Reducing Dimensionality to Improve Search in Semantic Genetic Programming

Luiz Otavio V. B. Oliveira[1], Luis F. Miranda[1], Gisele L. Pappa[1],
Fernando E. B. Otero[2], and Ricardo H. C. Takahashi[3]

[1] Computer Science Dep., Universidade Federal de Minas Gerais, Belo Horizonte, Brazil, {luizvbo,luisfmiranda,glpappa}@dcc.ufmg.br,
[2] School of Computing, University of Kent, Chatham Maritime, UK,
F.E.B.Otero@kent.ac.uk,
[3] Mathematics Dep., Universidade Federal de Minas Gerais, Belo Horizonte, Brazil,
taka@mat.ufmg.br

Abstract. Genetic programming approaches are moving from analysing the syntax of individual solutions to look into their semantics. One of the common definitions of the semantic space in the context of symbolic regression is a n-dimensional space, where n corresponds to the number of training examples. In problems where this number is high, the search process can became harder as the number of dimensions increase. Geometric semantic genetic programming (GSGP) explores the semantic space by performing geometric semantic operations—the fitness landscape seen by GSGP is guaranteed to be conic by construction. Intuitively, a lower number of dimensions can make search more feasible in this scenario, decreasing the chances of data overfitting and reducing the number of evaluations required to find a suitable solution. This paper proposes two approaches for dimensionality reduction in GSGP: (i) to apply current instance selection methods as a pre-process step before training points are given to GSGP; (ii) to incorporate instance selection to the evolution of GSGP. Experiments in 15 datasets show that GSGP performance is improved by using instance reduction during the evolution.

Keywords: dimensionality reduction, semantic genetic programming, instance selection

1 Introduction

Evolutionary computation methods have recently turned their attention to the semantics of the solutions represented by individuals instead of focusing only on their syntax [17]. Particularly, in the case of genetic programming, many methods are switching from the syntactic space to work on a n-dimensional semantic space, where n is the number of training instances we learn the function from. When applying any function—e.g., an individual—to the training set, the produced output corresponds to a point in the semantic space.

Given the definition above, the number of dimensions of the semantic space equals the number of training examples. In problems where this number is high—a common scenario in real-world applications—the search process can became harder as the number of dimensions increase, a problem well-known as the *curse of dimensionality* [5]. As the number of dimensions of the problem increases, the volume of the search space also increases exponentially.

One of the simplest ways to deal with the curse of dimensionality is to reduce the number of dimensions of the search space [5]. As in geometric semantic genetic programming each space dimension corresponds to a training instance, an alternative is to perform what in the machine learning literature is known as data instance selection. Data instance selection is a well-known problem within the context of data classification, but there is not extensive work regarding regression problems [2]. Instance selection methods are strongly based on distances between training instances from both the set of input and output features.

This paper evaluates the impact of reducing the number of dimensions of the semantic space in the context of geometric semantic genetic programming. This scenario is interesting since the crossover and mutation operators guarantee the semantic fitness landscape explored by GP is conic, which can be optimized by evolutionary algorithms with good results for virtually any metric, as indicated by [13]. Intuitively, a lower number of dimensions can make search more feasible, reducing the number of evaluations required to find a suitable solution while decreasing the chances of data overfitting.

Looking at how current instance selection methods work, we take advantage of previous knowledge and propose two approaches for instance selection: (i) apply current instance selection methods as a pre-process step before training points are given to GSGP; (ii) incorporate instance selection to the evolution of GSGP. In the first case, we use the methods Threshold Condensed Nearest Neighbor (TCNN) and Threshold Edited Nearest Neighbor (TENN) [9] to select instances, which are then given to GSGP. The second approach incorporates instance selection to the evolution of GSGP, through the proposed Probabilistic instance Selection based on the Error (PSE) method.

Computational experiments in 15 real-world and synthetic datasets, where the number of training instances varies from 50 to 4000 and instance number reduction (i.e. search space dimension reduction) of up to 68.50%, show that results obtained by TCNN and TENN are no better than those generated by a random selection scheme. PSE, in turn, shows results statistically significantly better than GSGP with all instances in 5, and no statistical difference in 7 cases.

