Filtering Outliers in One Step with Genetic Programming

Uriel López[1,2], Leonardo Trujillo[1,2(✉)], and Pierrick Legrand[3,4,5]

[1] Tecnológico Nacional de México/I.T. Tijuana, Tijuana, BC, Mexico
{uriel.lopez,leonardo.trujillo}@tectijuana.edu.mx
[2] BioISI, Faculty of Sciences, University of Lisbon, Lisbon, Portugal
[3] IMB, UMR CNRS 5251, 351 cours de la libération, Talence, France
[4] Inria Bordeaux Sud-Ouest, Talence, France
[5] University of Bordeaux, Bordeaux, France
pierrick.legrand@u-bordeaux.fr

Abstract. Outliers are one of the most difficult issues when dealing with real-world modeling tasks. Even a small percentage of outliers can impede a learning algorithm's ability to fit a dataset. While robust regression algorithms exist, they fail when a dataset is corrupted by more than 50% of outliers (breakdown point). In the case of Genetic Programming, robust regression has not been properly studied. In this paper we present a method that works as a filter, removing outliers from the target variable (vertical outliers). The algorithm is simple, it uses a randomly generated population of GP trees to determine which target values should be labeled as outliers. The method is highly efficient. Results show that it can return a clean dataset when contamination reaches as high as 90%, and may be able to handle higher levels of contamination. In this study only synthetic univariate benchmarks are used to evaluate the approach, but it must be stressed that no other approaches can deal with such high levels of outlier contamination while requiring such small computational effort.

Keywords: Outliers · Robust regression · Genetic programming

1 Introduction

The main application domain for Genetic Programming (GP) continues to be symbolic regression. The ability of GP to model difficult non-linear problems, and to produce relatively compact models when appropriate bloat control is used [1], makes it a good option in this common machine learning task. Unlike random ensemble regression, SVM regression or Neural Networks, for example, GP has the potential of delivering human-readable solutions that are also accurate and efficient. However, like any data-driven approach to modeling, much of the quality of the final solution will be determined by the nature of the training data; i.e.; even the best algorithm cannot produce a model when the output

A. Auger et al. (Eds.): PPSN 2018, LNCS 11101, pp. 209–222, 2018.
https://doi.org/10.1007/978-3-319-99253-2_17

variable shows no relationship to the input variables. One particularly difficult case in learning is when the training data is corrupted by *outliers*. Indeed, outlier data points can severely skew the learning process and bias the search towards unwanted regions in solution space.

A common way to deal with this situation is to apply a filtering process or to use a robust objective measure, such that performance estimation is not affected by the presence of outlier points [2]. Such methods can handle outliers effectively under the assumption that they are relatively rare; i.e. outliers represent a minority of the data points in the dataset. However, this work considers an extreme case; where the percentage of outliers far outnumbers the inliers, reaching as high as 90% of the entire training set[1]. In this case, even the most robust objective function cannot properly guide a learning algorithm, and it is the same for GP.

One way to deal with outliers is to use specialized sampling techniques such as Random Sampling Consensus (RANSAC) [3], which is often used in computer vision systems for data calibration [4]. Indeed, this method has been successfully combined with GP to derive accurate models even when the data contamination is above the breakdown point; i.e., the number of outliers is above 50%. However, a noteworthy drawback of such a method is that its computational cost is extremely high, with the number of expected samples required to build a sufficiently accurate model increasing exponentially with the total contamination in the dataset. However, and this cannot be stressed enough, in current literature no other methods exist that can deal with such extreme cases of data contamination in an automatic manner [5–7]. In many cases, the best suggestion is to simply inspect the data visually and clean it manually. In the case of GP, for instance, such conditions are never even tested, not even for robust GP systems [8–10].

The present work fills this notable gap by presenting an approach to clean highly contaminated datasets in regression tasks. Particularly, this work considers the case where the output variable is highly contaminated by outliers, measurements that deviate sharply from the true signal that is trying to be modeled. The proposed method can be used as a preprocessing step to clean the data, applied before the actual modeling process is performed. The method is efficient, requiring the same amount of computational effort that is required to evaluate a single GP population. It uses a single randomly generated population to determine which data points are outliers and which are not. Besides this effort, all that is required is to order the population based on fitness, and with that an efficient criterion for cleaning the data is proposed. The method can be integrated not just with GP, but with any regression method. If used with GP, however, then the outlier removal process is obtained basically for free since the initial population of the GP search could be used to perform the filtering at generation 0.

