mail: aler@inf.uc3m.es

In this work we present a method for attribute generation based in Genetic Programming (GP). We call this method GPPE (Genetic Programming Projection Engine). The main idea is to use GP to generate ad-hoc projections that transform the original attributes into new ones of higher quality, improving the predictions. The objective is to find a projection that can generate a new data space in which instances of the problem can be predicted or approximated by means of a linear method. Furthermore, when projecting it is possible to group attributes showing a high interrelationship and to eliminate irrelevant ones. This way we can improve the prediction rate and also reduce the dimensionality of the problem.

Financial prediction problems usually have a group of characteristics that make them especially suitable for a method like GPPE: They are complex and generally very hard problems, and they often have a large number of attributes. To demonstrate the usefulness of GPPE, in this work we have applied it to two important financial domains: Bankruptcy prediction and IPO Underpricing prediction. In addition to their interest, they have been chosen because from the ML point of view, the techniques to be used are very different. Bankruptcy prediction is a classification problem whereas IPO prediction is a regression problem. The aim is to show the generality of the GPPE approach.

The goal of Bankruptcy prediction is to forecast when a company faces bankruptcy. This is a difficult and interesting problem that requires a good knowledge of the company [4] and has traditionally been faced by experts applying heuristic rules, although some ML approaches have been used too.

An IPO is the first public stock offering of a company [5, 6]. The existence of important variations on the price of the IPOs in the first day of trading has been documented for many years. Thus, it is vital that the company and its investment bankers price the IPO as closely as possible to the final price.

These problems are only an example of how GPPE can help with predictions in financial domain, and they are described in more detail in Sect. 5.

The structure of the rest of the paper is as follows. Section 2 gives an overview of Genetic Programming. Section 3 describes GPPE. Section 4 summarizes the Principal Component Analysis, a commonly used technique for selecting attributes in financial domains. Section 5 introduces two important financial domains: the bankruptcy prediction and the IPO underpricing prediction. Then, Sect. 6 applies the method to these financial domains. Next, Sect. 7 reports on the related work. And finally, Sect. 8 draws some conclusions and describe possible future research directions.

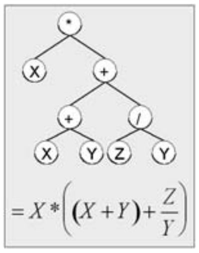

Fig. 1 Example of individual and the associated arithmetical expression

2 Genetic programming

In this section, we will introduce the field of Genetic Programming (GP) [7], and motivate the need for Automatic Defined Functions [8].

Genetic Programming is an evolutionary technique designed to generate programs automatically. It has three main elements:

- A population of individuals. In this case, the individuals are computer programs. They are usually represented as parse trees, made of functions (with arguments), and terminals (with no arguments: constants), as shown in Fig. 1. The initial population is made of individuals generated randomly.
- A fitness function. It is used to measure the goodness of the computer program represented by the individual. Usually, this is done by running the individual many times, using many (input, output) cases, as well known as fitness cases. The fitness of the individual is then computed by taking into account how many times the output of the program guesses the expected output.
- A set of genetic operators. In GP, the basic operators are reproduction, mutation, and crossover. Reproduction does not change the individual. Mutation changes a function, a terminal, or a subtree. And crossover exchanges two subtrees from two parent individuals, thereby combining characteristics from both of them into the offspring.

The GP algorithm enters into a cycle of fitness evaluation and genetic operator application, producing consecutive generations of populations of computer programs, until a good enough individual is found. In terms of classical search, GP is a kind of beam search, the heuristic being the fitness function. GP has many parameters, the most important ones being the size of the population (M) and the maximum number of generations (G). Also, every genetic operator has a probability of being applied. The crossover operator is usually the most likely to be used.

3 Description of GPPE

GPPE (Genetic Programming Projection Engine) is a Genetic Programming based system for computing data projections with the aim of improving prediction accuracy.

The training data E belongs 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. In the current paper, by "easier" we mean that data in the new space V is 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, 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 of dimensions of the projected space V). Fitness of individuals is computed by measuring the degree of linear behavior of data in the projected space, by using the individual as a projection function from U to V. The system stops if a 100% linear behavior has been achieved (i.e. 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).

For the implementation of our application, we have used Lilgp 1.1, the software package for Genetic Programming developed in the Michigan State University by Douglas Zongker and Bill Punch [9].

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

3.1 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 3 to 2 dimensions, made of the following function and terminals:

Functions = $\{+, *, {}^2\}$

Terminals = $\{3, u_0, u_1, u_2, 2\}$

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 unnecessarily the search space. In practice, different domains will require different function and terminal sets, which have to be chosen by the programmer. We rather 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 try to perform an analysis of 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 so-called Ephemeral Random Constants (ERC). The 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 has shown good results.

3.2 GP individuals

Projections can be expressed as:

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

In order 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]

3.3 The 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 three different fitness functions: two for classification problems and one for regression problems.

3.3.1 Classification

The first fitness function uses a Simple Perceptron (SP) to compute the degree of linear separation between the classes. This SP is run on the projected data for many learning cycles (we have found empirically that 500 iterations is enough). If data can be separated linearly, the SP will converge, and the individual will get the highest fitness (and it is a perfect solution to the problem). Otherwise, the SP will not converge. In that case, the best classification rate obtained by the SP during the learning process will be assigned as the fitness of the individual. This value provides a lower bound on the degree of linear separability.