2 Related Work

Instance selection methods are commonly used in the classification literature [6], and play different roles in noisy and noise-free application scenarios. In noise-free scenarios, the idea is to remove points from the training set without degrading accuracy, such as improving storage and search time. In noisy application domains, the main idea is to remove outliers. In classification, these methods rely on the class labels of neighbour instances to determine the rejection/acceptance of an instance to the selected set. However, there are not many methods for instance selection in regression problems. A few works have extended well-known instance selection methods for classification to the context of regression [2].

The authors in [7] introduced a method based on mutual information, inspired by feature selection methods that rely on this criterion. The method focuses on noise-free scenarios, and has as its main objective to choose the best subset of instances to build a model. In this same direction, the authors in [16] propose Class Conditional Instance Selection for Regression (CCISR). It extends the Class Conditional Instance Selection method for classification, which uses a class conditional nearest neighbour relation to guide the search process. The authors in [9] proposed the Threshold Condensed Nearest Neighbor (TCNN) and Threshold Edited Nearest Neighbor (TENN) algorithms—regression versions of the ENN and CNN methods for classification, respectively. These algorithms will be discussed in the next section, as they are used in this paper.

Recently, the authors in [2] compared different strategies for instance selection in regression: discretization techniques—which transform the continuous outputs of the problem into discrete variables and then apply the traditional version of instance selection methods for classification—TCNN and TENN. They also proposed an ensemble method, namely bagging, to combine several instance selection algorithms. Each algorithm within the ensemble returns an array of binary votes (0 means the instance is not selected and 1 otherwise), and the relevance of an instance in the training set is considered proportional to the number of accumulated votes. The final instance selection is given by a threshold, which defines the percentage of votes an instance must have to be selected. As expected, the ensemble method presented the best results overall.

In our context, the use of an ensemble is not justifiable, as it is a time consuming task and would add too much time overhead to the search. For this reason, we adopted the threshold versions of TCNN and TENN, as the first assumes noise-free scenarios and the second focuses on outliers.

3 Strategies for Semantic Space Dimensionality Reduction

This section introduces two strategies to reduce the dimensionality of the semantic search space. First, we formally introduce the problem and motivation for instance selection in this scenario. Given a finite set of input-output pairs representing the training cases, defined as $T = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$—where $(\mathbf{x_i}, y_i) \in \mathbb{R}^d \times \mathbb{R}$ $(i = 1, 2, \ldots, n)$—symbolic regression consists in inducing a model $p : \mathbb{R}^d \to \mathbb{R}$ that maps inputs to outputs, such that $\forall (\mathbf{x_i}, y_i) \in T : p(\mathbf{x_i}) = y_i$.

Let $I = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ and $O = [y_1, y_2, \ldots, y_n]$ be the input set and output vector, respectively, associated to the training instances. The semantics of a program p represented by an individual evolved by GSGP, denoted by $s(p)$, is the vector of outputs it produces when applied to the set of inputs I, i.e., $s(p) = p(I) = [p(\mathbf{x_1}), p(\mathbf{x_2}), \ldots, p(\mathbf{x_n})]$. The semantics of any program can be represented as a point in a n-dimensional topological space $\mathcal{S}$, called semantic space, where n is the size of the training set.

GSGP introduces geometric semantic operators for GP that act on the syntax of the programs, inducing a geometric behaviour on the semantic level [14]. These operators guarantee the semantic fitness landscape explored by GP is conic, a

```
Algorithm 1: TENN
Input: $T = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$, $k$, $\alpha$
Output: Instance set $P \subset T$
Shuffle $T$;
$P \leftarrow T$;
for $i \leftarrow 1$ to $n$ do
    $\hat{y} \leftarrow regression(\mathbf{x}_i, P \setminus (\mathbf{x}_i, y_i))$;
    $N \leftarrow knn(k, T)$;
    $\theta \leftarrow \alpha \cdot sd(N)$;
    if $\theta = 0$ then
        $\theta \leftarrow \alpha$
    if $|y_i - \hat{y}| > \theta$ then
        $P \leftarrow P \setminus (\mathbf{x}_i, y_i)$
return $P$;
```

```
Algorithm 2: TCNN
Input: $T = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$, $k$, $\alpha$
Output: Instance set $P \subset T$
Shuffle $T$;
$P \leftarrow (\mathbf{x}_1, y_1)$;
for $i \leftarrow 2$ to $n$ do
    $\hat{y} \leftarrow regression(\mathbf{x}_i, P)$;
    $N \leftarrow knn(k, T)$;
    $\theta \leftarrow \alpha \cdot sd(N)$;
    if $\theta = 0$ then
        $\theta \leftarrow \alpha$
    if $|y_i - \hat{y}| > \theta$ then
        $P \leftarrow P \cup (\mathbf{x}_i, y_i)$
return $P$;
```

property with positive effects on the search process. There is formal evidence that indicates evolutionary algorithms with geometric operators can optimise cone landscapes with good results for virtually any metric [13].

