# Predicting Problem Difficulty for Genetic Programming Applied to Data Classification

Leonardo Trujillo, Yuliana Martinez, Edgar Galvan-Lopez, Pierrick Legrand

# Predicting Problem Difficulty for Genetic Programming Applied to Data Classification

Leonardo Trujillo,
Yuliana Martínez
Instituto Tecnológico de Tijuana, México
leonardo.trujillo.ttl@gmail.com
ysaraimr@gmail.com

Edgar Galván-López
School of Computer Science and Electronic Engineering
University of Essex, United Kingdom
edgar.galvan@gmail.com

Pierrick Legrand
Université Victor Segalen Bordeaux 2
ALEA Team, INRIA Bordeaux
IMB, UMR CNRS, France
pierrick.legrand@u-bordeaux2.fr

## ABSTRACT

During the development of applied systems, an important problem that must be addressed is that of choosing the correct tools for a given domain or scenario. This general task has been addressed by the genetic programming (GP) community by attempting to determine the intrinsic difficulty that a problem poses for a GP search. This paper presents an approach to predict the performance of GP applied to data classification, one of the most common problems in computer science. The novelty of the proposal is to extract statistical descriptors and complexity descriptors of the problem data, and from these estimate the expected performance of a GP classifier. We derive two types of predictive models: linear regression models and symbolic regression models evolved with GP. The experimental results show that both approaches provide good estimates of classifier performance, using synthetic and real-world problems for validation. In conclusion, this paper shows that it is possible to accurately predict the expected performance of a GP classifier using a set of descriptors that characterize the problem data.

## Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming—*program synthesis*

## General Terms

Theory, Experimentation, Performance

## Keywords

Genetic Programming, Performance prediction, Classification

## 1. INTRODUCTION

The goal of developing truly autonomous intelligent systems is still one of the main areas of research in computer science. One property that an intelligent system might require is the ability to determine, or estimate, the success rate it expects to achieve when confronted with a particular problem instance. By doing so, a system could then be able to make an informed choice as to which is the best problem solving strategy to follow, without the need for a human designer to choose the strategy beforehand. Therefore, it is necessary to describe the problem in a meaningful way, and from this derive an estimate of its inherent difficulty.

One of the most common problems in computer science is data classification. As such, researchers have developed a large number of classification algorithms, based on a variety of mathematical and computational models. However, in many cases, when confronted with a particular classification task it is not evident which is the best classifier to use. In order to make a reasoned decision, a researcher must assess the principal characteristics of the data and predict the expected performance that each classifier at his disposal might achieve. However, a general algorithmic solution to this task currently does not exist. That is why it is mostly solved using the trial and error method or by the researcher choosing the classification algorithm he is most familiar with. Therefore, we propose an approach to predict the performance of a particular classifier, a GP system based on a probabilistic classification procedure [30], which is referred to in this paper as the Probabilistic Genetic Programming Classifier (PGPC). The goal of the proposal is to build a predictive model of the expected classification accuracy of PGPC, using descriptive features of the problem data [12, 24, 4, 26]. The approach is validated on an extensive set of synthetic problems, and on real-world multi-dimensional data.

The remainder of this paper proceeds as follows. Section 2 overviews related work. Afterwards, the PGPC classifier is described in Section 3. Section 4 defines the problem statement and Section 5 describes the descriptors used to characterize each classification problem. Then, the predictive models are presented in Section 6. Section 7 contains the experimental setup and gives a summary of the results. Finally, Section 8 outlines some concluding comments.

## 2. RELATED WORK

While the paradigm of evolutionary computation has achieved a wide acceptance over the years, there are still many questions that need to be addressed to gain a proper understanding of the merits and shortcomings of this biologically inspired approach towards problem solving. One such question is that of determining the expected performance of an evolutionary search when applied to a particular problem instance. Several strategies have been proposed to study this fundamental issue. For instance, to predict the convergence of an evolutionary algorithm researchers have relied on schema theorems [5, 17, 16] and Markov chain models [14, 19]. While these approaches have produced solid theoretical foundations for some practical techniques [15, 18], their general use in real-world scenarios is still not a realistic option. Another way to determine if a problem can be solved by an evolutionary search, is to study the structure of the fitness landscape [7]. This idea of problem difficulty, or evolvability, has also been addressed by analyzing a phenomena in biological evolution known as neutrality [6, 29, 2, 28] and by studying the concept of locality within the search space [11, 3]. Finally, some of the most promising results in characterizing problem difficulty have been achieved with fitness distance correlation [25], fitness clouds and negative slope coefficient [20]. This last group of methods use sampling techniques to determine the structure of the search space, and the manner in which genetic operators can explore this structure guided by fitness. One drawback of these techniques is that they require a new sampling of the search space each time a new problem instance is encountered, a practical issue that might, or might not, be of importance.

The present work takes a different approach. Firstly, here we study the performance of GP applied to data classification, a domain where promising results have been achieved [30]. Secondly, instead of focusing on the structure of the search space as defined by fitness, the genetic operators and the encoding, we study the structure of the problem data itself; i.e., the data that needs to be classified [1]. From this, the goal is to infer the expected performance, and by corollary the difficulty of a problem, for a GP classifier.