The idea behind this function is the following: SP is a linear classifier. Thus, optimizing its classification rate means that selection pressure will encourage projections that tends to spatially separate the classes. On the other hand, SP is one of the most simple classifiers, and we have preferred to use simple classification schemes in order to avoid overfitting: if both GP projections and the classification scheme have a lot of degrees of freedom, overfitting should be expected.

[1]Actually, we use the lil-gp mechanism for ADF (Automatically Defined Functions) for representing subtrees. That is, an individual is made of P ADF's and no main program. It is the responsibility of the fitness function to call each subtree sequentially.

The other 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 instances belonging to the same class to get close. The great advantage of this function is that it is fast (up to forty times faster than SP) and simple. We call this function CENTROIDS.

Fitness is actually computed using cross-validation on the training data. That is, in every cross-validation cycle, the SP or the Centroids-based classifier is trained on some of the training data and tested on the rest of the training data.

3.3.2 Regression

In this case, linear behavior is defined as data fitting an 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).

4 Principal component analysis (PCA)

In this work we want to compare GPPE with PCA. PCA is a statistical technique for reducing the dimensionality of datasets. It is based on eliminating information that is correlated. This is done by creating a new coordinate system in which the new axis (attributes) are linear combinations of the original variables. PCA creates a set of factors such that the greatest variance by any projection of the data comes to lie on the first factor, the second greatest variance on the second factor, and so on. PCA keeps creating factors until the cumulative percentage of the total variance covered by the factors reaches a threshold. This threshold has to be defined by the user (the most common value is $v = 0.95$, i.e. 95% of the total variance covered by the factors).

GPPE has some similarities with PCA: Both try to create a new attribute set by combining the original attributes, and both attempt to reduce the dimensionality of the problem. However, there are very important differences between them: First, factors generated by PCA are linear combinations of the original attributes, while in GPPE non-linear combinations are allowed, and even very desirable. This happens even with the most basic function set which includes multiplication between variables and division. Second, there is a very important difference in the philosophy and the objectives of GPPE and PCA: While PCA tries to optimize the amount of information that can be expressed with the minimum number of variables, GPPE tries to optimize the prediction capabilities of classic algorithms on the projected dataset. Assuming these important differences, we should expect very different results from these two methods. Anyway, PCA is probably among the most widely accepted methods that are used for the preprocessing of datasets in financial prediction problems. Because of that, we are very interested in making some experimental comparison between GPPE and PCA. We will do so in Sect. 6.

5 Description of the financial domains

The bankruptcy prediction and the IPO underpricing prediction are two important financial problems that we tackle in this work. In this section we will describe these domains.

5.1 Bankruptcy prediction

Predicting when a company is facing bankruptcy is a difficult and interesting problem that requires a good knowledge of the company [4]. This problem has traditionally been faced by experts applying heuristic rules. More recently, automatic approaches have been used; Some effort has been done in applying Artificial Neural Networks (ANN) to this kind of prediction problems [10–12]. These approaches take advantage of the capacity of ANNs to find non-linear relationships between variables of the problem. In [13] a training algorithm for classifying high dimensional data using Multilayer Perceptrons is applied to this problem. In [11], they use a model based on Genetic Algorithms for extracting rules that are easy to understand by users.

GPPE also uses an Evolutive Computing approach. By means of a Genetic Programming engine, it generates projections that transform data into a new coordinate system. In this new space, data can be more easily treated, simplifying the classification task. This transformation of the data usually will imply a variation in the number of dimensions of the problem. Thus, GPPE tries to improve the prediction rates and reduce the dimensionality of the problem at the same time.

5.2 IPO underpricing prediction

An IPO is the first public stock offering of a company. It has been documented for many years the existence of important variations on the price of the IPOs in the first day of trading. These variations usually come as price gains, which means that the price of an IPO at the end of the first trading day is usually greater than the initial price [5]. The problem for the company is that it is very difficult to accurately price an IPO: the company has little or no information about demand, acceptance, competitive response and many other factors that should influence in the IPO pricing. Once the issue price is selected, the company is committed to maintain this price for the entire offering. If the final price is significantly over or under the IPO issue price, company will suffer important losses in the form of underpricing or overpricing (for more information about underpricing/overpricing losses, reader can refer to [6]). Thus, it is vital that the company and its investment bankers price the IPO as closely as possible to the final price.

This has been the motivation of a vast amount of Academic Work contributed in the last 30 years. Most of the proposed approaches consists in statistical methods such as multiple linear regression. Jain and Nag apply in [14] an Artificial Neural Network model to IPO underpricing prediction.

Here, we use an Evolutionary Computation approach to perform transformations of the data set. These transformations make easier to adjust data to a classic regression model and they also reduce the number of attributes of the problem. Thus, the prediction task is improved and the dimensionality of the problem is reduced.

6 Application to financial domains

In this work we have tested GPPE against two different problems: first, we tackled the problem of bankruptcy prediction, second, we worked in IPO Underpricing prediction domain. Also, in order to verify the correct operation of GPPE and to make it more comprehensible, we have decided, as a previous step, to apply it to a toy domain. In this domain, the direct solution is known and the solution given by GPPE can be easily verified.