As the semantics in GSGP is defined as a point with a number of dimensions equivalent to the number of instances given as input to a candidate regression function, by reducing the number of input instances we automatically reduce the number of dimensions of the semantic space, which in turn reduces the complexity of the search space. Intuitively, the smaller the complexity the smaller the number of possible convex combinations, which may help the speed of convergence to the optimum. In this context, the first strategy we propose to reduce the number of dimensions of the search space is executed before data is given as input to GSGP, and depends only on the characteristics of the dataset. The second strategy, in turn, takes into account the median absolute error of an instance during GSGP evolution to select the most appropriate instances.

3.1 Pre-Processing Strategies

We first introduce two methods for instance selection in regression. The Threshold Edited Nearest Neighbor (TENN) and Threshold Condensed Nearest Neighbor (TCNN) [9] adapt instance selection algorithms for classification problems—ENN [18] and CNN [8]—to the regression domain. They are presented in Algorithms 1 and 2.

These algorithms employ an internal regression method to evaluate the instances according to the similarity-based error. The decision of keeping or removing the i-th instance from the training set is based on the deviation of the instance prediction $\hat{y}_i$ and the expected output y_i, given by $|\hat{y}_i - y_i|$. If this difference is smaller than a threshold θ, $\hat{y}_i$ and y_i are considered similar and the instance is accepted or rejected, depending on the algorithm. The threshold θ is computed based on the local properties of the dataset, given by $\alpha \cdot sd(N)$, where α is a parameter controlling the sensitivity and $sd(N)$ returns the standard de-

Algorithm 3: PSE method

```
Input: Training set ($T$), population ($pop$), lower bound ($\lambda$)
Output: Instance set $P \subset T$
foreach $inst = (\mathbf{x}_i, y_i) \in T$ do        // Compute the median absolute error
    $E \leftarrow [|p_1(\mathbf{x}_i) - y_i)|, |p_2(\mathbf{x}_i) - y_i)|, \ldots, |p_m(\mathbf{x}_i) - y_i)|]$;
    $inst.med \leftarrow median(E)$;
Sort $T$ by $med$ value in descending order;
$P \leftarrow \{\}$;
for $i \leftarrow 1$ to $|T|$ do
    $inst \leftarrow (\mathbf{x}_i, y_i) \in T$;
    $\tilde{r} \leftarrow \frac{(i-1)}{|T|-1}$ ;                        // Compute the normalized rank
    $prob_{sel} \leftarrow 1 - (1 - \lambda) \cdot \tilde{r}^2$ ;              // Probability of selecting $inst$
    if $prob_{sel} \geq rand()$ then       // Add $inst$ to $P$ with probability $prob_{sel}$
        $P \leftarrow P \cup \{inst\}$;
return $P$;
```

viation of the outputs of the set N, composed by the k nearest neighbours of the instance.

The internal regression method adopted by TCNN and TENN—the procedure *regression* presented in Algorithms 1 and 2—can be replaced by any regression method. Our implementation uses the version of the kNN (k-nearest neighbour) algorithm for regression to infer the value of $\hat{y}$. Besides the training set T, these algorithms receive as input the number of neighbours to be considered and a parameter α, which controls how the threshold is calculated. At the end, the set P of instances selected to be used to train the external regression method is returned.

TENN is a decremental method, starting with all training cases in the set P and iteratively removing the instances diverging from their neighbours. An instance $(\mathbf{x}_i, y_i)$ is considered divergent if the output $\hat{y}$ inferred by the model learned without the instance is dissimilar from its output (y_i). TCNN, on the other hand, is an incremental method, beginning with only one instance from the training set in P and iteratively adding only those instances that can improve the search. The instance $(\mathbf{x}_i, y_i)$ is added only if the output $\hat{y}$ inferred by the model learned with P diverges from y_i.