In data classification, the behavior of an individual classifier depends on the characteristics of the data that is to be classified. One of the first attempts to detect a useful relationship between classifier performance and the underlying structure of the data is the STATLOG project [12], which developed a meta-level binary rule that determines the applicability of a classifier to a given problem. Another example is [24], where a statistical meta-model was developed to predict the expected performance of several common classifiers on a small set of real-world problems. The above examples perform a meta classification using statistical descriptors of the data. The predictive accuracy of these approaches, however, has been criticized because they do not consider the geometric relationships between the data, a more relevant characteristic to classifier accuracy [4]. On the other hand, [4] provides a detailed analysis of several types of complexity measures directly related to classification difficulty. However, [4] does not attempt to use these measures to estimate classifier performance. Based on the above works we propose a characterization of problem difficulty and prediction of GP performance. A similar goal is pursued in [13] for feature selection algorithms, based on a different experimental framework.

## 3. DATA CLASSIFICATION WITH GP

In a supervised classification problem some pattern $\mathbf{x} \in \mathbb{R}^P$ has to be classified in one of $M$ classes $\omega_1, ..., \omega_M$ using a training set $\mathcal{X}$ of N P-dimensional patterns with a known classification. Then, the goal of a supervised classifier is to build a mapping $g(\mathbf{x}) : \mathbb{R}^P \rightarrow M$, that assigns each pattern $\mathbf{x}$ to a corresponding class $\omega_i$, where $g$ is derived based on the evidence provided by $\mathcal{X}$. GP can be used in different ways to solve the problem described above, such as evolving rule sets [8] or decision trees [1]. However, this paper focuses on the method proposed in [30], PGPC.

[1]What is called the set of fitness cases in GP literature.

### 3.1 Probabilistic GP Classifier

PGPC is based on a simpler approach towards classification, also described in [30], which proceeds as follows. First, $\mathbb{R}$ is divided into a series of $M$ non-overlapping regions, one for each class. Then, GP is used to evolve a mapping $h(\mathbf{x}) : \mathbb{R}^P \rightarrow \mathbb{R}$, such that the region in $\mathbb{R}$ where pattern $\mathbf{x}$ is mapped to, determines the class to which it belongs. The most obvious shortcoming of such an approach is that it requires an a priori definition of the region boundaries.

To overcome the above problem, in PGPC it is assumed that the behavior of $h$ can be modeled using multiple Gaussian distributions, each corresponding to a particular class [30]. The distribution of each class $\mathcal{N}(\mu, \sigma)$ is derived from the examples provided for it in set $\mathcal{X}$, by computing the mean $\mu$ and standard deviation $\sigma$ of the outputs obtained from $h$ on these patterns. Then, from the $\mathcal{N}$ of each class, [30] proposes two fitness measures. The first is based on the overlap area among the distributions, and the second is based on Fisher's linear discriminant. Given that both methods exhibited similar performance, in this work we arbitrarily choose the latter.

#### *3.1.1 Fitness evaluation*

Given a two class problem, which is the type of problem studied in this paper, and the estimated Gaussian distribution $\mathcal{N}$ for each class, [30] proposes a distance measure between both classes as

$$d = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} , \tag{1}$$

where $\mu_1$ and $\mu_2$ are the means of the Gaussian distribution of each class, and $\sigma_1$ and $\sigma_2$ their standard deviations. When this measure tends to 0 it is the worst case scenario because the mapping of both classes overlap completely, and when it tends to $\infty$ it represents the optimal case with maximum separation. Moreover, in order to normalize the above measure, the final distance between both classes is given by

$$f_d = \frac{1}{1 + d} , \tag{2}$$

which provides the fitness measure for an individual mapping $h$ generated by GP.

#### *3.1.2 Classification with PGPC*

Once the Gaussian distribution $\mathcal{N}_i$ for each class is determined using the best individual $h$ found by GP, a new test pattern $\mathbf{x}$ is assigned to class $i$ when $\mathcal{N}_i$ gives the maximum probability. While [30] studies the effects of using multiple individuals to compute the final probability of class membership, here we only use the single best solution. Finally, Table 1 presents the setup for the PGPC system, that uses standard Koza style crossover and mutation and dynamic depth control to minimize bloat [22]. The PGPC system was implemented using Matlab 2009a and the GPLAB toolbox [21].

Table 1: **Parameters for the PGPC system used in the experimental tests.**

| **Parameter** | **Description** |
|---|---|
| *Population size* | 200 individuals. |
| *Generations* | 200 generations. |
| *Initialization* | *Ramped Half-and-Half*, with 6 levels of maximum depth. |
| *Operator probabilities* | Crossover $p_c = 0.8$; Mutation $p_\mu = 0.2$. |
| *Function set* | $\left\{+, -, *, /, \sqrt{}, sin, cos, log, x^y, \lvert\cdot\rvert, if\right\}$ |
| *Terminal set* | $\{x_1, ..., x_i, ..., x_P\}$ Where each $x_i$ is a dimension of the data patterns $\mathbf{x} \in \mathbb{R}^P$ |
| *Bloat control* | Dynamic depth control. |
| *Initial dynamic depth* | 6 levels. |
| *Hard maximum depth* | 20 levels. |
| *Selection* | Lexicographic parsimony tournament |
| *Survival* | Keep best elitism |

## 4. PROBLEM STATEMENT

As stated above, the main goal is to estimate the performance of the PGPC classifier, based on a description of a particular problem's data. To accomplish this goal,we begin by building a set $P$ of synthetic 2-class multimodal problems, where each sample in $P$ represents an individual classification problem. Second, from each problem $p \in P$ we extract a vector of descriptive features $\beta$, described in Section 5. Third, we solve each problem in $p$ using the PGPC system described in Section 3, in order to determine the performance PGPC achieves, which we characterize with the classification error $\epsilon$. After which, we have a set of problems $P$, where each $p_i \in P$ is described by a descriptive vector $\beta_i$, as well as the expected classification error $\epsilon_i$ of PGPC applied to $p_i$. Therefore, the problem we pose is that of finding an optimal estimator $K^o$ of classifier performance, such that

$$K^o = \arg\min_K \{Err[K(\beta_i), \epsilon_i]\} \ \forall p_i \in P \qquad (3)$$

where $Err[,]$ represents an error measure, which in this work is given by the root-mean-square error (RMSE).