All the experiments share some common parameters, which we list in Table 1.

Table 1 Common parameters for the experimentation carried out in this work

Parameter	Value
Initialization method	Half and half
Init depth	2–6
Number of Genetic Ops.	3
Operator 1	Reproduction
	select = fitness proportionate
	rate = 0.15
Operator 2	Crossover
	select = tournament (size 4)
	rate = 0.80
Operator 3	Mutation
	select = tournament (size = 2)
	rate = 0.05

6.1 Synthetic domain

This domain is composed of two datasets: Ellipse and EllipseRT. Both are two-class classification problems with 1000 two-dimensional points. In dataset Ellipse, the examples belonging to class 0 are situated inside an ellipse centered in the origin, whose focuses are placed at points $(-10, 0)$ and $(10, 0)$. Class 1 instances are situated outside the ellipse. Dataset ElipseRT is similar, but the ellipse has been rotated and translated, so its focuses are now located at points $(10, -10)$ and $(1, 7)$.

We ran our application on the data set Ellipse with the following parameters: Maximum number of generations (G) = 500; population size (M) = 5000; function set = $\{+, -, *, /, \text{SQR}, \text{SQRT}\}$. The number of dimensions selected for the projected space V is 2 in this case, due to considering them sufficient for a so simple problem.

The graphical representation (not shown here) shows an almost perfect linear separability of projected data. A Simple Perceptron on the projected data obtains 100% accuracy.

The same process was followed with dataset ElipseRT. Parameters for the execution stay the same, but this time the dimension of the projected space is 3. A Simple Perceptron applied to it obtains 99,9% accuracy, which means that data has also been separated almost linearly. Figure 2 displays the projected data. Points belonging to the inside of the ellipse appear as black solid squares in the bottom of the valley-like figure, whereas points belonging to the outside appear as empty squares, in the rest (upwards) of the figure.

6.2 Bankruptcy prediction experiments

In this paper, we have applied GPPE to a bankruptcy prediction problem. We use a dataset provided and described by Vieira et al. [15]. This data set studies the influence of several financial and economical variables on the financial health of a company.

We have applied GPPE to a dataset formed by 1158 companies, half of which are in a bankruptcy situation (class 0) and the rest have a good financial health (class 1). Companies are characterized by 40 numerical attributes as described in [15]. For validation purposes, we have divided the data set into a training and a test set, containing 766 (64%) and 400 (36%) instances respectively.

With this dataset, we carried out two groups of experiments, using a different fitness function in each group and different configurations too.

6.2.1 CENTROIDS fitness function

In the first group of experiments, we ran 100 experiments and we used the fitness function CENTROIDS. Ten-fold

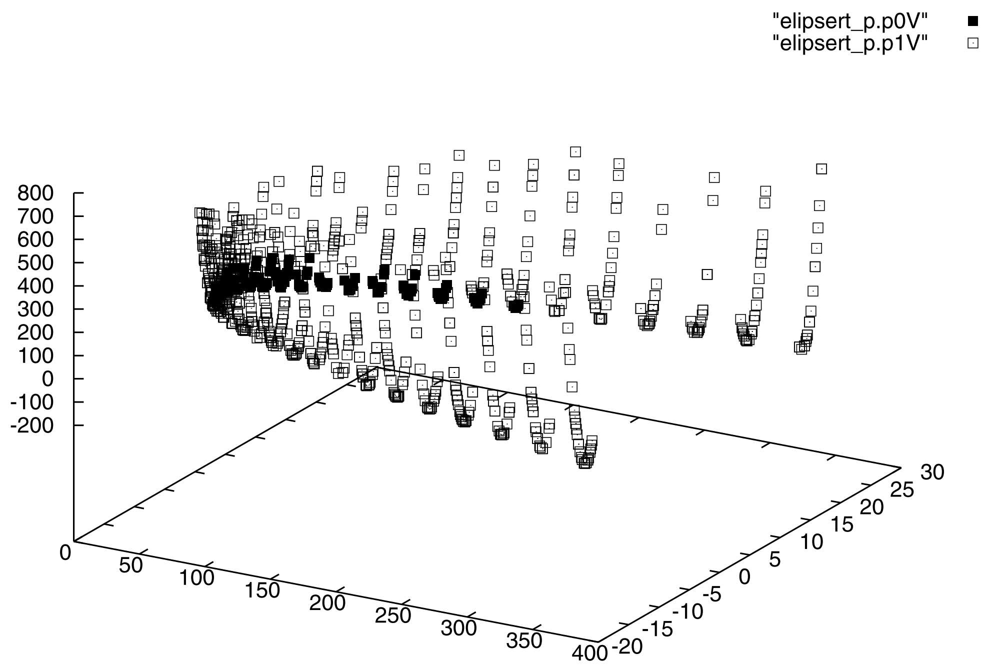

Fig. 2 Projected data for the rotated and translated ellipse. Two classes: black solid squares and empty squares

cross validation has been used in the fitness function. At the end of every run, GPPE selected the best individual obtained. This individual was then evaluated again with the test set. We have found empirically that good results were obtained using a population of 500 individuals, with a node limit of 75 nodes and 100 generations.