[1] In fact, while the reported experiments only consider up to this level of data contamination, it is straightforward to extend our approach to more severe scenarios.

The proposed method operates under the assumption that outlier points are more difficult to model than inliers, when the models (GP trees) are generated randomly. While we do not derive any formal proofs that show that this assumption will hold in general, the experimental work confirms that this assumption is valid for the set of test cases used to evaluate the proposal. We test the algorithm on datasets that are contaminated by as much as 90% of outliers, and are able to remove a sufficiently large proportion of the outlier instances, it then becomes feasible for a standard robust regression method to tackle the problem.

The remainder of this paper proceeds as follows. Section 2 reviews basic concepts on robust regression and outlier detection. Then, Sect. 3 presents our outlier filter, the reader will notice that the most important characteristic of the proposal is its simplicity; the section also reviews related works. Section 4 presents the experimental work, following the general methodology of [2]. Finally, Sect. 5 provides a discussion, outlines our main conclusions and describes future work.

2 Background

2.1 Outliers

All regression and automatic modeling systems are heavily influenced by the presence of anomalies in the training data [7]. These anomalies are usually referred to as outliers, and can be present in the input variables, the output variable, or in both. Outliers can be generated by several causes, such as gross human error, equipment malfunction, extremely severe random noise or missing data [5–7]. When outliers are rare, then it is possible to define them as data points that are very different from the rest of the observations included in the dataset. However, such a definition is not useful when the number of outliers exceeds the number of inliers (non-outlier data points). Moreover, it is important to distinguish between outliers and just signal noise. In our opinion, the two most important distinctions are: (1) noise can be effectively modelled, and thus filtered; and (2) outlier points deviate from inliers at a large scale, i.e. outliers are anomalous w.r.t. the inliers, which is not a formal definition but in practice it is a useful one. Therefore, since in this work we are concerned with the presence of outliers in the output or target variable in regression problems (also called vertical outliers), we can use the following definition for outliers [2]:

Definition 1. *An outlier is a measurement of a system that is anomalous with respect to the true behavior of the system.*

While this definition may be seen as a tautology, there is an aspect of it that is not immediately obvious. Notice that we are defining an outlier relative to the "true behavior of the system", whether this behavior is observable or not. The definition is not based on the observed behavior in a representative dataset from which we can pose a regression or learning task. This is crucial, because it may seem counter intuitive to have a dataset where the majority of samples are in fact outliers. However, if we know the true behavior of a system, in a controlled

experiment with synthetic problems, it is straightforward to build a dataset where the majority of points are outliers based on Definition 1. Moreover, such a scenario is often encountered in real-world problems as well, one well-known domain is computer vision [4].

2.2 Robust Regression

First, let us define the standard regression problem. Given a training dataset $\mathbb{T} = \{(\mathbf{x}_i, y_i); i = 1, ..., n\}$, the goal is to derive a model that predicts y_i based on $\mathbf{x}_i$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. In GP literature we can refer to each input/output pair $(\mathbf{x}_i, y_i)$ as a fitness case, a training instance or a data point. For linear regression, the model is expressed as

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i \quad i = 1, .., n \tag{1}$$

where the model parameters $\beta = (\beta_0, \beta_1, ..., \beta_p) \in \mathbb{R}^{p+1}$, can be estimated by $\widehat{\beta}_0, ..., \widehat{\beta}_p$ using the least squares method [11], which can be expressed as

$$(\widehat{\beta}_0, ..., \widehat{\beta}_p) \leftarrow \underset{\beta \in \mathbb{R}^{p+1}}{arg\ min} \sum_{i=1}^{n} r_i^2 \ , \tag{2}$$

to find the best fit parameters of the linear model, where r_i denotes the residuals $r_i(\widehat{\beta}_0, ..., \widehat{\beta}_p) = y_i - (\widehat{\beta}_0 + \widehat{\beta}_1 x_{i1} + \cdots + \widehat{\beta}_p x_{ip})$ and the errors ε_i have an expected value of zero [12]; if the summation in Eq. 2 is divided by n, the error measure that must be minimized is the mean squared error (MSE). The issue with outliers is that they bias the standard objective measures defined above (and others, such as regularized approaches). In classical regression, there are several robust regression methods that deal with the presence of outliers by modifying the objective function used to perform the regression. For instance, Least Median Squares (LMS) [13]

