Instance Selection for Geometric Semantic Genetic Programming

Luis Fernando Miranda, Luiz Otavio V. B. Oliveira, Joao Francisco B. S. Martins, Gisele L. Pappa
Computer Science Department, Federal University of Minas Gerais, Belo Horizonte, Brazil
{luisfmiranda, luizvbo, joaofbsm, glpappa}@dcc.ufmg.br

***Abstract*—Geometric Semantic Genetic Programming (GSGP) is a method that exploits the geometric properties describing the spatial relationship between possible solutions to a problem in an n-dimensional semantic space. In symbolic regression problems, n is equal to the number of training instances. Although very effective, the GSGP semantic space can become excessively big in most real applications, where the value of n is high, having a negative impact on the effectiveness of the GSGP search process. This paper tackles this problem by reducing the dimensionality of GSGP semantic space in symbolic regression problems using instance selection methods. Our approach relies on weighting functions—to estimate the relative importance of each instance based on its position with respect to its nearest neighbours—and on dimensionality reduction techniques—to improve the notion of closeness between instances, generating datasets with simplified input spaces. Experiments were performed on a set of 15 datasets and our experimental analysis shows that using instance selection by instance weighting and dimensionality reduction does improve the effectiveness of the search with almost no impact on root mean square error results.**

***Index Terms*—geometric semantic genetic programming, instance selection, semantic space, symbolic regression**

I. Introduction

Data regression is among the most popular machine learning tasks [1]. Given a set of training cases, each of them described by a set of inputs and a scalar output, the task of regression induces a model capable of mapping inputs to outputs. Within the field of evolutionary computation, methods for symbolic regression have been extensively studied and overall successful in solving a great variety of problems [2]. In contrast with other regression models, in which the shape of the function to be induced is previously defined (e.g. linear) and its parameters optimized, symbolic regression defines simultaneously the shape of the function and its parameters.

As any other regression method, symbolic regression requires as input a set of examples to learn the regression function from, named the training set. Usually people believe that the larger the training set, the better will be the model induced by the regression method. But this is not always true. In some datasets, certain regions of the input space may be excessively well-covered, while others may lack representativeness. This can bias regression models induced by learning methods to perform well only on these over-represented regions, decreasing their generalization capability. In this scenario, removing instances from these dense regions can improve the induced model by leading the regression model to consider the entire input space with similar interest [3]. In addition, making the training set smaller by removing instances can reduce the computational effort employed to induce the regression model, helping applications in which the computational time spent during the learning process has crucial importance.

Following this idea of instance selection, this paper explores its use and advantages in another scenario: to reduce the number of dimensions of the semantic space in Geometric Semantic Genetic Programming (GSGP) [4] for symbolic regression. GSGP is a Genetic Programming (GP) framework that introduces geometric semantic operators capable of inducing a known semantic effect through syntactic manipulation. The semantics of a candidate solution in GSGP captures its behavior considering the problem context. In symbolic regression problems the semantics of a given solution is defined as the output vector generated when the function it represents is applied to the training set [5].

This definition allows the semantics of a solution to be represented in a n-dimensional space, called semantic space, in which n corresponds to the size of the training set. The genetic operators employed by GSGP use the geometry of this space to search for solutions that minimize the fitness function, defined with respect to a given metric. The fitness value is proportional to the distance between the solution and the target output vector—also represented in the semantic space—according to the adopted metric. Thus, by reducing the size of the training set, the number of dimensions of the semantic space is also decreased, leading to a smaller search space, which can be simpler to be explored.

The main goal of this paper is to assess if, by decreasing the number of training cases and, consequently, the number of dimensions of the semantic space, we can improve the search performed by GSGP, making that search simpler and more efficient. In an initial analysis performed in [6] we applied instance selection methods adapted from the data classification literature, and showed that the reduction of the size of the search space can bring beneficial impacts to the search performed by GSGP.

Following this same line, this paper proposes a different approach for instance selection: to weight data instances. We propose to use four weighting functions to estimate the relative importance of each instance based on its position with respect to its k nearest neighbours. These weighting function are combined with four dimensionality reduction techniques, which we show are important to improve the

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

notion of closeness between instances, generating datasets with simplified input spaces. Results show this approach results in the construction of instances subsets capable of capturing the underlying structure of the datasets, allowing GSGP to induce regression models with similar quality faster.

The remainder of this paper is organized as follows. Section II introduces the main concepts of GSGP. Section III reviews related work, while Section IV describes and evaluates our strategies for reducing the size of the search space in GSGP. Finally, Section V presents experimental results, and Section VI draws conclusions and presents directions for future work.

II. Geometric Semantic Genetic Programming

Recent works in GP have shown that the semantics of the programs can play a crucial role during the evolutionary process [7]. Exploring this scenario, researchers have proposed a variety of methods that employ semantically-aware operators capable of guiding the search towards more promising regions of the search space, improving the chances of reaching better solutions. In contrast with syntax, which defines the structure of the program represented by an individual, semantics describes the behaviour of this structure.

Following the concept of semantic space, Moraglio and colleagues [4] presented a new GP framework, capable of manipulating the syntax of the individuals with geometric implications on their disposition in the semantic space. The framework, called Geometric Semantic GP (GSGP), searches directly the space of the underlying semantics of the programs, inducing a unimodal fitness landscape—which can be optimized by evolutionary algorithms with good results for virtually any metric [8]. The geometric semantic crossover and mutation operators presented by GSGP exploit semantic awareness and geometric shapes to describe the spatial relationship between parents and offspring, inducing precise geometric properties in the semantic space.