### 4.1 Classification problems and PGPC performance

We randomly create 300 classification problems with two classes using Gaussian mixture models (GMMs), these conform set $P$. The use of randomly generated GMMs allows us to generate either unimodal or multimodal classes, with different amounts of class overlap and geometry. All problems are set in the $\mathbb{R}^2$ plane with $x, y \in [-10, 10]$, and 200 sample points were randomly generated for each class. The parameters for the GMM of each class were also randomly chosen using the following ranges of values:

- Number of Gaussian components: $\{1, 2, 3\}$.
- Median of each Gaussian component for each feature dimension: $[-3, 3]$.
- Each element of the covariant matrix of each Gaussian component: $(0, 2]$.
- The rotation angle of each covariance matrix: $[0, 2\pi]$.
- The proportion of sample points generated with each Gaussian component: $[0, 1]$.

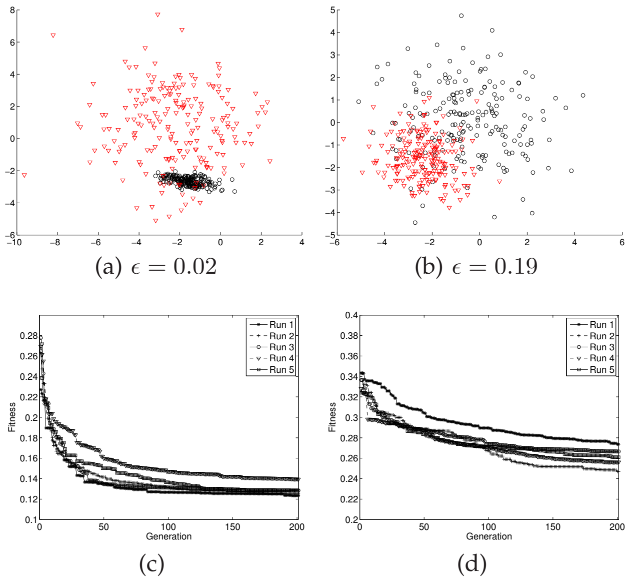


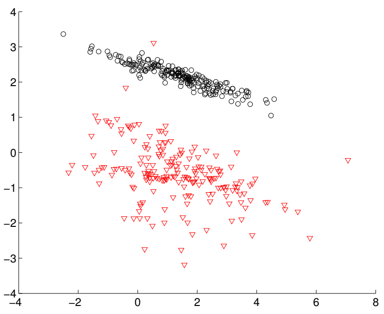
**Figure 1: Figure shows two classification problems (a,b), with the median classification error achieved by PGPC, and the five convergence plots of the best individual fitness obtained in each of the five independent runs (c,d).**

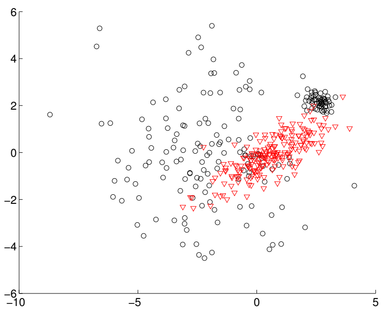
Afterwards, $\forall p \in P$ we compute the set of descriptive features $\beta$ defined in Section 5. Then, for every problem we determine the median test error of the PGPC from 5 independent runs, using 70% of the 200 sample points for fitness evaluation (training) and the rest for testing; the sets were randomly chosen at the beginning of each run. Figure 1 shows the type of performance achieved by PGPC, showing two classification problems, the median test error of PGPC, and the evolution of the best fitness from each of the five runs. The median error is used as an approximation of the expected PGPC performance on each problem instance. The convergence plots in Figure 1 illustrate that there is only a small variability in the five independent runs for each problem. In general, the average difference in classification error between the best run and the worst run for all 300 problems is only 8.3%, with a standard deviation of 4.4%, which gives some confidence regarding the accuracy of the performance estimate. Figure 2 shows three other problems with different degrees of difficulty, which increases from left to right. Notice, that the expected performance of PGPC is related to the amount of overlap between both classes and the compactness of each, as should be expected [4].

## 5. DESCRIPTORS

The research goal in this paper is to build an estimator of classifier performance based on a set of descriptive measures that are extracted from the problem itself. We consider two groups of measures, statistical measures and measures of data complexity. The statistical measures are a subset of those proposed in [24]; we choose those that produced the best results with the meta-model developed in that work. These descriptors are some basic statistics computed for each feature, or dimension, of the problem data.