$$(\widehat{\beta}_0, ..., \widehat{\beta}_p) \leftarrow \underset{\beta_0, ..., \beta_p}{arg\ min}\ med\ \{r_1^2, ..., r_n^2\} \tag{3}$$

where *med* represents the median. Another approach is Least Trimmed Squares (LTS) [13], given by

$$(\widehat{\beta}_0, ..., \widehat{\beta}_p) \leftarrow \underset{\beta_0, ..., \beta_p}{arg\ min} \sum_{i=1}^{hp} \{r_1^2, ..., r_n^2\}_{i:n.} \tag{4}$$

where hp with $p \leq hp \leq n$ is typically set to $hp = (n + p + 1)/2$ for maximum breakdown point, and p (size of the sample) is an algorithm parameter. In the case of LMS, the idea is to use the median of the residuals instead of an aggregate fitness such as the average error. A generalization of this method is quantile regression [14]. Moreover, similar approaches have been applied with more sophisticated regression methods, such as random decision trees [15]. LTS

searches for a subset of training cases that give the lowest error, since the lowest error will be obtained when only inliers are present in the subset. Moreover, there is an efficient implementation of this algorithm called FAST-LTS; a review of robust methods can be found in [16]. These methods are indeed robust for linear regression, but only when the number of outliers does not exceed 50% of the training data, which is referred to as the breakdown point of the method. Beyond this breakdown point, these methods also fail, but consider that standard LS has a breakdown point of 0 and theoretically the 50% breakdown cannot be exceeded for linear regression problems. Moreover, recently it was shown that combining LMS and LTS with GP can allow it to solve symbolic regression problems with the same order of accuracy when the dataset is contaminated by as much as 50% of outliers, empirically showing that their breakdown point holds in symbolic regression with GP [2].

For problems where the contamination of the dataset is above 50%, sampling and approximate methods must be used. For instance, one approach is RANSAC [3], a sampling method to solve parameter estimation problems where the contamination level exceeds 50%. RANSAC has proven to be very useful in at least one domain, computer vision [4], and many different version of the algorithm have been derived such as the M-estimator Sample Consensus (MSAC), the Maximum Likelihood Estimation Sample and Consensus (MLESAC) [17], and the Optimal RANSAC algorithm [18]. Moreover, [2] has also shown that combining RANSAC with GP can achieve robustness in symbolic regression modeling in extreme contamination scenarios, with empirical evidence presented for up to 90% contamination by outliers. The main drawback with RANSAC is that it requires multiple random samples of the dataset, which have a low probability of being sufficiently clean (composed primarily by inliers) when the original dataset is highly contaminated, making a computationally expensive method[2].

3 Outlier Removal with Genetic Programming

Before describing the proposed algorithm, we must first state the main assumption on which it is based. Consider a training set $\mathbb{T} = \{(\mathbf{x}_i, y_i)\}$ where some fitness cases are inliers and other are outliers, where the l-th fitness case represents an outlier while the j-th fitness case represents an inlier. The assumption is that for a randomly generated GP tree (model or program) K, there is a high probability that the residual r_l is larger than the residual r_j. In other words, the residuals on the outliers will be larger than the residuals on the inliers for randomly generated models. While not at all obvious, there are some clear motivations for this assumption. First, random GP trees will only be able to detect simple and coarse relationships between the input and output variables, what can also be considered to be as low-frequencies in the signal. On the other hand, outliers will mostly appear as singularities in the training data, or high-frequency

[2] While it would be relatively simple to parallelize the algorithm, since all samples are taken independently, the cost can still become quite high if the modelling is done with GP.

components. Second, it is conceivable that a particular program might actually produce a low residual for one (or a few) outlier(s), and in this cases the assumption will not hold. However, since outliers do not follow a particular model (they are not noise), then the residuals in all other outliers can be expected to be relatively high. Finally, even if the majority of points in the training set are outliers this assumption can be expected to hold since the models are not fitted to the training data; i.e. the GP trees do not *learn* the outliers since they are randomly generated.

In what follows we will define an algorithm for detecting outliers based on this assumption and validate it in the experimental work reported afterward.