The semantics of a GSGP individual is defined as a point in a space with dimensionality equivalent to the number of training instances, which implies that geometric semantic operators are affected by the growth of the dimensionality of the semantic space [6]. Therefore, by reducing the number of instances we automatically reduce the number of dimensions of the semantic space, which in turn reduces the complexity of the search space. The smaller the complexity of the search space, the smaller the number of possible combinations, which may increase the speed of convergence to the optimum.

III. Related Work

Although the importance of Instance Selection (IS) methods is recognized in both classification and regression tasks, the number of works applying IS to regression problems is still much smaller than those applying IS to classification (see [9] for a comprehensive survey of these techniques or [10] for a study on how IS can improve classification accuracy on imbalanced datasets).

The CNN for Regression (RegCNN) and ENN for Regression (RegENN) [11] adapt the Condensed Nearest Neighbour (CNN) and Edited Nearest Neighbour (ENN) methods for instance selection in classification problems to the regression domain. Both methods estimate the importance of each instance by analyzing how they compare to their neighbours. RegCNN and RegENN replace the label comparison used in their classification versions by an error-based comparison. Instead of comparing the label predicted by a k-NN classifier and the expected label to make a decision, RegCNN and RegENN compare the error between the output predicted by a regression method and the expected output to a threshold, in order to make the decisions of removing (RegENN) or keeping (RegCNN) an instance. The RegCNN and RegENN methods were renamed Threshold ENN (TENN) and Threshold CNN (TCNN) in [12] and compared with a discretization approach, which converts the continuous outputs of the instances into discrete values representing their labels and then applies the original ENN and CNN to select the instances.

In [6], we evaluated methods used during data pre-processing (namely TENN and TCNN) for instance selection in GSGP, and also proposed the Probabilistic instance Selection based on the Error (PSE), a method integrated to GSGP evolution that selects a subset of instances from the original training set—with a user-defined frequency—with probability proportional to the median absolute error of the instance over the current population. Results showed that TENN and TCNN had inferior performance in terms of RMSE in this task when compared to PSE. However, the main drawback of PSE is that it further increases the computational time required by GSGP to induce the regression models. Therefore, it is still relevant to investigate other pre-processing strategies with a smaller impact on the computational time of regression methods.

In this direction, the approach presented in this paper is based on instance weighting. We took as inspiration the work of [13], in which the authors introduced a framework to automatically assign weights to instances that takes into account the relative importance of the instance. However, instead of using a canonical version of the GP algorithm, we analyse the impact of these weights on the semantic space in which the search performed by GSGP takes place. They evaluated the framework using four weighting schemes, defining the importance of an instance relative to proximity, surrounding, remoteness, and nonlinear deviation from k nearest-in-the-input-space neighbours. They explored their framework in two different ways: (i) by using the weights within the fitness function of a canonical GP, giving different importance to each instance on the final fitness value and (ii) by selecting a subset from the training set, composed by the instances with the highest metric value. Regarding the former approach, they developed a simple procedure for partitioning data into balanced nested subsets of arbitrary size, called Simple Multidimensional Iterative Technique for Subsampling (SMITS). The pre-processing instance selection strategy presented in this paper is based on the former approach and Section IV provides further details on that.

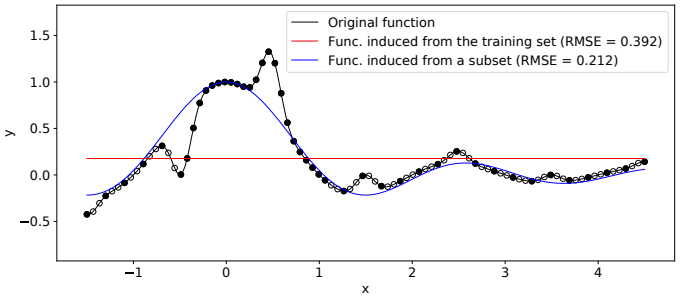

Fig. 1: Example of an unbalanced dataset and the impact of a instance selection method on the regression.

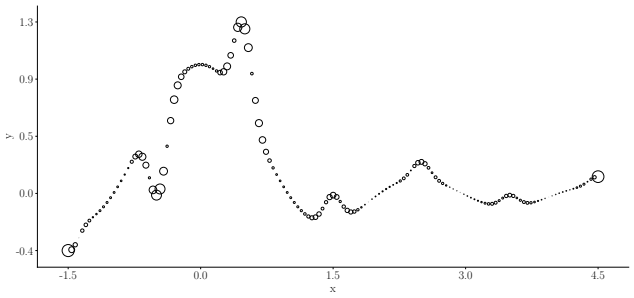

Fig. 2: Weights obtained by applying the surrounding function to the training set from Fig. 1.

IV. Strategies for Reducing the Size of the Search Space

This section presents the proposed approach to deal with instance selection: it assigns weights to each instance according to their importance in the definition of the search space, then selects a subset of these instances according to their weight. We first introduce the four weighting metrics we apply: proximity, surrounding, non-linearity and remoteness [13]. Given that these metrics are calculated in a very high-dimensional space, we also investigate the use of four dimensionality reduction techniques in the original space of features of each instance to improve the way the weights are calculated. These techniques are: Principal Component Analysis (PCA), Isomap Mapping, Multi-Dimensional Scaling (MDS) and t-distributed Stochastic Neighbour Embedding (t-SNE).