- The geometric mean ratio of the pooled standard devi-


(a) $\epsilon = 0.00$


(b) $\epsilon = 0.15$

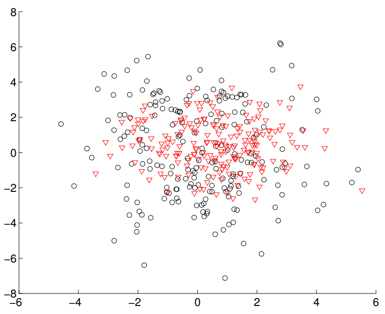

(c) $\epsilon = 0.33$

**Figure 2: Three classification problems and the median error achieved by PGPC; difficulty increases from left to right.**

ations to standard deviations of the individual populations (SD).

- The mean absolute correlation coefficients between two features (CORR) and its squared value (CORR2).
- The mean skewness of features (SKEW).
- The mean kurtosis of features (KURT).
- The average entropy of features (HX) and its squared value (HX2).

Additionally, we also employ various measures of problem complexity that consider the geometry of the distribution of samples over the feature space. These descriptors are a subset of those proposed in [4]. We choose data descriptors that consider the geometry of the distribution of points and the density, or proximity, of the points. [4] also includes descriptors based on the performance of a linear classifier, which we omit because these measures are an indirect description of the data based on a specific classifier.

*Fisher's discriminant ratio (FD).*
The discriminant ratio proposed by Fisher is defined as

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1 + \sigma_2^2} , \qquad (4)$$

where $\mu_1$, $\mu_2$, $\sigma_1$, $\sigma_2$ are the means and standard deviations of the two classes. This measure is computed independently for each feature of the data. Therefore, for a multidimensional problem the maximum $f$ over all feature dimensions is chosen as the representative value FD [4].

*Volume of overlap region (VOR).*
This measure provides an estimate of the amount of overlap between both classes in feature space. This measure is computed by finding, for each feature, the maximum and minimum value of each class and then calculating the length of the overlap region. The length obtained from each feature can then be multiplied in order to obtain a measure of volume overlap. VOR is zero when there is at least one dimension in which the two classes do not overlap.

*Feature efficiency (FE).*
It is a measure of how much each feature contributes to the separation of both classes. When there is a region of overlap between two classes on a feature dimension, then data is considered ambiguous over that region along that dimension. However, it is possible to progressively remove the ambiguity between both classes by separating those points that lie inside the overlapping region. The efficiency of each feature is defined as the fraction of all remaining points separable by that feature, and the maximum feature efficiency (FE) is taken as the representative value for a two-class problem.

*Class distance ratio (CDR).*
This is a measure that compares the dispersion within the classes to the gap between the classes, and is computed as follows [4]. For each data sample the Euclidean distance to its nearest-neighbor is computed, within and outside its own class. Then, the CDR is the ratio of the averages of all intraclass and interclass nearest-neighbor distances.

### 5.1 Correlation with PGPC performance

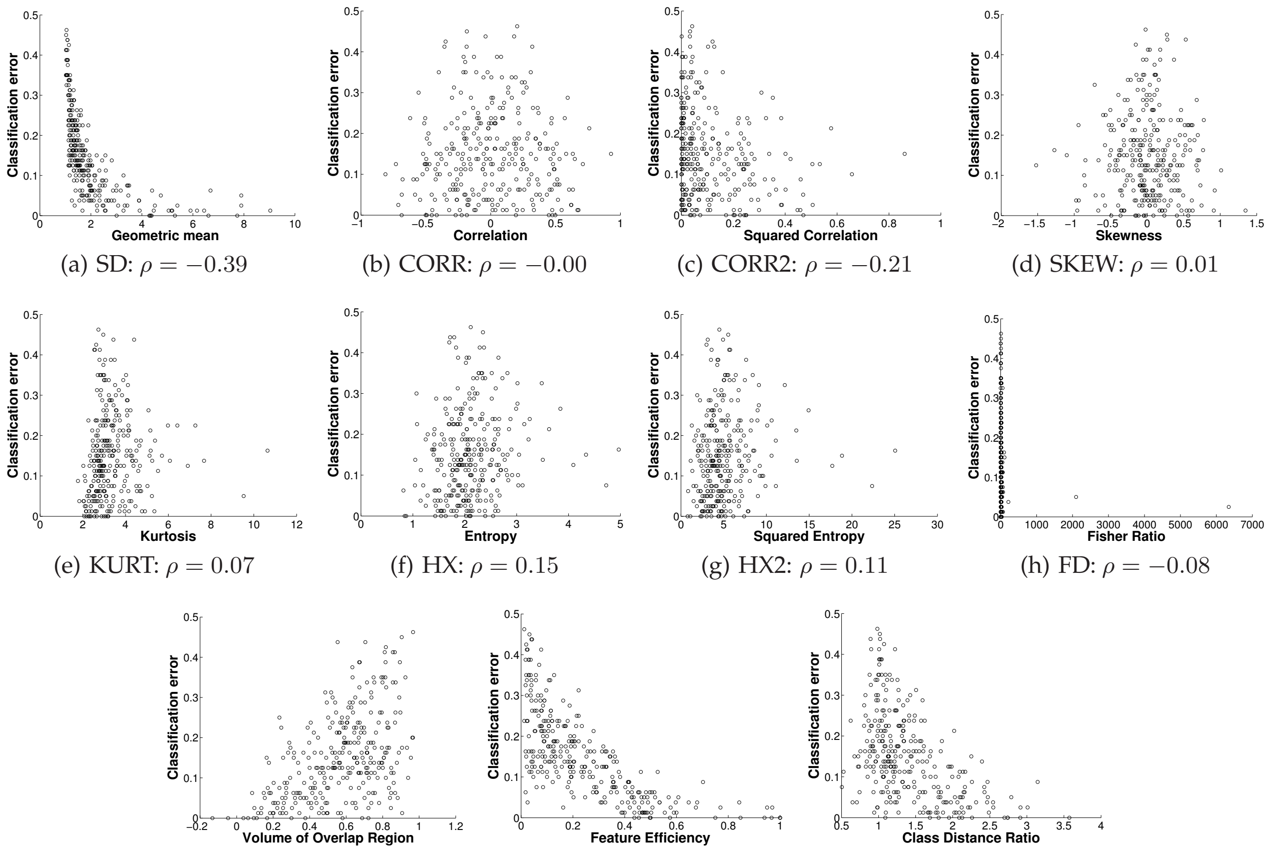
In Figure 3, we present scatter plots between the performance achieved by PGPC on each problem, and each of the data descriptors described above. These plots illustrate the relationship between each problem, represented by a descriptive value, and the classification error of PGPC. Moreover, to quantify this relationship, Figure 3 also provides the value of Pearson's correlation coefficient $\rho$. From these plots we can see that some measures are more strongly correlated with the performance of PGPC than others. More precisely, features SD, VOR, FE and CDR obtain the highest correlation scores. This result is consistent with the conclusions of [4], where measures based on the geometric distribution of the data and those based on nearest-neighbor criteria provide the most informative description of the problem. Moreover, these results are very similar to those published in [26], where the correlation of these descriptors was measured with respect to a neural network classifier.