3.1 Proposed Algorithm

Based on the previous assumption, the proposed filtering process is summarized in Algorithm 1. The main inputs are the training set $\mathbb{T}$ of size n, and a percentile parameter ρ which defines the percentage of fitness cases that will be returned. In step 1, Ramped Half and Half is used to generate a total of p GP trees, using a specified function set F and a maximum depth d. The terminal set required to generate the random models is always composed by the input variables and randomly generated constants in the range $[-1, 1]$. Several informal tests showed that the method was quite robust to parameters d, p and F.

In step 2, the residuals of each GP tree on each fitness case is computed, constructing the matrix of residuals $R_{p\times n}$, where each element $r_{i,j}$ is the residual from the i-th model K_i (GP tree) on the j-th training instance $x_j \in \mathbb{T}$.

Step 3 is the key step, where the information contained in $R_{p\times n}$ is used to sort the training set and identify outliers, working under the assumption that outliers will have higher associated residuals for most GP trees. Therefore, we compute the column wise median of $R_{p\times n}$, generating a vector V of size n containing the median residual of each training instance evaluated over all random models. Therefore, set C will contain the ρ% of training instances from $\mathbb{T}$ that have the lowest associated median residuals.

Algorithm 1. Proposed algorithm for outlier removal.

Input: Contaminated training set $\mathbb{T}$ of size n.
Input: Cut-off percentile ρ $in(0, 1]$.
Input: Number of GP trees p.
Input: Function set F and model size parameter d.
Output: Set $C \subset \mathbb{T}$ of inliers.

1. Generate a random set P of models $k : \mathbb{R}^m \rightarrow \mathbb{R}$, with $|P| = p$ using F and d.
2. Obtain the matrix of residuals $R_{p\times n}$ such that each $r_{i,j}$ is the residual from each model $k_i \in P$ and each training instance $x_j \in \mathbb{T}$
3. Sort $\mathbb{T}$ based on the column wise median vector of $R_{p\times n}$, and return the lowest ρ% training instances in set C.

3.2 Discussion

There are two general strategies to deal with outliers. The first approach is to use the regression process to detect outliers and to basically build a model while excluding the outliers. This approach is taken by most of the robust techniques described above, such as LMS, LTS and even RANSAC, since the determination of which points are outliers depends on obtaining the residuals from a fitted model.

The second approach is to use a filtering process. A particularly well known filter is the Hampel identifier, where a data point (x_i, y_i) is tagged as an outlier if

$$| y_i - y_o | > t\zeta \tag{5}$$

where y_i is the value to be characterized, y_o is a reference value, ζ is a measure of data variation, and t is a user defined threshold [7]. The Hampel identifier uses a window W centered on x_i to compute y_o and ζ, with y_o set to the median of all y_j in W and ζ is 1.4826×MAD (Mean Absolute Deviation) within W; the value 1.4826 is chosen so that the results are not biased towards a Gaussian distribution.

The proposed method can be considered to be a hybrid between these two approaches. On the one hand, it is meant as a preprocessing step, used to remove outliers before another learning algorithm is applied to the data, thus it can be considered to be a filter. On the other hand, it is also based on the residuals computed for each training instance. However, unlike other robust methods, the residuals are derived from a random sampling of models, basically a population of GP trees, and learning or parameter fitting is not performed at all.

3.3 Related Works in GP

As stated above, [2] presents several results that are relevant to robust regression in GP. That work showed that both LMS and LTS are applicable to GP, and empirically their breakdown also applies to GP. Also, given the general usefulness of sampling the training instances to perform robust regression [16], that work also tested the applicability of sampling techniques in GP, such as interleaved sampling [19] and Lexicase selection [20]. Results showed that none of those approaches were useful for robust regression. The best results were obtained using RANSAC for sampling the training set and applying LMS on each selected subset, achieving almost equal test set prediction than directly learning on a clean training set. The method was called RANSAC-GP. The main drawback of RANSAC-GP is the high computational cost, since GP had to be executed on each sample and many samples were required as the percentage of outliers increases. Moreover, one underlying assumption of RANSAC-GP is that the GP search will be able to find a fairly accurate model on a clean subset of training examples, since models obtained from different samples will be discriminated based on their training performance. This assumption might not hold for some real-world problems.