The idea of using instance weights to select instances is based on the hypothesis that the level of concentration of instances in the input space has a crucial impact on the training stage performed by regression methods, including GP-based ones. This hypothesis comes from the fact that error metrics used to guide the search performed by GP give the same weight to every instance. Consequently, regions of the input space with a higher concentration of instances bias the search, in the sense that the fitness function gives a better reward to individuals with good performance on these dense regions.

Fig. 1 gives an example of such behavior. Consider a training set $\mathcal{T}$ composed by 60 points evenly distributed in the input space in $[-1.5, 4.5]$, represented as circles (filled and empty) in the figure. Notice that the intervals $[-1.5, -0.5]$ and $[1, 4.5]$, although with the same distribution of instances in the input space, have a denser distribution in the output space when compared to the interval $(-0.5, 1)$. In order to make the distribution in the output space more balanced, we selected the subset $\mathcal{S}$ from $\mathcal{T}$, represented by filled circles. The red and blue curves represent functions induced by a GP using $\mathcal{T}$ and $\mathcal{S}$ as training sets, respectively. The red curve converges to a constant that minimizes the error in the dense region, while the blue curve is able to capture the tendency of the original function.

In such cases, instance selection methods can be useful, as they can prevent the creation of a model that is inclined to perform well only on regions with a higher instance density and, in addition, they have the potential to reduce running time without reducing accuracy. We are interested in a method capable of identifying instances with low variation in the output space w.r.t. cases in the same region. These instances could be removed from the training set, resulting in a smaller set in which the distribution of the instances in the input space better reflects the variations in the output space.

A. Weighting Process

We first introduce the notation used in the remainder of this work. Given the input training set $\mathcal{T} = \{I_1, I_2, \ldots, I_n\}$—with $I_i = (\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ and $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$, for $i = \{1, 2, \ldots, n\}$, where n is the size of the training set and d is the number of dimensions in the input space. The p-norm in the input space of a given vector $\mathbf{u} \in \mathbb{R}^d$, defined as:

$$\|\mathbf{u}\|_p = \left(\sum_{i=1}^{d} |u_i|^p \right)^{1/p}, \tag{1}$$

is used to compute the subset of $\mathcal{T}$ corresponding to the k nearest neighbours of a given instance I_i—$\mathcal{N}_i = \{N_{i1}, N_{i2}, \ldots, N_{ik}\}$—and its set of associates $\mathcal{A}_i = \{I_j | I_i \in \mathcal{N}_j \text{ for } j \in \{1, 2, \ldots, n\} \text{ and } i \neq j\}$—i.e., instances that have I_i among their k-nearest neighbours. We used $p = 2$ in all experiments performed.

The proximity function γ, introduced by the authors in [14], tries to estimate how isolated an instance is by measuring the average distance to its k nearest neighbours:

$$\gamma(I_i, \mathcal{N}_i, k) = \frac{1}{k} \sum_{(\mathbf{x}_j, y_j) \in \mathcal{N}_i} \|\mathbf{x}_i - \mathbf{x}_j\|_p \ . \tag{2}$$

However, the proximity function does not take into account the relative directions between an instance and its neighbours. Therefore, a situation like the one depicted in Fig. 3 would be indistinguishable by this metric. This kind of information can be captured using the surrounding function δ, which tries to identify instances on the border of the response surface—i.e., instances that are not uniformly surrounded by its neighbours—by measuring the average length of the vectors pointing from I_i to its k nearest neighbours:

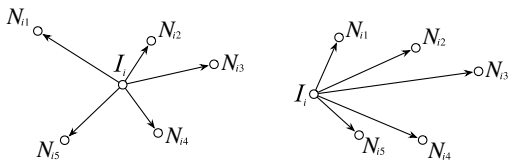

Fig. 3: Since the surrounding function also takes the directions to the neighbours into account, it assigns higher weight values when the neighbours are located in the same direction.

$$\delta(I_i, \mathcal{N}_i, k) = \left\| \frac{1}{k} \sum_{(\mathbf{x}_j, y_j) \in \mathcal{N}_i} (\mathbf{x}_i - \mathbf{x}_j) \right\|_p . \quad (3)$$

The choice of k influences the perception of the data: smaller values for k increase the local impact on the metrics while larger values lead to a more global influence. When determining the neighbours and associates of the instances, only the input space is considered.

To illustrate how the surrounding function works, we apply it to the set of instances presented in Fig. 1. The result is shown in Fig. 2, in which the size of each instance is proportional to its weight.

The nonlinearity function ν tries to emphasize areas of nonlinear changes and is defined as the distance from an instance I_i to the least-squares hyperplane passing through its k nearest neighbours:

$$\nu(I_i, \mathcal{N}_i, k) = \|\mathbf{x}_i - \mathbf{x}_{\Pi_i})\|_p \ , \quad (4)$$

where $\mathbf{x}_{\Pi_i}$ is the orthogonal projection of $\mathbf{x}_i$ on the hyperplane Π_i passing through the k neighbours of I_i in the input space.