The individual selected from the best of the 100 GP-runs (labeled as e95) obtained a training fitness of 81.33% and a test fitness of 80.00%. The selection of the best individual was based exclusively on its training fitness. The mathematical expression corresponding to this individual is shown next:

```
v0 =
 (/ (- (- (- (/ (- u19 u33)
                (- (/ (- u30
              (- u19 u33)) u3) u8))
            (/ u18
                (- (/ (- u30
              (- u8 u33)) u3) u8)))
                         u8) u33) u39)


v1 =
(- (+ u33
      (- (- u33
            (- 8.80 u16))
         (+ u5
            (- (- (- (- (- u23 u35) u35)
            u35) u35) u35))))
   (- u23 u35))

v2 =
 (- u32 u11)
```

v0, v1, and v2 are the coordinates in the projected space (i.e. the newly generated attributes). u0 to u39 are the original attributes of the problem.

Figure 3 summarizes the results obtained by all the 100 individuals selected from the 100 GP-runs. The x-axis represents the classification rate and the y-axis represents the frequency of GP-runs that achieved a classification rate equal or better than a particular x-value. For instance, point $(79.75, 0.20)$ means that 20% of GP-runs will achieve an accuracy equal to or better than 79.75%.

In order to check these results with independent algorithms, we used some well-known Machine Learning Algorithms (MLA) in the Weka tool. We compared the performance of the learning algorithm on the original dataset and then on the projected data. Individual e95 was used for this purpose. Table 3 displays the results using a Multilayer Perceptron, a Support Vector Machine (SMO), Simple Logistics, a Radial Basis Function Network, and K*. Default parameters were used. We used the 766 instances of the training set for training the MLAs, and the 400 instances of the test set for testing. In the second row, we highlight the values which are better than those obtained by the same algorithm but with the original data. Individual e95 improves the classification rate of all the tested algorithms. It has to be remarked that GPPE reduced the number of attributes of the problem from 40 to only 3. Also, we applied PCA on the dataset: PCA generated 26 factors for describing 95% of the variance of the original data. The third row of the table shows results of the MLAs when using the factors generated by PCA. We highlight those values which are equal to or better than those obtained by GPPE.

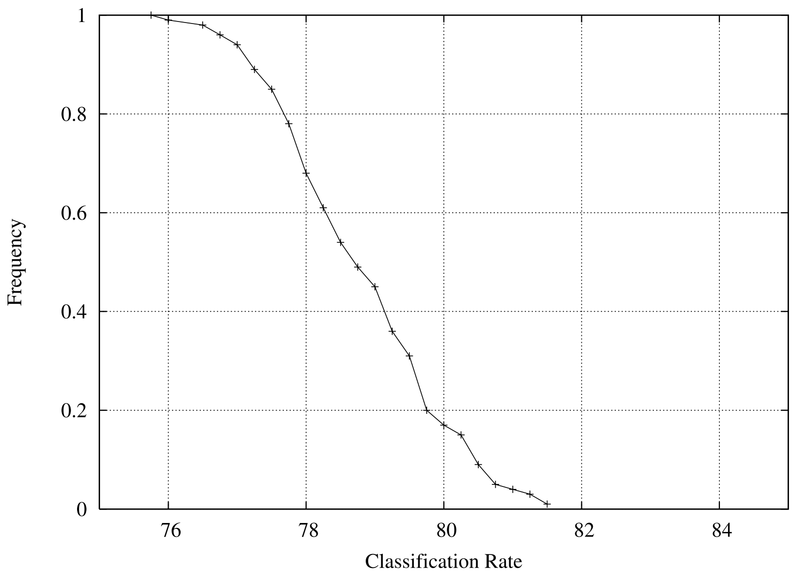

Fig. 3 Results obtained by all the 100 individuals selected from the 100 GP-runs

Table 2 Comparison of MLAs performance when using original data and projected data

	MLP	SMO	Simple logistic	RBF network network	K*
Original	78.50	79.25	79.25	61.75	72.75
Proj. by e95	**80.75**	**80.25**	**80.25**	**72.75**	**75.75**
PCA $v = 0.95$	76.25	**80.25**	79.75	**73.25**	67.75

Results in Table 3 show that in all cases projecting the data with GPPE improves the classification rate. Also, in all cases but two, GPPE obtained better results than PCA. Unfortunately, except for the K* algorithm, the differences between the classification rates of projected and original data are not statistically significant. But on the other hand, GPPE also managed to reduce the number of dimensions from 40 to only 3 while retaining competitive results.

The best classification rate achieved in our experiments with the original data is 79.25%. Looking on Fig. 3 one can see that one single run of GPPE will find a projection which will perform (in our tests) equal or better than the original data with a probability of 0.36 (i.e. 36% of the runs will find projections that outperform the original data).

Figure 4 shows the test dataset after being projected by individual e45. Crosses represent companies in bankruptcy situation and circles are companies with a good financial health. Crosses and circles are distributed in two separated groups, with overlapping in the center. This fact shows the high descriptive power of the three brand-new attributes.