## 6. PREDICTIVE MODELS

The estimators of PGPC performance are built using two different approaches. The first is to derive least squares linear regression models. The second is to pose a symbolic regression problem with GP, similar to the approach of [26].

### 6.1 Linear Models

In this approach, the first step is to choose which descriptors will be used as predictors in the linear models. For simplicity, we employ those descriptors that exhibit the highest



(a) SD: $\rho = -0.39$ (b) CORR: $\rho = -0.00$ (c) CORR2: $\rho = -0.21$ (d) SKEW: $\rho = 0.01$

(e) KURT: $\rho = 0.07$ (f) HX: $\rho = 0.15$ (g) HX2: $\rho = 0.11$ (h) FD: $\rho = -0.08$

(i) VOR: $\rho = 0.50$ (j) FE: $\rho = -0.71$ (k) CDR: $\rho = -0.50$

**Figure 3: Scatter plots that show the relationship between each descriptive measure of problem data (x-axis) and classification error (y-axis). Legend of each plot shows the value of Pearson's correlation coefficient $\rho$.**

correlation with PGPC performance, as discussed in the previous section; these are: SD, VOR, FE and CDR. Then, it must be determined if the predictive data exhibits multicollinearity; i.e., if the descriptors are highly correlated. Figure 4 shows scatter plots between each of the chosen descriptors and the value of their linear correlation. Given that the problem descriptors are not strongly correlated, it is possible to use standard least squares regression.

## 6.2 Symbolic regression with GP

This approach is similar to the one followed in [26], where the optimization problem of Equation 3 is solved using symbolic regression and GP. Here, the same descriptors used with the linear models makeup the terminal set $T$,

$$T = \{SD, VOR, FE, CDR\} \ . \tag{5}$$

The function set $F$, on the other hand, contains common primitive functions used for symbolic regression problems with GP; these are

$$F = \left\{+, -, *, /, \sqrt{}, sin, cos, log, x^y, |\cdot|, if\right\} \ , \tag{6}$$

with protected functions for $/$, $log$, $\sqrt{\cdot}$ and $x^y$.

Fitness is posed as a cost function, given by the RMSE computed on the set of $n$ training samples,

$$f(K) = \sqrt{\frac{\sum_{i=1}^{n} (K(\beta_i) - \epsilon_i)^2}{n}} \ , \tag{7}$$

where $\beta_i$ is the descriptive vector of problem $p_i$, and $\epsilon_i$ its classification error.

## 7. EXPERIMENTS AND RESULTS

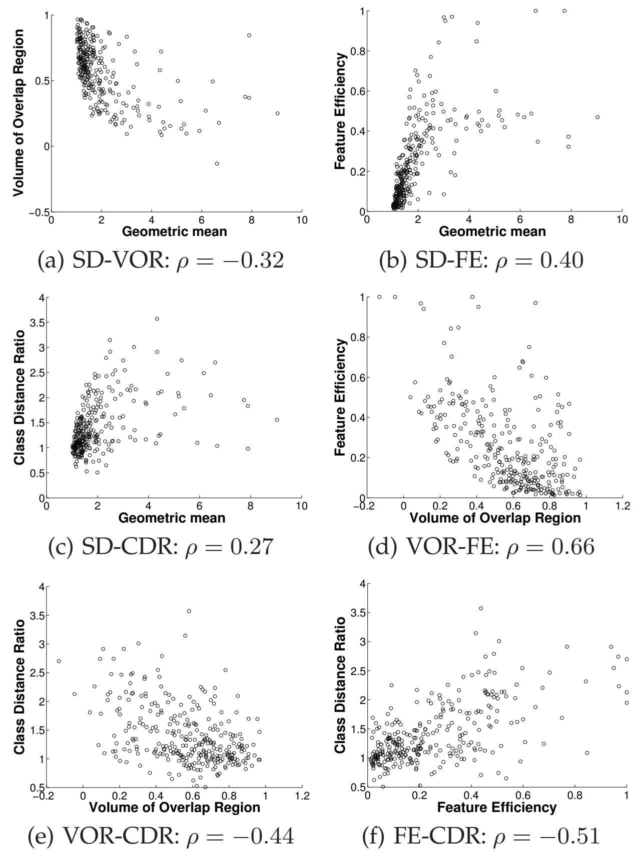
This section presents implementation details and summarizes the main results.