3.2 GSGP Integrated Strategies

Both TENN and TCNN disregard any information about the external regression algorithm, since they are used in a pre-processing phase. In order to overcome this limitation, we propose a method to select instances based on their median absolute error, considering the output of the programs in the current population. The method, called Probabilistic instance Selection based on the Error (PSE), probabilistically selects a subset of the training set at each ρ generations, as presented in Algorithm 3. The higher the median absolute error, the higher the probability of an instance being selected to compose the training subset used by GSGP. The rationale behind this approach is to give higher probability to instances which are, in theory, more difficult to be predicted by the current population evolved by GSGP.

Given a GSGP population $P = \{p_1, p_2, \ldots, p_m\}$, the median absolute error of the i-th instance $(\mathbf{x}_i, y_i) \in T$ is given by the median value of the set $E = \{|p_1(\mathbf{x}_i) - y_i|, |p_2(\mathbf{x}_i) - y_i|, \ldots, |p_m(\mathbf{x}_i) - y_i|\}$. These values are used to sort T in descending order, and the position of the instance in T is used to calculate its probability of being selected to be part of the training set.

In order to compute this probability, the method normalizes the rank of the instance in T to the range $[0, 1]$ by

$$\tilde{r} = \frac{(i-1)}{|T|-1} \ , \tag{1}$$

where i is the position of the instance in the ordered set T, $|.|$ denotes the cardinality of the set and $\tilde{r} \in [0, 1]$ is the normalized rank. The value of $\tilde{r}$ is used to calculate the probability of selecting the instance, given by

$$prob_{sel} = 1 - (1 - \lambda) \cdot \tilde{r}^2 \ , \tag{2}$$

where λ is a parameter that determines the lower bound of the probability function. The higher the value of λ, the more instances are selected. The area under the function, equivalent to $\frac{2+\lambda}{3}$, corresponds to the proportion of instances selected from T.

4 Experimental Results

This section presents an experimental analysis of the instance selection strategies. The results obtained by GSGP with instance selection performed by TCNN and TENN (Section 4.1), and PSE (Section 4.2) are compared with GSGP with all instances.

The experiments were performed in a collection of datasets selected from the UCI machine learning repository [11], GP benchmarks [12] and a GSGP study from the literature [1], as presented in Table 1. For real-world datasets, we performed 5-fold cross-validations with 10 replications, and for synthetic ones, the data was sampled five times—according to Table 3 from [12]—and the algorithms were applied 10 times, both cases resulting in 50 executions. This sampling strategy justifies the adoption of the t-test in the statistical analysis performed in this section—the number of replications is larger than 30 [4] For compatibility purposes, we removed the categorical attributes of the datasets.

All executions used a population of 1,000 individuals evolved for 2,000 generations with tournament selection of size 10. The grow method [10] was adopted to generate the random functions inside the geometric semantic operators, and the ramped half-and-half method [10] used to generate the initial population, both with maximum individual depth equals to 6. The terminal set included the variables of the problem and constant values randomly picked from the interval $[-1, 1]$. The function set included three binary arithmetic operators $(+, -, \times)$ and the analytic quotient (AQ) [15] as an alternative to the arithmetic division. The GSGP method employed the crossover for Manhattan-based fitness function and mutation operators from [3] with probabilities 0.9 and 0.1, respectively.

Table 1: Datasets used in the experiments.

Dataset	**Size**	**Nature**	**Source**	**Dataset**	**Size**	**Nature**	**Source**
airfoil	1503	Real	[1, 11]	keijzer-7	100	Synthetic	[12]
bioavailability	359	Real	[1]	ppb	131	Real	[1]
concrete	1030	Real	[1, 11]	towerData	4999	Real	[1]
cpu	209	Real	[1, 11]	vladislavleva-1	100	Synthetic	[1, 12]
energyCooling	768	Real	[1, 11]	wineRed	1599	Real	[1, 11]
energyHeating	768	Real	[1, 11]	wineWhite	4898	Real	[1, 11]
forestfires	517	Real	[1, 11]	yacht	308	Real	[1, 11]
keijzer-6	50	Synthetic	[1, 12]				

Table 2: Median training and test RMSE and reduction (*% red.*) achieved by the algorithms for each dataset. Values highlighted in bold corresponds to test RMSE statically worst than GSGP, according to a t-test with 95% confidence.