6.2.2 Simple perceptron (SP) fitness function

In the second group of experiments we used the SP fitness function. 100 runs were also carried out keeping the same parameters. Results are slightly worse than those obtained with CENTROIDS. Furthermore, when using this fitness function GPPE is much slower (about 40 times slower). Therefore we will not provide a more detailed analysis here.

6.3 IPO underpricing prediction

The IPO sample is composed of 1000 companies going into the US Stock Market for the first time, between April 1996 and November 1999 [6]. Each company is characterized by seven explicative variables: underwriter prestige, price range width, price adjustment, offer price, retained stock, offer size and relation to tech sector.

For validation purposes, we have divided the data set into a training and a test set, containing 800 (80%) and 200 (20%) instances respectively. GPPE will project the data from its original seven-dimensional space to a new three-dimensional one. We carried out 100 experiments with the following GP parameters: G = 100; M = 5000; max nodes = 75. We used the fitness function REGRESSION. Ten-fold cross validation has been used in the fitness function. The quality of the regression is measured by means of the Normalized Mean Square Error (NMSE), as we already explained in Sect. 3.3. Figure 5 summarizes the results obtained by all the 100 individuals selected from the 100 GP-runs.

The individual selected from the best of the 100 GP-runs (labeled as e22) obtained a training NMSE of 0.59868 and a test NMSE of 0.844891. The selection of the best individual was based exclusively on its training fitness. The