The last weighting function, remoteness (ρ), defines the weight of an instance I_i as the rank of the average value of its proximity and the surrounding weights:

$$\rho(\gamma, \delta, I_i, \mathcal{N}_i, k) = \frac{R_i^{(\gamma)} + R_i^{(\delta)}}{2}. \quad (5)$$

The weighting functions presented above allow us to rank the instances according to their inferred importance. Let w be one of these weighting functions. The vector $[w(I_1, \mathcal{N}_1, k), w(I_2, \mathcal{N}_2, k), \ldots, w(I_n, \mathcal{N}_n, k)]^T$ results from the application of w to each instance from $\mathcal{T}$ considering its k nearest neighbours. By ordering the instances according to their weights we induce a ranking $R^{(w)}$, with $R_i^{(w)}$ as the index of instance I_i in this ranking.

B. Input Space Dimensionality Reduction

Most datasets used in real-world applications—and in our experiments—represent a complex high dimensional input space. The weighting functions presented in the last section rely heavily on the notion of closeness between each instance and their nearest neighbours, which can be deceiving in high dimensional spaces. As an attempt to mitigate this potential problem, as well as gain some insight about the datasets, we applied dimensionality reduction techniques before performing the instance selection process.

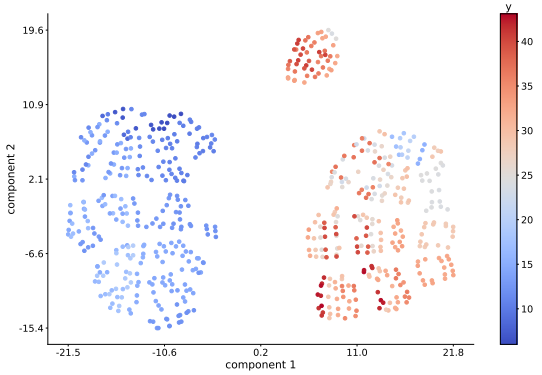

Fig. 4: Embedding created by applying the t-SNE technique on the dataset *energy-heating* (for two components).

Dimensionality reduction methods allow us to convert a high-dimensional dataset into a two or three-dimensional space, with the distances between instances in the low-dimensional representation reflecting, as much as possible, the similarities between instances in the high-dimensional dataset.

By incorporating dimensionality reduction methods in our instance selection process we expect to enhance the perception of adjacency between instances, eventually improving the significance of the values produced by the weighting functions. We embedded the input attributes of all datasets containing four or more attributes into a two dimensional input space using the following dimensionality reduction techniques:

- Principal Component Analysis (PCA) [15]: uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.
- Isomap Mapping [16]: seeks a lower-dimensional embedding maintaining geodesic distances between all points.
- Multi-Dimensional Scaling (MDS) [17]: seeks a low-dimensional representation of the data in which the distances respect well the distances in the original high-dimensional space.
- t-distributed Stochastic Neighbour Embedding (t-SNE) [18]: converts affinities of data points to probabilities. The affinities in the original space are represented by Gaussian joint probabilities and the affinities in the embedded space are represented by Student's t-distributions. Fig. 4 shows one of the ways the *energy-cooling* dataset can be represented after applying the t-SNE technique, in which the method was able to create a clear distinction between the instances, considering their output values.

C. Selection Process

We adopted the instance selection method presented in Algorithm 1, which is based on the SMITS procedure [13]. It takes a training set $\mathcal{T}$, a weighting function and five control variables as arguments and outputs a set of instances $\mathcal{S}$, selected according to the amount of new information the

Algorithm 1: Instance selection

Input: training set ($\mathcal{T}$), number of neighbours (k), distance metric (L), weighting function (w), selection factor (s)
Output: instance set $\mathcal{S} \subseteq \mathcal{T}$

1 $D \leftarrow distBetweenInstances(\mathcal{T}, L)$;
2 $\mathcal{A}_i \leftarrow \emptyset$, for $i \in \{1, 2, \ldots, n\}$; // Associates
3 **foreach** $I_i = (\mathbf{x}_i, y_i) \in \mathcal{T}$ **do**
4 $\quad \mathcal{N}_i \leftarrow k$ nearest neighbours of I_i;
5 $\quad$ **foreach** $N_{ij} \in \mathcal{N}_i$ **do**
6 $\quad\quad \mathcal{A}_j \leftarrow \mathcal{A}_j \cup \{I_i\}$; // Find associates
7 $W \leftarrow calculateWeights(\mathcal{T}, D, w, L)$;
8 Let $R[|\mathcal{T}| - k]$ be a new array; // Ranking
9 **for** $i \leftarrow 1$ **to** $|\mathcal{T}| - k$ **do**
10 $\quad I_l \leftarrow$ instance with the lowest weight;
11 $\quad R[i] \leftarrow I_l$;
12 $\quad W_l \leftarrow \infty$; // Ignore I_l thereafter
13 $\quad updateAssociatesWeights(D, W, \mathcal{A}_l, l)$;
14 Randomly assign ranks from $|\mathcal{T}| - k + 1$ to $|\mathcal{T}|$ to the last k instances;
15 $\mathcal{S} \leftarrow selectInstances(\mathcal{T}, R, s)$;
16 **return** $\mathcal{S}$;

instances bring to the selected subset. The algorithm starts by creating, in line 1, a matrix D containing the distance between each pair of instances in $\mathcal{T}$. Note that the distances are calculated considering only the input space, since the closeness between them can be misleading if we also include the output space. In lines 3-6, the algorithm builds, for each instance, its initial sets of neighbours and associates (i.e., those instances that have the instance as one of their k nearest neighbours). In lines 9-13, the algorithm iteratively ranks all instances according to the given weighting function w, the distance metric L and the number of neighbours k. It starts each iteration by finding the instance with the lowest weight and registering it in a ranking array. That instance has to be ignored thereafter, which is forced by setting its weight to ∞. After that, the algorithm updates the weights of the instances that had I_l among its k nearest neighbours (line 13). The algorithm terminates after creating, in line 15, the set $\mathcal{S}$ with the selected instances, which is done by selecting a subset of instances from $\mathcal{T}$ based on the ranking created. The size of the subset is defined by the parameter s.