	GSGP		**GSGP-TCNN**			**GSGP-TENN**			**GSGP-Rnd**	
Dataset	**tr**	**ts**	**tr**	**ts**	**% red.**	**tr**	**ts**	**% red.**	**tr**	**ts**
airfoil	7.89	8.42	7.76	**8.74**	38.60	8.06	**8.60**	1.90	7.65	8.38
bioavailability	9.89	30.74	4.95	**36.29**	46.30	9.84	31.38	0.90	4.55	**34.39**
concrete	3.65	5.39	2.80	**6.40**	38.20	3.65	5.21	3.20	3.18	**5.95**
cpu	6.13	30.92	5.46	**33.61**	11.20	5.06	**51.54**	65.40	5.67	32.28
energyCooling	1.26	1.51	1.28	**2.49**	14.70	1.28	**1.83**	36.60	1.19	**1.71**
energyHeating	0.80	0.96	0.83	**1.87**	11.10	0.67	**1.84**	45.40	0.77	**1.11**
forestfires	30.74	51.63	13.68	**101.90**	42.80	30.75	51.94	5.80	22.49	**57.57**
keijzer-6	0.01	0.40	0.01	0.36	10.60	0.00	**1.25**	53.00	0.01	0.32
keijzer-7	0.02	0.02	0.02	0.02	5.30	0.01	**0.40**	68.50	0.01	**0.05**
ppb	0.92	28.74	0.20	32.08	41.50	0.91	28.04	3.80	0.25	30.50
towerData	20.44	21.92	19.82	**22.71**	12.60	20.44	**43.86**	41.90	20.40	**22.06**
vladislavleva-1	0.01	0.04	0.01	**0.07**	20.90	0.01	**0.07**	43.40	0.01	**0.06**
wineRed	0.49	0.62	0.40	**0.73**	51.10	0.49	0.62	0.10	0.41	**0.66**
wineWhite	0.64	0.70	0.66	**0.78**	52.30	0.64	0.69	0.10	0.60	**0.71**
yacht	2.12	2.52	2.20	**5.19**	36.90	2.11	**2.83**	24.30	2.01	**2.63**

The mutation step required by the mutation operator was defined as 10% of the standard deviation of the outputs (O) given by the training data.All instances in the training set were used as input for the instance selection methods and GSGP.

4.1 Comparing Instance Selection Methods

In this section we compare the results obtained by GSGP with and without the instance selection performed before the evolutionary stage (pre-processing). The selection was performed by TCNN (GSGP-TCNN) and TENN (GSGP-TENN) methods, with $k = 9$ and 10 different values for α equally distributed in the intervals $[0.1, 1]$ and $[5.5, 10]$, respectively. Table 2 presents the median training and test RMSE's and the data reduction obtained with α resulting in the largest data reduction by TCNN and TENN methods—1 and 5.5, respectively.

In order to investigate the significance of instance selection methods in GSGP, we randomly selected l instances from each dataset, with no replacement, to compose a new training set used as input by GSGP. The value of l is defined as the smaller of the sizes of the sets resulting from TENN and TCNN. Table 2 presents the median training and test RMSE's of these experiments in the last two columns (denoted as 'GSGP-Rnd'). The results obtained show that using

Table 3: Median training RMSE of the GSGP-PSE with different values of λ and ρ for the test bed. The smallest RMSE for each dataset is presented in bold.

	$\rho = 5$			$\rho = 10$			$\rho = 15$		
Dataset	$\lambda = 0.1$	$\lambda = 0.4$	$\lambda = 0.7$	$\lambda = 0.1$	$\lambda = 0.4$	$\lambda = 0.7$	$\lambda = 0.1$	$\lambda = 0.4$	$\lambda = 0.7$
airfoil	8.03	8.15	8.11	**7.97**	8.05	8.16	8.12	8.05	8.11
bioavailability	9.66	9.66	9.88	9.70	9.69	9.83	**9.53**	9.77	9.81
concrete	3.35	3.49	3.56	3.35	3.45	3.58	**3.34**	3.45	3.56
cpu	**4.93**	5.46	5.70	5.02	5.33	5.89	5.01	5.39	5.88
energyCooling	1.13	1.19	1.23	1.12	1.18	1.22	**1.12**	1.17	1.23
energyHeating	**0.66**	0.72	0.77	0.67	0.72	0.76	0.67	0.71	0.76
forestfires	25.94	27.67	29.21	25.72	27.61	29.43	**25.55**	27.87	29.58
keijzer-6	0.01	0.01	0.01	0.01	**0.01**	0.01	0.01	0.01	0.01
keijzer-7	0.02	0.02	0.02	0.02	**0.02**	0.02	0.02	0.02	0.02
ppb	**0.50**	0.65	0.81	0.53	0.65	0.80	0.52	0.63	0.76
towerData	19.22	19.74	19.98	19.22	19.61	20.09	**19.18**	19.61	19.92
vladislavleva-1	0.01	0.01	0.01	0.01	0.01	0.01	**0.01**	0.01	0.01
wineRed	0.47	0.48	0.49	0.47	0.48	0.49	**0.47**	0.48	0.49
wineWhite	**0.63**	0.64	0.64	0.63	0.64	0.64	0.63	0.64	0.64
yacht	1.94	2.02	2.09	**1.94**	2.01	2.08	1.94	2.00	2.09