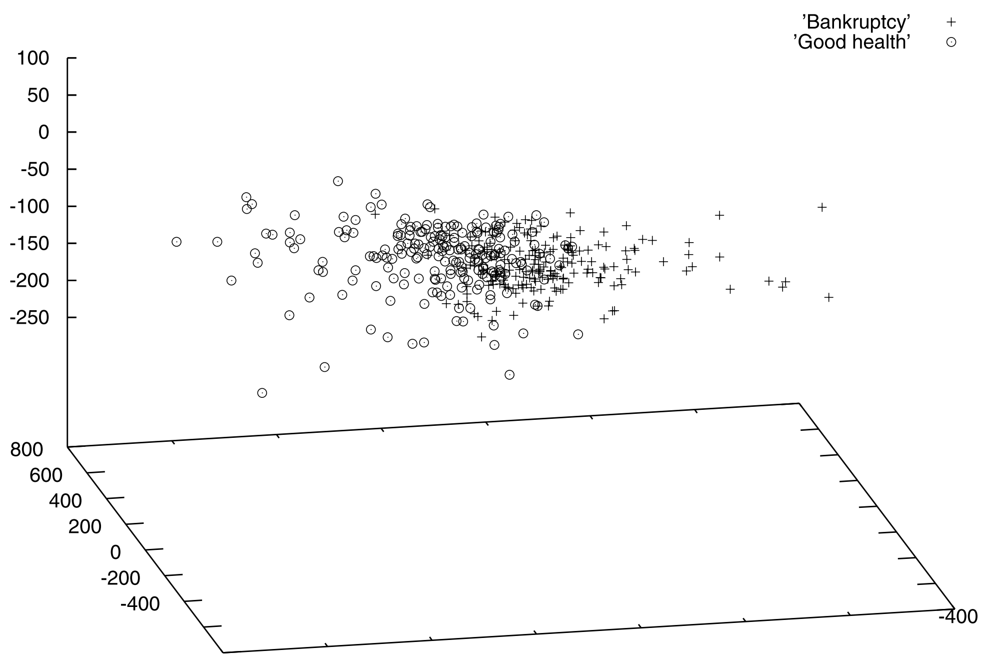

Fig. 4 Bankruptcy test dataset after being projected by best-of-run-e95

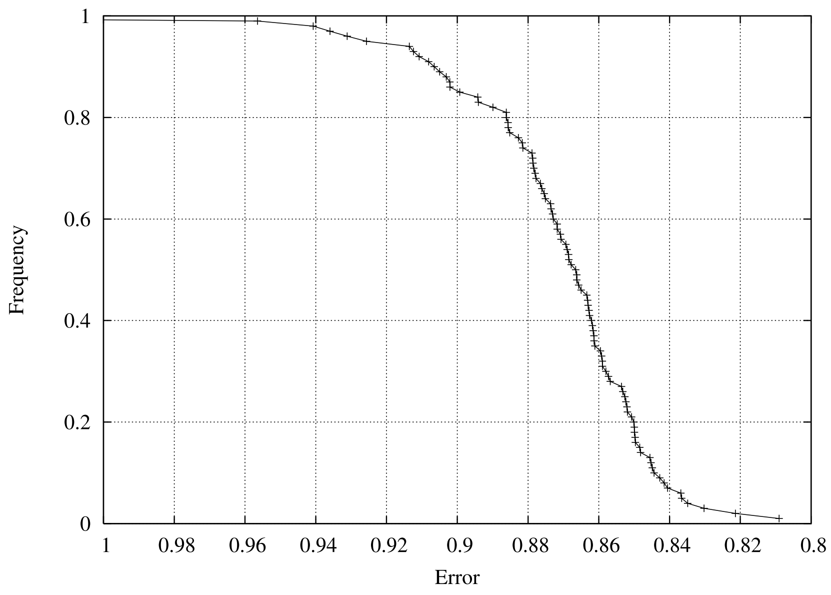

Fig. 5 IPO Underpicing test dataset after being projected by best-of-run-e22

mathematical expression corresponding to this individual is shown next:

```
v0 = (* (- u5
        (- (- (- (* u3 u4) u2) u2) u2))
         u5)

v1 = (/ (* (+ (+ (/ 9.50 (/ u6 u2))
        (/ 5.10 u1)) (* (+ (+ (/ u6 u2)
        (/ 0.16 u2)) u6) (+ 1.26 u5)))
      (- (+ (- u6 u3) 1.26) (/ 0.16 u2)))
   (/ (* (* (/ 2.89 u3) (+ (+ u2 u2) u0))
            (+ (+ (+ u2 u2) u3) u6))
          (+ (- (/ u6 0.57) u1)
              (+ 1.26 u5))))

v2 = u6
```

Again, v0, v1, and v2 stands for the newly generated attributes. Also, u0 to u6 stands for the original attributes of the problem:

u0	= underwriter prestige	(*PRES*)
u1	= price range width	(PRZ_{RG})

u2	= price adjustment	(PRZ_{ADJ})
u3	= offer price	(OF_{PRZ})
u4	= retained stock	(RET_{ST})
u5	= offer size	(OF_{SZ})
u6	= relation to tech sector	$(TECH)$

Looking at the individual e22, one can find some interesting features. Attribute v1 is obscure, we cannot understand its nature or the interactions between original attributes that it encodes. Very often it is not easy to interpret the solutions produced by GP, so the unclearness of attribute v1 is not surprising. However, attribute v2 could not be more clear. It simply represent the original variable TECH. TECH is a dummy variable which is used to discriminate between companies related to the technological sector and companies not related to it. Companies related to the tech sector have some characteristics, such as the size or the risk profile, that are considered as important for linear models. TECH has been proved to be a crucial factor for the estimation of the initial price of a stock offer [16]. So it makes a lot of sense that GPPE selects the TECH variable to be one of the final attributes. As for the v0 attribute, next we show a simplified version in which we replaced the u0, u1, ..., u6 with the name of the real variables of the model:

$$v0 = [OF_{SZ} - ((OF_{PRZ} * RET_{ST}) \\ - 3 * PRZ_{ADJ})] * OF0_{SZ}.$$

Attribute v0 encodes the relationship between the offer size, the offer price, the price adjustment and the amount of retained stock. This is also interesting because using PCA on the original data one can find that the first principal component only has high coefficients (between 0.5 and 0.6) in variables u0, u3 and u5 (*PRES*, OFF_{PRIZ} and OFF_{SZ}); and second principal component has high coefficients (around 0.68) only in variables u2 and u4 ($PRIZ_{ADJ}$ and RET_{STO}). Only these two principal components cover 55.4% of the total variance. And all the variables that appear in the two most important factors (but *PRESS*) also appear in v0. Again, this gives us reasons to think that e22 could make perfect sense.

We have used the Weka tool to compare the performance of some classic regression techniques when using the original and the projected data. We have selected the best individual labeled e22, and projected the original data. Then we used four different regression algorithms (LinearReg, Simple Linear Regression, LeastMedSq and Additive Regression) on both the original and the projected datasets. We also used PCA to generate new factors. PCA needed to generate 6 factors to cover 95% of the total variance. This seems to indicate that the original variables were reasonably correlation-free. Results measured in terms of NMSE are shown in Table 3.

In the second row we highlighted those results that were better than the equivalent results with original data. One can see that projection of individual e22 reduced the NMSE of all the regression techniques and at the same time, it reduced the number of variables of the problem from 7 to only 3. PCA needed 6 factors and PCA-projected data always performed worse than (or equal to) the original data. However the statistical test showed that the differences in Table 3 are not significant. Nevertheless, GPPE achieved competitive results and the dimensionality of the problem was reduced from 7 to 3.

The best NMSE achieved by the original data in our experiments was 0.878816. In Fig. 5, the closest record to 0.878816 that we registered is 0.878691, and this measure corresponds to a frequency of 0.72, which means that 72% of the times one run GPPE on the IPO Underpricing dataset, one obtain a projected dataset that works better (in our tests) for the classification than the original data.

7 Related work

There are many different Constructive Induction algorithms that use a vast number of different approaches. In [2], authors give a good starting point in the exploration of research into feature extraction, construction and selection. This book compiles contributions from researchers in this field and offer a very interesting general view.

Here we only discuss 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 any resemblance to GPPE.

In [17], the authors use typed GP for building feature extractors. Functions are arithmetic and relational operators. Terminals are the original (continuous) attributes of the original data set. Every individual is an attribute and the

Table 3 Comparison of MLAs performance (NMSE) when using original data and projected data

	LinearReg	Simp. Lin. Reg.	LeastMedSq	Aditive Reg
Original	0.878816	0.932780	1.056546	0.884140
Proj. by e22	**0.838715**	**0.837349**	**1.012531**	**0.851886**
PCA v = 0.95	0.904745	0.932780	1.061460	0.899076

fitness function uses the info gain ratio. Testing results, using C4.5, show some improvements in some UCI domains. Our approach differs in that our individuals contain as many subtrees as new attributes to be constructed and that the fitness function measures the degree of linear separation in the training data.