### 7.1 Linear Models

Four different linear models are derived using the predictive descriptors selected above, these are: (1) Linear model (L1); (2) Linear model with interaction terms (L2); (3) Linear model with interaction and quadratic terms (L3); and (4) Linear model with quadratic terms (L4). All models also contain a constant term. Each model is derived using half of the ground truth data, the rest is used for testing. Moreover, for statistical significance 30 different models are built with different random partitions of the data.

### 7.2 Symbolic regression with GP

The parameters of the GP used to evolve performance estimators for PGPC are given in Table 2. It is a Koza-style GP



(a) SD-VOR: $\rho = -0.32$ (b) SD-FE: $\rho = 0.40$ (c) SD-CDR: $\rho = 0.27$ (d) VOR-FE: $\rho = 0.66$ (e) VOR-CDR: $\rho = -0.44$ (f) FE-CDR: $\rho = -0.51$

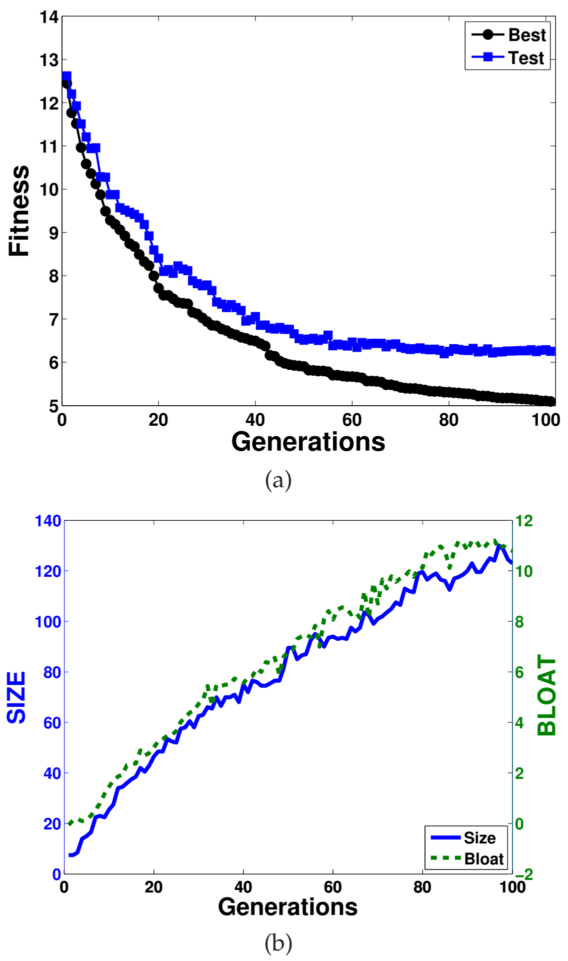
**Figure 4: Scatter plots show the correlation among the chosen descriptors: (a) SD-VOR; (b) SD-FE; (c) SD-CDR; (d) VOR-FE; (e) VOR-CDR; (f) FE-CDR. The caption of each contains the value of Pearson's correlation coefficient $\rho$.**

**Table 2: Parameters for the GP used to derive performance predictors for PGPC.**

| Parameter | Description |
|---|---|
| *Population size* | 200 individuals. |
| *Generations* | 100 generations. |
| *Initialization* | *Ramped Half-and-Half at 6 levels.* |
| *Operator probabilities.* | Cross. $p_c = 0.8$; Mutation $p_\mu = 0.2$. |
| *Bloat control* | Dynamic depth control. |
| *Initial dynamic depth* | 6 levels. |
| *Hard maximum depth* | 12 levels. |
| *Selection* | Lexicographic tournament. |
| *Tournament size* | 6 individuals. |
| *Survival* | Keep best elitism. |

with bloat control using the dynamic depth method [22] and lexicographic parsimony pressure during tournament selection [10]. The GP was executed 30 times, thus the results are statistics computed over all of the runs. In each run, 50% of the 300 classification problems were used for fitness evaluation (training) and the rest for testing. The training and test sets were randomly chosen for each run.

A summary of the evolutionary search performed by the GP is presented in Figure 5. Figure 5(a) shows the median of the best individual training fitness at each generation, and



(a) (b)

**Figure 5: Evolutionary search of the GP for performance prediction: (a) Evolution of best individual at each generation evaluated with the training and testing sets; (b) Evolution of size and bloat during the search process.**

the median of the test fitness computed with the best individual at each generation. It is evident that the GP search is not overfitted given the similarity between the training and testing fitness. Figure 5(b) illustrates the effect of bloat during the GP runs, also showing median values. This figure is a double y-axis plot, where the left axis shows the size of the best individual at each generation, given in number of nodes. The right y-axis measures the bloat incurred in the experiments using the measure proposed in [27]. It appears that most program growth is due to bloat, since both size and bloat behave in an almost identical manner.