Robust GP regression has not received much attention in GP, but some works are notable. In [9] GP and Geometric Semantic Genetic Programming (GSGP) are compared to determine which method was more sensitive to noisy data. The training sets are corrupted with Gaussian noise, up to a maximum of 20% of the training instances, concluding that GSGP is more robust when the contamination is above 10%. However, outliers are not considered. Another example is [10], in this case focusing on classification problems with GP-based multiple feature construction when the data set is incomplete, which can also considered to be outliers. The proposed method performs well, even when there is up to 20% of missing data, but extreme cases such as the ones tested here are not reported. A more related work is [21], where the authors build ensembles of GP models evolved using a multiobjective approach, where both accuracy and program size are minimized. The proposed approach is based on the same general assumption of many techniques intended to be robust to outliers, that model performance will be worse on outlier points that inliers. The ensembles are built from hundreds of indenpendent GP runs, a process that is much more expensive than the one proposed in the present work. Moreover, results are only presented for a single test case, where it is not known how many outliers are present, but results indicate that it is not higher than 5%. The method also requires human interpretation and analysis of the results, while the method proposed in this work is mostly automated except for the algorithm parameters.

4 Experimental Evaluation

4.1 Experimental Setup

As a first experimental test, we use the same procedure followed in [2]. First, we use the synthetic problems defined in Table 1. The datasets for each problem consist of 200 data points; i.e. input/output pairs of the form (x_i, y_i). The independent variable (input) was randomly sampled using a uniform distribution within the domain of each problem (see Table 1), and the corresponding value of the dependent variable (output) was then computed with the known model syntax. These represent the clean data samples or inliers of each problem. Then, these datasets were contaminated by different amounts of outliers, from 10% to 90% contamination in increments of 10%, for each. Thus, for each problem we have nine different datasets, each with a different amount of outliers. The proposed method is executed 30 times on each dataset, for each problem and for each level of contamination, to evaluate the robustness of the approach. To turn a particular fitness case (x_i, y_i) into an outlier, we first solve inequality 5 for y_i, such that

$$\begin{aligned} & y_i > y^o + t\zeta \\ \text{or} \quad & y_i < y^o - t\zeta. \end{aligned} \tag{6}$$

The decision to add or subtract from y_i, as defined in Eq. 5, is done randomly, and the value of t is set randomly within the range $[10, 100]$ to guarantee a large

Table 1. Benchmark problems used in this work, where $U[a, b, c]$ denotes c uniform random samples drawn from a to b, that specifies how the initial training sets are constructed consisting solely of inliers.

Objective function	Training set
$x^4 + x^3 + x^2 + x$	U[−1, 1, 200]
$x^5 - 2x^3 + x$	U[−1, 1, 200]
$x^3 + x^2 + x$	U[−1, 1, 200]
$x^5 + x^4 + x^3 + x^2 + x$	U[−1, 1, 200]
$x^6 + x^5 + x^4 + x^3 + x^2 + x$	U[−1, 1, 200]