[18] follows a similar approach to ours, where every individual contains several subtrees, one per feature. C4.5 is used to classify in feature-space. Their work allows to cross over subtrees from different features, whereas we use homologous crossover so that only subtrees from the same features from two individuals can be crossed over. We believe that it would be desirable for constructed features to be independent and, even, orthogonal. Therefore, they should evolve independently and not allow to share code between features via crossover, as we do. This assumption might not work in all domains, but in any case, differences in empirical results should be expected with our approach.

In [1] authors discuss the importance of applying GA as a global search strategy for CI methods and the advantages of using these strategies instead of classic greedy methods. Also, they present MFE2/GA: a CI method that uses GA to search through the space of different combination of attributes subsets and functions defined over them. MFE2/GA uses a non-algebraic form of representation to extract complex interactions between the original attributes of the problem. There are obviously great differences between this work and our approach, but it is still a very interesting application of evolutionary approaches to the generation of CI methods.

In [19] authors present the GCI system. GCI is a CI method based in GP. It is similar to GPPE in the facts that it uses basic arithmetic operators and that the fitness is computed measuring the performance of a MLA (a quick-prop net) using the generated attributes. However, each individual represents a new attribute instead of a new attribute set. 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 only mentioned as a possible and very interesting future work.

In [20] Hu introduces another CI method based in Genetic Programming: GPCI. As in GCI, in GPCI each individual represents a new generated attribute. The fitness of a individual is evaluated by combining two different functions: an absolute measure and a relative measure. The absolute measure evaluates the quality of a new attribute using gain ratio. The relative measure evaluates the improvement of the attribute over its parents. Function set is formed by two Boolean operators: AND and NOT. GPCI is proved in twelve UCI domains and against two other CI methods, achieving some competitive results. While the basic scheme of GPCI is similar to GPPE, there are important differences including the function set, the representation of the attributes and the fitness function.

In [21], GP is used 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 the GP-evolved kernel. The hyperplane margin is used as tiebreaker to avoid overfitting. No attempt is made so that kernel satisfies standard properties (like Mercer's) but results in testing datasets are very good, compared to standard kernels. Instead of evolving distance or kernel functions, we evolve projections to spaces with larger, equal, or smaller number of dimensions. We believe that evolving actual distance functions or kernels is difficult, because some properties (like transitivity or Mercer's) are not easy to impose in the fitness computation.

In [22], Genetic Programming was used 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 (like convolution), statistical, and arithmetic operations. SVM was then used for classification in feature-space. Cross-validation on training data was used as 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, [23, 24] assembles image-processing primitives (edge-detectors, ...) to extract multiple features from the same scene to classify terrains containing objects of interest (golf courses, forests, etc.). Linear fixed-length representations for the GP trees are used. A Fisher Linear Discriminant is used for fitness computation. Results are quite encouraging but they restrict themselves to image-processing domains.

Results from the bibliography show that, in general, the GP-projection approach has merit and obtains reasonable results, but more research in the subject is needed. New variations of the idea and more domains should be tested.

Finally, an important contribution of our work is the fact that we do not restrict GPPE to classification problems. Instead, we have used our CI to improve prediction rates in regression problems.

8 Conclusions

When tackling prediction problems, there is a strong need for high-quality attributes: Machine Learning algorithms are designed to extract patterns and similarities from the attributes of the problem, but sometimes these attributes are not good enough to express the inner relationships between the instances of the same class. In these cases, these algorithms may perform badly. This is a capital problem not only in financial problems, but in every Data Mining domain.

Constructive Induction methods help to overcome this problem by mapping the original representation into a new better-quality one, where the regularities of the problem become more apparent. This way, Machine Learning performance is improved.

In this work we have presented GPPE: a GP-based method that projects data form an original data space into another one where prediction becomes easier. In the current paper, our goal is to be able to approximate linear behavior in the target space, so that classification or regression can be more easily achieved by general purpose Machine Learning Algorithms. In addition, GPPE has been able to transform data from a high dimensional space into a target space with fewer dimensions, while maintaining competitive results. Therefore it is specially suitable for problems with a high number of attributes.

We have applied GPPE to two financial domains: Bankruptcy prediction and IPO Underpricing Prediction. The first one is a classification problem which consists on predicting whether a company is facing bankruptcy. The second one is a regression problem since the goal is to predict the variations on the price of the IPOs in the first day of trading. Both of them are very relevant problems with important economical repercussions. In both cases, GPPE was able to greatly reduce the dimensionality of the problem: from forty to three in the first problem, and from seven to three in the second one. Furthermore, the attributes generated by GPPE were shown to be at least as suitable for prediction as the original ones. Also, it is important to remark that we tackled a regression problem and a classification problem. GPPE is a general method that could potentially be applied to a big number of financial forecasting problems.