The time taken by Algorithm 1 depends on the number of instances n, the number of dimensions d, and the number of neighbours k of each instance. In line 2 of Algorithm 1, we compute the distance matrix, which requires $O(n^2 d)$ operations. The proximity and surrounding functions require all k neighbours, which can be found in $O(kn)$ (using k times selection in expected linear time) or $O(n \log n)$ if we sort the neighbours. In our implementation, we opted for the latter approach for considering that large values of k will often be used. Therefore, the complexity of the proximity and surrounding functions is $O(n^2 d + n^2\, max(k, \log n))$.

For the nonlinearity function, the process of determining the plane approximating k neighbours requires solving a system of k linear equations, which makes the complexity of the nonlinearity function equals to $O(nk^3)$.

TABLE I: Datasets used in the experiments.

Dataset	Attrs	Inst.	Nature	Src	Exp. estrategy
airfoil (air)	6	1503	Real	[19], [21]	10 × 5-CV
ccn	123	1994	Real	[19]	10 × 5-CV
ccun	125	1994	Real	[19]	10 × 5-CV
concrete (con)	9	1030	Real	[19], [21]	10 × 5-CV
energyCooling (eneC)	9	768	Real	[19], [21]	10 × 5-CV
energyHeating (eneH)	9	768	Real	[19], [21]	10 × 5-CV
keijzer-6 (kei6)	2	50	Synthetic	[23]	50 × D
keijzer-7 (kei7)	2	100	Synthetic	[23]	50 × D
parkinsons (par)	19	5875	Real	[19]	10 × 5-CV
ppb	627	131	Real	[21]	10 × 5-CV
towerData (tow)	26	4999	Real	[21]	10 × 5-CV
vladislavleva-1 (vla1)	3	100	Synthetic	[24]	10 × 5-ND
wineRed (winR)	12	1599	Real	[19], [21]	10 × 5-CV
wineWhite (winW)	12	4898	Real	[19], [21]	10 × 5-CV
yacht (yac)	7	308	Real	[19], [21]	10 × 5-CV

V. Experimental Analysis

We carried out the experiments using a group of 15 datasets selected from the UCI machine learning repository [19], GP benchmarks [20], and GP studies from the literature [21], [22], as presented in Table I [1]. For real datasets, we randomly partitioned the data into 5 disjoint sets of the same size and executed the methods 10 times with a 5-fold cross-validations (10 × 5-CV). For the synthetic ones, we used two different strategies according to the way the dataset was defined in its original work: datasets generated by non-deterministic sampling functions were resampled 5 times and the experiments were repeated ten times for each sampling (10 × 5-ND); experiments with datasets deterministically sampled were repeated 50 times with the same data folds (50 × D). Training and test sets were sampled with the same strategy. In the end, all methods were executed 50 times. The sets of selected instances used were kept fixed throughout all executions.

All executions used a population of 1,000 individuals evolved for 250 generations and a tournament selection size of 10. The grow method [25] was adopted to generate the random functions inside the geometric semantic operators, while the ramped half-and-half method [25] was used to generate the initial population, both with maximum individual depth equal to 6. The terminal set included the input variables of each dataset and constant values randomly picked from the interval $[-1, 1]$. The function set included three binary arithmetic operators $(+, -, \times)$ and the analytic quotient (AQ) [26].

We employed the crossover for Manhattan-based fitness function and mutation operators, both with probability 0.5. The mutation step required by the mutation operator was defined as 10% of the standard deviation of the outputs given by the

[1]The code used in our experimental analysis is freely available for download on GitHub at https://github.com/laic-ufmg/ISR.

training data. We used the root mean squared error (RMSE) as the fitness function.

To avoid that the difference between the range of attribute values bias the data weighting process towards high range attributes, we scaled the input and output values of all datasets in our test bed to the interval $[0, 1]$. Note, however, that although these scaled values were used by the instance selection method to decide which instances should be kept, the resulting subset is always formed by the original training instances.

A. Results of Weighting Functions

In this section, we focus on the impact the instance selection process has in the search performed by GSGP. To quantify this impact, we employ two metrics: test RMSE—to measure the error associated with the regression models produced—and execution time—aiming to assess if the selection process actually reduces computational complexity.

For each dataset, we fed the algorithm with the same set of parameters, with exception of the number of instances removed from the training set, resulting in training sets of sizes ranging from 75% to 99% relative to their original sizes.

We carried out experiments using subsets built with the four weighting functions previously introduced. However, since the *ppb* dataset has more attributes than instances, the nonlinearity function could not be applied. In order to identify significant differences between the overall accuracy obtained by them, we adopted the Friedman with Nemenyi post-hoc test. We performed a Friedman test under the null hypothesis that the performances of the weighting functions are equal. Nevertheless, considering a confidence level of 95%, the resulting p-value (0.18) implies that we cannot discard the null hypothesis, and hence no weighting function can be considered better than the others. Therefore, we present in this section only the results regarding the surrounding function, which obtained the best overall results. The corresponding median test RMSE values for the experiment are presented in Table II, in which values highlighted in bold correspond to results better than the ones obtained by GSGP with the complete set of instances.