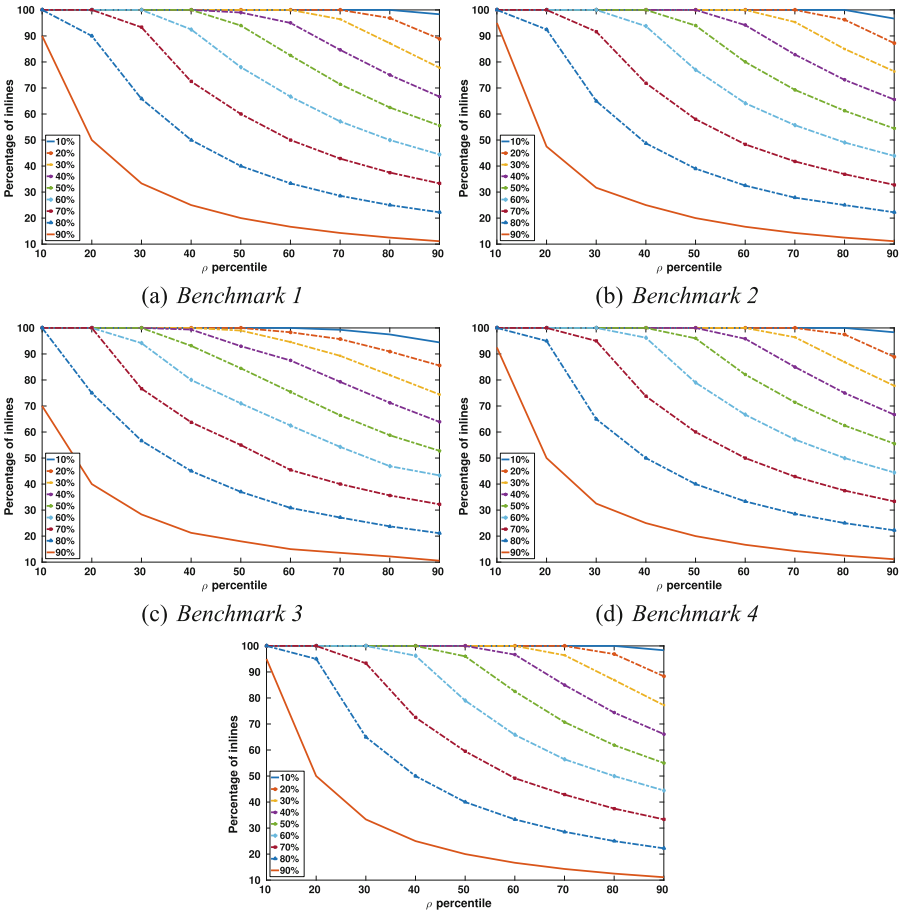
amount of deviance from the original data, with ζ computed by the median of all y_i within the function domain of each symbolic regression benchmark.

The parameters for the proposed method are set as follows. The function set is given by $F = \{+, -, \times, \div, sin, cos\}$ where $\div$ is the protected division, the maximum tree depth is set to $d = 3$, and the number of randomly generated models is $p = 100$. The percentile parameter ρ is evaluated from 10% to 90% in 10% increments. The method was coded using the Distributed Evolutionary Algorithms in Python library (DEAP) [22], basically building on top of the population initialization function.

4.2 Results

Figure 1 presents the main results. In each plot, the horizontal axis corresponds to the value of the ρ parameter, while the vertical axis represents the level of contamination in the output set C. In other words, the vertical axis shows the percentage of inliers contained in the *clean* set C, which in the best case would be 100%. However, it is important to remember, particularly when the contamination is above 50%, that a desired goal is for the vertical axis to be as high as possible, but in practice it can be sufficient if it is above 50%. In such a case it would be possible to use a robust regression method to solve the resulting modeling problem with set C. Each plot corresponds to one of the benchmarks from Table 1, and each shows nine curves, one for each contamination level. Each curve corresponds to the median performance over all 30 executions on each of the contaminated training sets.

All of the curves show a regular and informative pattern. First, on each problem the top curve corresponds to the lowest level of contamination 10%. As ρ increases, more points are returned as possible inliers but might in fact be outliers; i.e., C is larger, therefore the probability of the set being completely clean gradually declines. While the 10% level of contamination seems rather low in our tests, it is far above the breakdown point of non-robust regression methods. However, for this simplest case the percentage of inliers never falls below 90%. Second, as the level of contamination increases the performance on each problem gradually degrades, but not in a significant manner. Take for



(a) *Benchmark 1* (b) *Benchmark 2*

(c) *Benchmark 3* (d) *Benchmark 4*

(e) *Benchmark 5*

Fig. 1. Performance on the benchmark problems. The horizontal axis corresponds to the percentile parameter ρ, and the vertical axis represents the percentage of inliers in the resulting clean set C. Each curve represents the median value performance over 30 independent runs for each level of contamination.

instance the most extreme case, when contamination is at the 90% level. Using a conservative value for ρ of only 10%, the set returned contains a high amount of inliers. In 4 problems it is above 90% and in only one case it falls to about 70%. In this latter case, Benchmark 3, this means that the new training set C now contains only 30% of outliers instead of the original 90%. This is useful, since it is now possible to build a model using a robust regression approach, such as LMS or LTS. For all other contamination levels, the performance is even more encouraging. For example, for contamination at 80% or lower it would be possible to set $\rho = 30\%$ and produce a clean dataset that contains less than 40% of outliers. These are highly encouraging results, showing that the proposed

Table 2. Median performance on Benchmark 1, shown as the percentage of inliers in the returned clean set C; bold values represent the level of contamination where the number of detected inliers falls below 100%.