## 7.3 Comparison

All of the predictive models are compared based on their median, average, and standard deviation of the prediction RMSE, which is summarized in Table 3. In every case, these statistics are computed over the 30 independent runs of each algorithm. Moreover, a boxplot comparison is depicted in Figure 6. In general, the figure shows that all models give a good prediction of PGPC performance. Among the linear models, the best performance is achieved by the quadratic linear model (L4). Moreover, the GP models achieve the low-

**Table 3: Comparison of the RMSE between every predictive model, obtained over thirty independent runs; bold indicates best (minimum).**

| | Median | Average | Std. |
|---|---|---|---|
| *Linear - L1* | 7.40 | 7.42 | **0.19** |
| *Interaction - L2* | 7.67 | 8.62 | 2.28 |
| *Quadratic - L3* | 7.30 | 8.36 | 3.19 |
| *Pure Quadratic - L4* | 6.80 | 7.00 | 0.92 |
| *GP* | **6.25** | **6.41** | 1.06 |

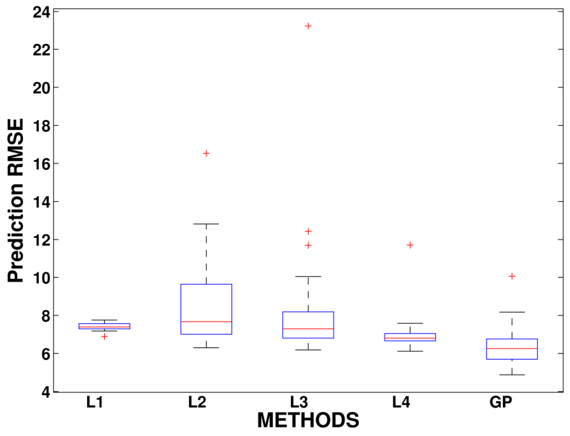


**Figure 6: Boxplot comparison of each predictive model.**

est median and average error, which is depicted in the boxplot. An ANOVA test reveals that the observed difference among the models is statistically significant with a $p$ value of 0.00. However, not all of the observed differences are statistically significant. A multiple comparison test, using the HSD criterion and a significance level $\alpha = 0.05$, reveals that the GP and L4 models are better than the L1 and L2 models, but no differences is detected with respect to L3, and no difference is detected between GP and L4.

With respect to size, it appears that the GP models are larger than the linear models, their median size is around 120 nodes, see Figure 5(b). On the other hand, the linear model L1 only contains 5 terms, L2 11 terms, L3 15 terms, and L4 9 terms. However to compare their size, we must consider all addition and multiplication operations performed by each linear model. Moreover, if we also consider the terminal elements of each model, when the linear models are expressed as GP trees then L1 requires 13 nodes, L3 43, L3 63 and L4 33 nodes. While the difference is still substantial between GP and the linear models, the difference in size appears to be a result of bloating, see Figure 5(b). Therefore, in order to improve upon the size of the predictors produced by GP, a better bloat control method is required.

## 7.4 Real-world test

Finally, the predictive models are tested on two real-world problems from the medical field, see Table 4. The first is Parkinson's diagnosis, with 23 real-valued features and 197 data samples [9]. The second is diabetes diagnosis, with 8 real-valued features and 768 samples [23]. Notice how these problems are highly multi-dimensional, which adds a degree of difficulty not present in the synthetic data used for

**Table 4: First column shows the classification error achieved by PGPC on each problem. The final five columns show the predicted classification error given by each model; bold indicates the best prediction.**

| | PGPC | L1 | L2 | L3 | L4 | GP |
|---|---|---|---|---|---|---|
| *Park.* | 23% | 13.7% | 12.4% | 17.6% | 14.3% | **23.5**% |
| *Diabetes* | 25.3% | **24.3**% | 27.3% | 27.5% | 27.5% | 42.7% |

training. For each problem, PGPC was executed 5 times and the median classification error was computed, shown in the PGPC column of Table 4. Then, SD, VOR, FE and CDR were computed on each dataset, and each predictive model was used to determine the expected performance of PGPC, the final five columns of Table 4. In these tests, the models that achieved the minimum predictive error on the set of synthetic two-class problems are used. For the Parkinson's data set, the GP model performs the best, achieving an almost perfect prediction of PGPC performance. On the other hand, the linear models perform quite poorly for this dataset. However, for the diabetes problem the results are exactly the opposite, with the linear models achieving an accurate prediction and the GP model severely over-stating the difficulty of the problem. Such discrepancies in performance show that it is not yet clear which predictive models provide the best tool for real-world situations. Nonetheless, it is evident that it is possible to accurately predict the performance of a GP classifier using the general proposal outlined in this paper.

These are promising results, because they suggest that an artificial system might be able to analyze a particular problem, and autonomously determine if the computational tools it posses can solve the problem in an acceptable manner. Moreover, if the system is endowed with several computational tools, then the system could choose which algorithm to use based on the performance it expects to achieve. Such capabilities are the type of features that a truly autonomous intelligent system might exhibit in real-world scenarios.

## 8. CONCLUDING REMARKS

This paper presents an approach to determine the difficulty of a problem for a GP-based classifier. The goal is to predict the performance, in classification error, of a particular GP approach towards data classification, based on a set of descriptors of the problem data. These descriptors are based on statistical properties of the data and on their geometric dispersion within the feature space of the problem. In order to derive the predictive models, we use a standard linear regression approach and a GP-based symbolic regression approach. Experimental evaluations show that both approaches produce models that achieve an accurate estimation of classifier performance. Moreover, tests on real-world problems show the ability of the predictive models to generalize to more complex problem instances. However, further research is still required to determine which type of model provides the best predictive performance.