TCNN and TENN do not make any systematic improvement on GSGP results. Moreover, the results obtained by them are no better than those generated by a random selection scheme. Hence, the strategies used by these methods do not seem appropriate for the scenario we have.

4.2 Evaluating the Effects of PSE

In this section, we first investigate the sensitivity of PSE parameters and then compare the performance of GSGP with and without the PSE method. PSE parameters ρ and λ have a direct impact on the number of instances selected and how they are selected. In order to analyse their impact on the search, we fixed the GSGP parameters and focus on looking at the results as we varied these parameters. The values of ρ were set to 5, 10 and 15 while we varied the value of λ in 0.1, 0.4 and 0.7. Table 3 presents the median training RMSE obtained by the GSGP with these PSE configurations. The results show that higher values of ρ (15) with lower values of λ (0.1) tend to reduce the training RMSE.

The experiments with PSE adopt the values of ρ and λ resulting in the smallest median training RMSE, as presented in Table 3. Table 4 presents the median training and test RMSE's obtained by GSGP and by GSGP with PSE (GSGP-PSE). In order to identify statistically significant differences, we performed t-tests with 95 % confidence level, regarding the test RMSE of both methods in 50 executions. The symbol in the last column indicates datasets where the results present significant difference. Overall, GSGP-PSE performs better in terms of test RMSE than GSGP, being better in five datasets and worse in three.

Figure 1 compares the evolution of the fitness of the best individual along the generations in the training and test sets for GSGP and GSGP-PSE, for two different datasets. Note that GSGP errors are overall higher than PSE. For instance, looking at the convergence of the dataset *towerData*, if we stop the evolution at generation 1,000, GSGP would have a test error of 25.02 and GSGP-PSE of 23.64. GSGP needs 293 more generations to reach that same error.

Table 4: Median training and test RMSE's obtained for each dataset. The symbol ▲(▼) indicates GSGP-PSE is statistically better (worse) than GSGP in the test set according to a t-test with 95% confidence.

	GSGP		GSGP-PSE	
Dataset	**tr**	**ts**	**tr**	**ts**
airfoil	7.88	8.42	7.97	8.55
bioavailability	9.89	30.74	9.53	32.16 ▼
concrete	3.65	5.39	3.34	5.24 ▲
cpu	6.13	30.92	4.93	33.44
energyCooling	1.26	1.51	1.12	1.38 ▲
energyHeating	0.80	0.96	0.66	0.84 ▲
forestfires	30.74	51.63	25.55	51.32
keijzer-6	0.01	0.40	0.01	0.32
keijzer-7	0.02	0.02	0.02	0.02
ppb	0.92	28.74	0.50	28.96
towerData	20.44	21.92	19.18	20.95 ▲
vladislavleva-1	0.01	0.04	0.01	0.05
wineRed	0.49	0.62	0.47	0.62 ▼
wineWhite	0.64	0.70	0.63	0.69 ▼
yacht	2.12	2.52	1.94	2.47 ▲

5 Conclusions and Future Work

This paper presented a study about the impact of instance selection methods on GSGP search. Two approaches were adopted: (i) selecting the instances in a pre-processing step; and (ii) selecting instances during the evolutionary process, taking into account the impact of the instance on the search.