	ρ value								
Outliers	$\rho = 10$	$\rho = 20$	$\rho = 30$	$\rho = 40$	$\rho = 50$	$\rho = 60$	$\rho = 70$	$\rho = 80$	$\rho = 90$
10%	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**98.3**
20%	100.0	100.0	100.0	100.0	100.0	100.0	100.0	**96.8**	88.8
30%	100.0	100.0	100.0	100.0	100.0	100.0	**96.4**	87.1	77.7
40%	100.0	100.0	100.0	100.0	99.0	**95.0**	84.6	75.0	66.6
50%	100.0	100.0	100.0	100.0	**94.0**	82.5	71.4	62.5	55.5
60%	100.0	100.0	100.0	**92.5**	78.0	66.6	57.1	50.0	44.4
70%	100.0	100.0	**93.3**	72.5	60.0	50.0	42.8	37.5	33.3
80%	100.0	**90.0**	65.8	50.0	40.0	33.3	28.5	25.0	22.2
90%	**90.0**	50.0	33.3	25.0	20.0	16.6	14.2	12.5	11.1

Table 3. Median performance on Benchmark 3, shown as the percentage of inliers in the returned clean set C; bold values represent the level of contamination where the number of detected inliers falls below 100%.

	ρ value								
Outliers	$\rho = 10$	$\rho = 20$	$\rho = 30$	$\rho = 40$	$\rho = 50$	$\rho = 60$	$\rho = 70$	$\rho = 80$	$\rho = 90$
10%	100.0	100.0	100.0	100.0	100.0	100.0	99.2	97.5	**94.4**
20%	100.0	100.0	100.0	100.0	100.0	98.3	95.7	**90.9**	85.5
30%	100.0	100.0	100.0	100.0	99.0	94.5	**89.2**	81.8	74.4
40%	100.0	100.0	100.0	99.3	93.0	**87.5**	79.2	71.2	63.8
50%	100.0	100.0	100.0	93.1	**84.5**	75.4	66.4	58.7	52.7
60%	100.0	100.0	94.1	**80.0**	71.0	62.5	54.2	46.8	43.3
70%	100.0	100.0	**76.6**	63.7	55.0	45.4	40.0	35.6	32.2
80%	100.0	**75.0**	56.6	45.0	37.0	30.8	27.1	23.7	21.1
90%	**70.0**	40.0	28.3	21.2	18.0	15.0	13.5	12.1	10.5

method can identify outliers fairly easily using the proposed configuration. To better grasp these results, the numerical results for Benchmark 1 and Benchmark 3 are respectively shown in Tables 2 and 3. In each table, the bold value indicates when the median percentage of returned inliers falls below 100%.

5 Conclusions and Future Work

Dealing with outliers is a notoriously hard problem in regression. The algorithm presented in this work can effectively clean highly contaminated datasets. Standard regression techniques breakdown with even a single outlier in the training

set, while robust regression techniques fail when the contamination by outliers is greater than 50% on the training set. In such a cases, sampling techniques such as RANSAC are required, but the number of samples required grows rapidly with the percentage of outliers.

The proposed algorithm uses a random GP population to determine which training instances are inliers and which are not. It works under the assumption that outliers will be more difficult to model for randomly generated GP trees than inliers are; i.e. the residuals on outliers will be larger than on inliers. While robust regression methods also work under this assumption, this only holds after the model has been tuned, after learning has been performed. Moreover, this will only be possible if outliers represent a minority in the training set. On the other hand, the proposed algorithm does not perform any learning, basing its decision entirely on a random set of models. The proposed algorithm seems related to several other machine learning approaches. As stated above, it is obviously related to robust regression methods, particularly quantile regression, but without performing any model fitting. It is also related to RANSAC, since it performs a random sampling, but of models instead of training instances.

Results are encouraging, compared to other methods, only RANSAC can attempt to deal with problems where the level of contamination exceeds 50%. Take for instance the Hampel identifier, it would be useless since the median value in the dataset would be an outlier. Moreover, while RANSAC can deal with similar problems, its computational cost can become excessive and depends on the ability of the learning or modeling algrithm to extract relatively accurate models [2]. The proposed method is efficient, since it only requires generating and evaluating a single GP population.