Therefore, this work shows that for data classification, it is possible to accurately predict the performance of a GP system without actually running the GP search! The predictive models allow us to estimate the difficulty associated with a problem, and thus make a reasoned choice regarding the usefulness of GP. Finally, future work should look to integrate such predictors into an autonomous system that is capable of analyzing a problem and determining which computational tool is best suited to solve it.

## Acknowledgements

Second author supported by scholarship 298654 from Consejo Nacional de Ciencia y Tecnología (CONACYT), México.

## References

[1] J. Eggermont, J. N. Kok, and W. A. Kosters. Genetic programming for data classification: partitioning the search space. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 1001–1005, New York, NY, USA, 2004. ACM.

[2] E. Galván-López, S. Dignum, and R. Poli. The effects of constant neutrality on performance and problem hardness in gp. In *Proceedings of the 11th European conference on Genetic programming*, EuroGP'08, pages 312–324, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon. Defining locality as a problem difficulty measure in genetic programming. *Genet. Program. Evolv. Mach. (accepted)*.

[4] T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:289–300, March 2002.

[5] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[6] M. Kimura. *The neutral theory of molecular evolution*. Cambridge University Press., 1983.

[7] K. E. Kinnear. Fitness landscapes and difficulty in genetic programming. In *Proceedings of the First IEEE Conference on Evolutionary Computing*, pages 142–147, Piscataway, NY, 1994. IEEE Press.

[8] J. R. Koza. *Genetic programming II: automatic discovery of reusable programs*. MIT Press, Cambridge, MA, USA, 1994.

[9] M. A. Little, P. E. McSharry, E. J. Hunter, and L. O. Raming. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 56(4):1015–1022, 2008.

[10] S. Luke and L. Panait. Lexicographic parsimony pressure. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 829–836, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[11] J. McDermott, E. Galvan-Lopez, and M. O'Neill. A fine-grained view of GP locality with binary decision diagrams as ant phenotypes. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *PPSN 2010 11th International Conference on Parallel Problem Solving From Nature*, volume 6238 of *Lecture Notes in Computer Science*, pages 164–173, Krakow, Poland, 2010. Springer.

[12] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, editors. *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA, 1994.

[13] L. C. Molina, L. Belanche, and A. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 306–324, Washington, DC, USA, 2002. IEEE Computer Society.

[14] A. E. Nix and M. D. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.

[15] R. Poli. A simple but theoretically-motivated method to control bloat in genetic programming. In C. Ryan, T. Soule, M. Keijzer, E. P. K. Tsang, R. Poli, and E. Costa, editors, *Genetic Programming, 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003. Proceedings*, volume 2610 of *Lecture Notes in Computer Science*, pages 204–217. Springer, 2003.

[16] R. Poli and N. F. McPhee. General schema theory for genetic programming with subtree-swapping crossover: Part i. *Evol. Comput.*, 11(1):53–66, 2003.

[17] R. Poli and N. F. McPhee. General schema theory for genetic programming with subtree-swapping crossover: Part ii. *Evol. Comput.*, 11(2):169–206, 2003.

[18] R. Poli and N. F. McPhee. Parsimony pressure made easy. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1267–1274, New York, NY, USA, 2008. ACM.

[19] R. Poli, N. F. McPhee, and J. E. Rowe. Exact schema theory and markov chain models for genetic programming and variable-length genetic algorithms with homologous crossover. *Genet. Program. Evolv. Mach.*, 5:31–70, March 2004.

[20] R. Poli and L. Vanneschi. Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1335–1342, New York, NY, USA, 2007. ACM.

[21] S. Silva and J. Almeida. Gplab–a genetic programming toolbox for matlab. In L. Gregersen, editor, *Proceedings of the Nordic MATLAB conference*, pages 273–278, 2003.

[22] S. Silva and E. Costa. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genet. Program. Evolv. Mach.*, 10(2):141–179, 2009.

[23] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. *Johns Hopkins APL Technical Digest*, 10:262–266, 1988.

[24] S. Y. Sohn. Meta analysis of classification algorithms for pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21:1137–1144, November 1999.

[25] M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evol. Comput.*, 13:213–239, June 2005.

[26] L. Trujillo, Y. Martínez, and P. Melin. Estimating classifier performance with genetic programming. In S. Silva et al., editor, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 275–286, Turin, Italy, 2011. Springer Verlag.

[27] L. Vanneschi, M. Castelli, and S. Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 877–884, New York, NY, USA, 2010. ACM.

[28] L. Vanneschi, M. Tomassini, P. Collard, S. Vérel, Y. Pirola, and G. Mauri. A comprehensive view of fitness landscapes with neutrality and fitness clouds. In *Proceedings of the 10th European conference on Genetic programming*, EuroGP'07, pages 241–250, Berlin, Heidelberg, 2007. Springer-Verlag.

[29] T. Yu and J. F. Miller. Neutrality and the evolvability of boolean function landscape. In *Proceedings of the 4th European Conference on Genetic Programming*, EuroGP '01, pages 204–217, London, UK, 2001. Springer-Verlag.

[30] M. Zhang and W. Smart. Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recogn. Lett.*, 27:1266–1274, August 2006.