PCA and GPPE have some similarities. Both try to create a new attribute set by combining the original attributes, and both attempt to reduce the dimensionality of the problem. However, there are important differences between them: the attributes generated by PCA are linear combinations of the original ones, whereas GPPE allows non-linear combinations. Besides, PCA tries to optimize the amount of information that can be expressed with the minimum number of variables, whereas GPPE tries to optimize the prediction capabilities of classic algorithms on the projected dataset. We believe that GPPE is not in competition with PCA, but quite the opposite: GPPE works much better when the original dataset has a lower dimensionality. This reduces the search space and eliminates possible redundancy in the data. In fact, it will be interesting to try to combine both methods: first using PCA to filter redundant information, and then using GPPE to optimize the prediction rate of MLAs. Studying the possible combinations between GPPE and PCA could be a very interesting future work.

Acknowledgements This article has been financed by the Spanish founded research MCyT project OP:LINK, Ref:TIN2005-08818-C04-02.

References

1. Shafti LS, Pérez E (2005) Constructive induction and genetic algorithms for learning concepts with complex interaction. In: Beyer H-G, O'Reilly U-M (eds) Genetic and evolutionary computation conference, GECCO 2005, proceedings, Washington, DC, USA, 25–29 June, 2005. ACM, New York, pp 1811–1818
2. Liu H, Motada H (eds) (1998) Feature extraction, construction and selection: a data mining perspective. Kluwer Academic, Norwell
3. Jolliffe IT (1986) Principal component analysis. Springer, Berlin
4. Altman EI (1984) Business failure prediction models: an international survey. J Bank Account Financ 8:171
5. Ritter JR, Welch I (2002) A review of ipo activity, pricing, and allocations. J Financ 57:1795–1828
6. Quintana D, Luque C, Isasi P (2005) Evolutionary rule-based system for ipo underpricing prediction. In: Proceedings of the genetic and evolutionary computation conference GECO 2005, vol 1
7. Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT, Cambridge
8. Koza JR (1994) Genetic programming II: automatic discovery of reusable programs. MIT, Cambridge
9. Zongker D, Punch B (1998) lil-gp 1.1 www homepage. http://garage.cps.msu.edu/software/lil-gp/
10. Tam KY, Kiang MY (1992) Managerial applications of neural networks: the case bank failure predictions. Manag Sci 38:926–947
11. Han I, Jo H, Shin KS (1997) The hybrid systems for credit rating. J Korean Oper Res Manag Sci Soc 22(3):163–173
12. Fletcher D, Goss E (1993) Forecasting with neural networks: An application using bankruptcy data. Inf Man 24(3):159–167
13. Vieira A, Bas N (2003) A training algorithm for classification of high-dimensional data. Neurocomputing 50:461–472
14. Jain BA, Nag BN (1995) Artificial neural network models for pricing initial public offerings. Decis Sci 26:283–299
15. Vieira A, Ribeiro B, Neves JC (2005) A method to improve generalization of neural networks: application to the problem of bankruptcy prediction. In: Proceedings of 7th international conference on adaptive and natural, computing algorithms. ICANNGA 2005. Springer series on adaptive and natural computing algorithms, vol 1. Springer, Berlin, p 417
16. Quintana D, Isasi P (2005) Estructura de colocación y rendimiento inicial de salidas a bolsa: tecnológicas frente a no tecnológicas. Actual Contab Faces 11:80–86
17. Otero FEB, Silva MMS, Freitas AA, Nievola JC (2003) Genetic programming for attribute construction in data mining. In: Ryan C, Soule T, Keijzer M, Tsang E, Poli R, Costa E (eds) Genetic programming, proceedings of EuroGP'2003, Essex, 14–16 April 2003. Lecture notes in computer sciences, vol 2610. Springer, Berlin, pp 389–398
18. Krawiec K (2002) Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genet Program Evol Mach 3(4):329–343
19. Kuscu I (1999) A genetic constructive induction model. In: 1999 congress on evolutionary computation. IEEE Service Center, Piscataway, pp 212–217
20. Hu Y-J (1998) A genetic programming approach to constructive induction. In: Koza JR, Banzhaf W, Chellapilla K, Deb K, Dorigo M, Fogel DB, Garzon MH, Goldberg DE, Iba H, Riolo R (eds) Genetic programming 1998: proceedings of the third annual conference, University of Wisconsin, Madison, Wisconsin, USA, 22–25 July 1998. Kaufmann, Los Altos, pp 146–151
21. Howley T, Madden MG (2005) The genetic kernel support vector machine: description and evaluation. Artif Intel Rev 24(3–4):379–395
22. Eads D, Hill D, Davis S, Perkins S, Ma J, Porter R, Theiler J (2002) Genetic algorithms and support vector machines for time series classification. In: Proceedings SPIE 4787 conference on visualization and data analysis, pp 74–85

23. Szymanski JJ, Brumby SP, Pope P, Eads D, Esch-Mosher D, Galassi M, Harvey NR, McCulloch HDW, Perkins SJ, Porter R, Theiler J, Young AC, Bloch JJ, David N (2002) Feature extraction from multiple data sources using genetic programming. In: Shen SS, Lewis PE (eds) Algorithms and technologies for multispectral, hyperspectral, and ultraspectral imagery VIII. SPIE, vol 4725, pp 338–345

24. Harvey NR, Theiler J, Brumby SP, Perkins S, Szymanski JJ, Bloch JJ, Porter RB, Galassi M, Young AC (2002) Comparison of GENIE and conventional supervised classifiers for multispectral image feature extraction. IEEE Trans Geosci Remote Sens 40(2):393–404