In order to get a visual overview of the results, we analyzed the percentage variation in the test RMSE value as the number of removed instances increases, shown in Fig. 5. Observe that most lines are within the 5% variation, which means that, for most datasets, the selection process does not have a strong impact on the regression performed by GSGP. In other words, the compressed training sets successfully capture the underlying structure of the data. More precisely, for 12 out of 15 datasets, the test RMSE values reveal no substantial quality changes when compared to the models built using the original and the compressed datasets, since, for any selection level, the corresponding RMSE variations are confined to the range $[-5\%, 5\%]$. Furthermore, we observed negative RMSE variation after removing 25% of the training instances when compared to the original dataset in 9 out of 15 datasets.

We observed poor or inconclusive results—in which the error value seems to grow or shift arbitrarily as we increase the selection level—for the synthetic datasets. For the *keijzer-6* and *keijzer-7* datasets, this behavior could be explained by the fact that we used an odd neighbourhood size (5) in order to assign a weight value to instances with only one input attribute, with values equally distributed along a single dimension. In such cases, the initial weights assigned to instances that are not on the edge of the input space are certainly flawed. As the selection progresses, this problem tends to be reduced, but not mitigated. However, if the neighbourhood size was, in fact, the only reason behind these results, the behavior of these two datasets should change when we use the nonlinearity function, since for them we used $k = 2$. What we see, however, is a reduction in the level of randomness of the results, but with error values still indicating poor results when compared to those obtained in the other datasets.

It is also interesting to point out that the three datasets with poor results are also those with the lowest number of input attributes (one and two). This may indicate that using a low neighbourhood size—close to the number of input attributes—impairs the selection process. In conclusion, considering the

TABLE II: Test RMSE obtained by GSGP on a training set reduced using the surrounding function.

	Training instances removed (%)						
Dataset	**0**	**1**	**5**	**10**	**15**	**20**	**25**
air	27.083	27.213	**27.073**	27.142	**27.013**	**26.993**	**26.969**
ccn	0.139	**0.138**	**0.139**	**0.139**	**0.139**	**0.139**	0.140
ccun	382.00	385.84	**380.98**	382.71	383.36	**381.17**	**382.00**
con	6.871	**6.806**	**6.795**	**6.828**	**6.862**	**6.854**	**6.851**
eneC	2.422	**2.414**	**2.389**	**2.368**	**2.330**	**2.337**	**2.315**
eneH	1.913	**1.843**	**1.878**	**1.847**	**1.845**	**1.844**	**1.872**
kei6	0.454	**0.422**	0.458	0.468	0.489	0.500	0.477
kei7	0.025	0.027	**0.025**	**0.024**	0.026	0.027	0.027
par	9.805	**9.804**	9.816	9.823	9.839	9.840	9.836
ppb	29.697	**28.805**	**28.367**	**28.615**	**29.549**	**29.057**	**28.738**
tow	33.799	**33.797**	**33.579**	**33.675**	**33.573**	**33.779**	**33.790**
vla1	0.061	0.064	**0.061**	**0.057**	**0.058**	**0.056**	**0.056**
winR	0.629	0.631	0.630	**0.628**	0.630	0.632	0.633
winW	0.719	**0.719**	0.720	0.720	0.721	0.721	0.722
yac	6.251	**6.213**	**6.135**	**6.070**	**6.177**	**6.085**	**5.956**
# of wins	-	10	11	10	9	10	9

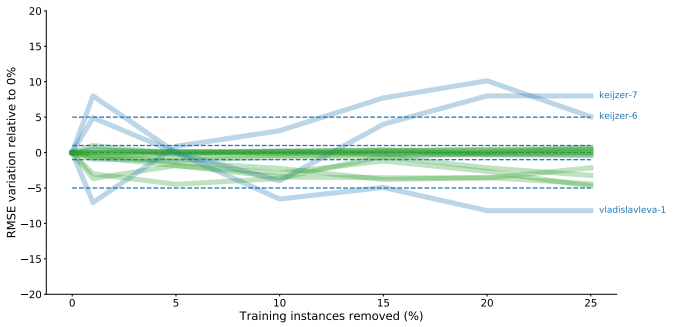

Fig. 5: Evolution of the RMSE values obtained by GSGP according to the number of instances removed using the surrounding function. Each line corresponds to a dataset, and the outer dashed blue lines represent RMSE variations corresponding to -5% and 5%, delimiting a range in which the results can be seen as stable.

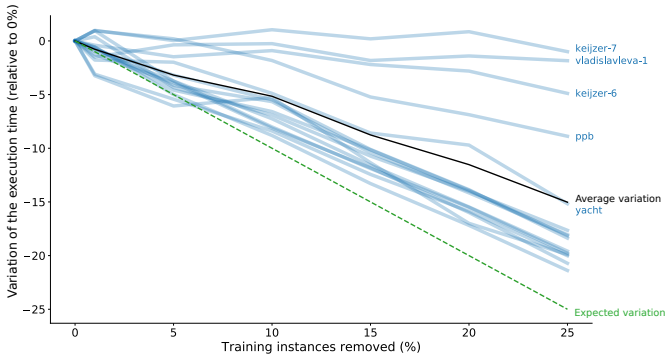

Fig. 6: Variation of the median execution time for GSGP runs using the proximity function, for each dataset. The reduction in the execution time is expected to be linear according to the number of instances removed.