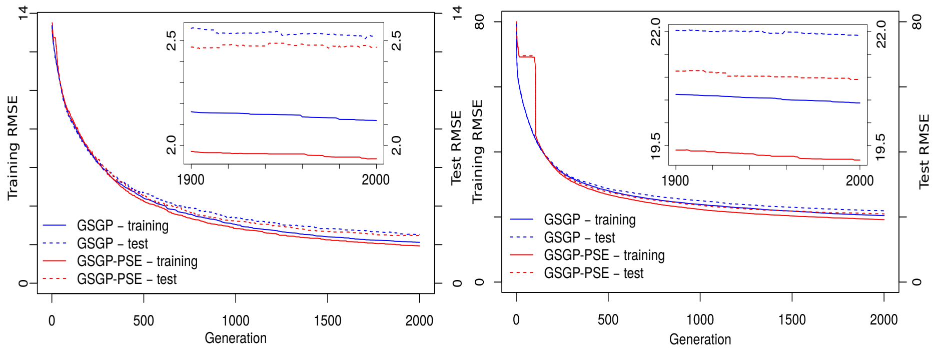
Experiments were performed in a collection of 15 datasets in order to evaluate the impact of the instance selection. The first analysis showed GSGP fed with the whole dataset performs better in terms of test RMSE than when using subsets selected with TENN, TCNN or randomly. The second analysis showed that overall GSGP with PSE performs better in terms of test RMSE than the GSGP alone, and that instance selection to reduce the semantic space is worth further investigation.

Potential future works include analysing the effect of fitness functions that weight semantic space dimensions, exploring the impact of noise in the PSE method and studying approaches to insert information about the noisy instances during the selection.

Acknowledgements. The authors would like to thank CNPq (141985/2015-1), CAPES and Fapemig for their financial support.

References

1. Albinati, J., Pappa, G.L., Otero, F.E.B., Oliveira, L.O.V.B.: The effect of distinct geometric semantic crossover operators in regression problems. In: Proc. of EuroGP. pp. 3–15 (2015)
2. Arnaiz-González, Á., Blachnik, M., Kordos, M., García-Osorio, C.: Fusion of instance selection methods in regression tasks. Information Fusion 30, 69–79 (2016)
3. Castelli, M., Silva, S., Vanneschi, L.: A C++ framework for geometric semantic genetic programming. Genetic Prog. and Evolvable Machines 16(1), 73–81 (2015)



(a) *yacht* dataset.

(b) *towerData* dataset.

Fig. 1: Median RMSE in the training and test sets over the generations for GSGP with and without PSE for *yacth* and *towerData* datasets.

4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 7, 1–30 (2006)
5. Domingos, P.: A few useful things to know about machine learning. Commun. ACM 55(10), 78–87 (Oct 2012)
6. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(3), 417–435 (2012)
7. Guillen, A., Herrera, L.J., Rubio, G., Pomares, H., Lendasse, A., Rojas, I.: New method for instance or prototype selection using mutual information in time series prediction. Neurocomput. 73(10-12), 2030–2038 (2010)
8. Hart, P.: The condensed nearest neighbor rule (corresp.). Information Theory, IEEE Transactions on 14(3), 515–516 (1968)
9. Kordos, M., Blachnik, M.: Instance selection with neural networks for regression problems. In: Villa, A.E.P., et al. (eds.) Proc. of the ICANN'12, part II. pp. 263–270. Springer Berlin Heidelberg (2012)
10. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT Press (1992)
11. Lichman, M.: UCI mach. learning repository (2015), http://archive.ics.uci.edu/ml
12. McDermott, J., White, D.R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., O'Reilly, U.M.: Genetic programming needs better benchmarks. In: Proc. of GECCO. pp. 791–798 (2012)
13. Moraglio, A.: Abstract convex evolutionary search. In: Proc. of the 11th FOGA. pp. 151–162 (2011)
14. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Proc. of PPSN XII, vol. 7491, pp. 21–31. Springer (2012)
15. Ni, J., Drieberg, R.H., Rockett, P.I.: The use of an analytic quotient operator in genetic programming. Evolut. Computation, IEEE Trans. on 17(1), 146–152 (2013)
16. Rodrguez-Fdez, I., Mucientes, M., Bugarn, A.: An instance selection algorithm for regression and its application in variance reduction. In: Fuzzy Systems (FUZZ), 2013 IEEE International Conference on. pp. 1–8 (July 2013)
17. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Prog. and Evolvable Machines 15(2), 195–214 (2014)
18. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. Systems, Man and Cybernetics, IEEE Trans. on 2(3), 408–421 (1972)