TABLE III: Test RMSE obtained by GSGP on a training set embedded using the *t-SNE* method and reduced using the proximity function.

	Training instances removed (%)						
Dataset	**0**	**1**	**5**	**10**	**15**	**20**	**25**
air	27.083	27.139	**27.066**	27.086	27.140	27.296	27.202
ccn	0.139	**0.138**	**0.138**	**0.138**	**0.138**	**0.138**	**0.137**
ccun	382.00	**380.17**	**380.51**	**381.74**	386.63	383.92	386.10
con	6.871	**6.783**	6.944	**6.800**	**6.803**	**6.815**	7.031
eneC	2.422	**2.375**	**2.365**	**2.354**	**2.323**	**2.293**	**2.329**
eneH	1.913	1.938	**1.902**	**1.853**	**1.869**	**1.846**	**1.806**
par	9.805	**9.805**	9.835	9.837	9.837	9.851	9.864
ppb	29.222	**28.848**	**29.157**	**27.498**	**29.148**	29.449	29.618
tow	33.799	34.066	34.211	34.070	**33.691**	34.006	34.278
winR	0.629	**0.629**	0.630	0.631	0.633	0.633	0.636
winW	0.719	0.720	**0.719**	0.720	0.722	0.724	0.725
yac	6.251	**6.243**	**6.176**	**6.230**	**6.137**	**6.182**	**6.215**
# of wins	-	8	8	7	7	5	4

results of RMSE obtained and the stability of the datasets as we increase the number of instances removed (considering that an instance is stable if its RMSE variation is not greater than 5% as we change the number of instances removed), the surrounding function would be the more appropriate to consider. However, notice that if we make our definition of stability tighter, and consider an error variation of at most 1%, both the surrounding, remoteness and nonlinearity functions present similar results.

In order to verify how the time complexity of GSGP is affected by the selection process, we analyze the median execution time required by GSGP to create the regression models for each dataset. This analysis for the proximity function is shown in Fig. 6 and considers both training and test phases. Overall, the results agree with our expectations albeit in a less pronounced way than it was expected. The synthetic datasets once again exhibited a contradictory behavior, with execution times essentially constant regardless of the number of instances removed. The time spent in the selection process is relatively small (corresponding, on average, for only 1.8% of the time spent by GSGP to induce its regression models).

B. Results considering Dimensionality Reduction Methods

In this section, we analyze if the application of input space dimensionality reduction methods as the first step in the selection process improved the error values. We used the four methods—Isomap, MDS, PCA, and t-SNE with number of dimensions equals to two—to the training instances of all datasets with dimensionality ≥ 3. The resulting embeddings were used to decide which instances to remove during the selection process. However, GSGP was run with the selected instances with their original number of input attributes, given the results from preliminary experiments showed that the space reduction has a negative impact on its accuracy.

All methods performed similarly and, due to space limitations, we restrict ourselves to the method with the best results (t-SNE, using the proximity function) in terms of RMSE improvement. Table III presents the median RMSE

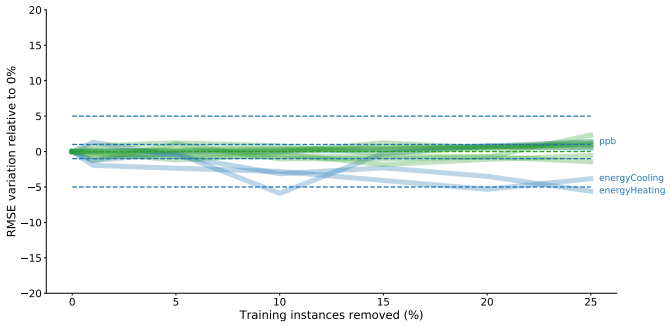

Fig. 7: Evolution of the RMSE values obtained by GSGP combined with *t-SNE*, according to the number of instances removed. Each line corresponds to a dataset, and the outer dashed blue lines represent RMSE variations corresponding to -5% and 5%, delimiting a range where results are stable.

in the test sets, according to 50 executions, and results are visualized in Fig. 7. In order to identify significant differences between the weighting functions, we performed a Friedman test in the same way as presented in the preceding section. The resulting p-value, 5.7×10^{-4}, implies in discarding the null hypothesis (that the performances of the weighting functions are equal) with a confidence level of 95%, leading us to analyse the output of the Nemenyi post-hoc test, presented by the critical difference diagram from Figure 8. The proximity and surrounding functions are significantly better than the nonlinearity function and present no statistical difference to the remoteness function. All the other pairwise comparisons present no statistical significant differences.

After the results of the experiments performed in this section, we recommend the use of instance selection methods with GSGP, using embedding during the instance selection to minimize the impact of the curse of the dimensionality when calculating the distances between two examples. With respect to the functions, we recommend the proximity or surrounding.

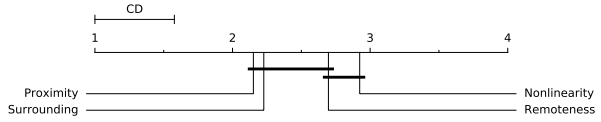

Fig. 8: Critical difference diagrams for the Nemenyi post-hoc tests. For each metric, the mean rank is shown (being rank 1 the best result). Horizontal line segments group together methods with ranks that are not significantly different. The critical difference (CD) for each task is shown in the upper left of each diagram.

VI. Conclusions and Future Work

In this work, we presented an analysis regarding the impact of instances selection methods on the search process performed by GSGP. After a series of experiments with different combinations of metrics and dimensionality reduction methods, we showed that by decreasing the number of training cases and, consequently, the number of dimensions of the semantic space, we can improve the search process performed by GSGP, making that search simpler and more efficient, which, in turns, allows it to induce regression models faster with similar quality.

From all methods and metrics tested, we recommend the use of feature selection during the instance selection process to minimize the impact of the curse of the dimensionality when calculating the distances between two examples. With respect to the weighting functions, we recommend the proximity or surrounding functions.

Potential future works include investigating techniques to identify the noisy instances in order to remove or minimize their importance during the search, and then insert this information into the instance selection process, and analyzing the effect of fitness functions that weight semantic space dimensions, among others.

Acknowledgements

The authors would like to thank FAPEMIG (through the grant no. CEX-PPM-00098-17), MPMG (through the project Analytical Capabilities), CNPq (through the grant no. 310833/2019-1), CAPES, MCTIC/RNP (through the grant no. 51119) and H2020 (through the grant no. 777154) for their partial financial support.

References

[1] J. Le, "The 10 algorithms machine learning engineers need to know," 2016. [Online]. Available: https://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html

[2] T. Mitchell, *Machine Learning*, 1st ed., ser. McGraw-Hill International Editions. McGraw-Hill, 1997.

[3] C. Cardie and N. Howe, "Improving minority class prediction using case-specific feature weights," in *ICML*, 1997, pp. 57–65.

[4] A. Moraglio, K. Krawiec, and C. G. Johnson, "Geometric semantic genetic programming," in *Parallel Problem Solving from Nature, PPSN XII (part 1)*. Springer, 2012, vol. 7491, pp. 21–31.

[5] L. O. V. B. Oliveira, F. E. B. Otero, and G. L. Pappa, "A dispersion operator for geometric semantic genetic programming," in *Proc. of the Genetic and Evolutionary Computation Conference 2016*. ACM, 2016.

[6] L. O. V. B. Oliveira, L. F. Miranda, G. L. Pappa, F. E. B. Otero, and R. H. C. Takahashi, "Reducing dimensionality to improve search in semantic genetic programming," in *Proc. of the 14th International Conference on Parallel Problem Solving from Nature (PPSN XIV)*, ser. LNCS, vol. 9921. Springer International Publishing, 2016, pp. 375–385.

[7] L. Vanneschi, M. Castelli, and S. Silva, "A survey of semantic methods in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 15, no. 2, pp. 195–214, Jun 2014.

[8] A. Moraglio, "Abstract convex evolutionary search," in *Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms*, ser. FOGA '11. New York, NY, USA: ACM, 2011, pp. 151–162.

[9] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133–143, 2010.

[10] L. I. Kuncheva, Á. Arnaiz-González, J.-F. Díez-Pastor, and I. A. D. Gunn, "Instance selection improves geometric mean accuracy: a study on imbalanced data classification," *Progress in Artificial Intelligence*, vol. 8, no. 2, pp. 215–228, Jun 2019.

[11] M. Kordos and M. Blachnik, "Instance selection with neural networks for regression problems," in *Artificial Neural Networks and Machine Learning–ICANN 2012*. Springer, 2012, pp. 263–270.

[12] Á. Arnaiz-González, M. Blachnik, M. Kordos, and C. García-Osorio, "Fusion of instance selection methods in regression tasks," *Information Fusion*, vol. 30, pp. 69–79, 2016.

[13] E. Vladislavleva, G. Smits, and D. den Hertog, "On the importance of data balancing for symbolic regression," *IEEE Trans. Evolutionary Computation*, vol. 14, no. 2, pp. 252–277, 2010.

[14] S. Harmeling, G. Dornhege, D. Tax, F. Meinecke, and K.-R. Müller, "From outliers to prototypes: Ordering data," *Neurocomput.*, vol. 69, no. 13-15, pp. 1608–1618, Aug. 2006.

[15] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psych.*, vol. 24, 1933.

[16] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[17] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, ser. Springer series in statistics. Springer, 1997.

[18] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[19] M. Lichman, "UCI mach. learning repository," 2015. [Online]. Available: http://archive.ics.uci.edu/ml

[20] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, and U.-M. O'Reilly, "Genetic programming needs better benchmarks," in *Proc. of GECCO*, 2012, pp. 791–798.

[21] J. Albinati, G. L. Pappa, F. E. B. Otero, and L. O. V. B. Oliveira, "The effect of distinct geometric semantic crossover operators in regression problems," in *Proc. of EuroGP*, 2015, pp. 3–15.

[22] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, Oct 2017.

[23] M. Keijzer, "Improving symbolic regression with interval arithmetic and linear scaling," in *6th European Conference, EuroGP 2003*, vol. 2610. Springer Berlin Heidelberg, 2003, pp. 70–82.

[24] E. J. Vladislavleva, G. F. Smits, and D. den Hertog, "Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 333–349, April 2009.

[25] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992, vol. 1.

[26] J. Ni, R. H. Drieberg, and P. I. Rockett, "The use of an analytic quotient operator in genetic programming," *Evolut. Computation, IEEE Trans. on*, vol. 17, no. 1, pp. 146–152, 2013.