Future work will focus on the following. First, extend the evaluation to real-world multi-variate problems, a more challenging scenario. Second, determine how specific parameters of the proposed algorithm affect performance, particularly the number of random models generated. Third, attempt to determine a *general* setting for ρ, at least experimentally. Fourth, clearly define how the proposed algorithm relates to other robust regression and learning algorithms. Finally, extend the method to deal with outliers in the input variables.

Acknowledgments. This research was funded by CONACYT (Mexico) Fronteras de la Ciencia 2015-2 Project No. FC-2015-2:944, BioISI R&D unit, UID/MULTI/04046/2013 funded by FCT/MCTES/PIDDAC, Portugal, and first author supported by CONACYT graduate scholarship No. 573397.

References

1. Trujillo, L., et al.: Neat genetic programming: controlling bloat naturally. Inf. Sci. **333**, 21–43 (2016)
2. López, U., Trujillo, L., Martinez, Y., Legrand, P., Naredo, E., Silva, S.: RANSAC-GP: dealing with outliers in symbolic regression with genetic programming. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E., García-Sánchez, P. (eds.) EuroGP 2017. LNCS, vol. 10196, pp. 114–130. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55696-3_8

3. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
4. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2004). ISBN 0521540518
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (2009)
6. Hodge, V.J., Austin, J.: A survey of outlier detection methodologies. Artif. Intell. Rev. **22**, 85–126 (2004)
7. Pearson, R.: Mining Imperfect Data: Dealing with Contamination and Incomplete Records. Society for Industrial and Applied Mathematics. SIAM, Philadelphia (2005)
8. Kotanchek, M.E., Vladislavleva, E.Y., Smits, G.F.: Symbolic regression via genetic programming as a discovery engine: insights on outliers and prototypes. In: Riolo, R., O'Reilly, U.M., McConaghy, T. (eds.) Genetic Programming Theory and Practice VII. Genetic and Evolutionary Computation, pp. 55–72. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-1626-6_4
9. Miranda, L.F., Oliveira, L.O.V.B., Martins, J.F.B.S., Pappa, G.L.: How noisy data affects geometric semantic genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 985–992. ACM, New York (2017)
10. Tran, C.T., Zhang, M., Andreae, P., Xue, B.: Genetic programming based feature construction for classification with incomplete data. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO 2017, pp. 1033–1040. ACM, New York (2017)
11. Rousseeuw, P.J.: Least median of squares regression. J. Am. Stat. Assoc. **79**(388), 871–880 (1984)
12. Alfons, A., Croux, C., Gelper, S.: Sparse least trimmed squares regression for analyzing high-dimensional large data sets. Ann. Appl. Stat. **7**(1), 226–248 (2013)
13. Giloni, A., Padberg, M.: Least trimmed squares regression, least median squares regression, and mathematical programming. Math. Comput. Model. **35**(9), 1043–1060 (2002)
14. Bertsimas, D., Mazumder, R.: Least quantile regression via modern optimization. ArXiv e-prints (2013)
15. Meinshausen, N.: Quantile regression forests. J. Mach. Learn. Res. **7**, 983–999 (2006)
16. Hubert, M., Rousseeuw, P.J., Van Aelst, S.: Statist. Sci. High-breakdown robust multivariate methods **23**, 92–119 (2008)
17. Torr, P.H., Zisserman, A.: MLESAC: a new robust estimator with application to estimating image geometry. Comput. Vis. Image Underst. **78**(1), 138–156 (2000)
18. Hast, A., Nysjö, J., Marchetti, A.: Optimal RANSAC-towards a repeatable algorithm for finding the optimal set (2013)
19. Gonçalves, I., Silva, S.: Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A.Ş., Hu, B. (eds.) EuroGP 2013. LNCS, vol. 7831, pp. 73–84. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37207-0_7
20. Spector, L.: Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion, GECCO Companion 2012, pp. 401–408. ACM (2012)

21. Kotanchek, M., Smits, G., Vladislavleva, E.: Pursuing the pareto paradigm: tournaments, algorithm variations and ordinal optimization. In: Riolo, R., Soule, T., Worzel, B. (eds.) Genetic Programming Theory and Practice IV. Genetic and Evolutionary Computation. Springer, Heidelberg (2007). https://doi.org/10.1007/978-0-387-49650-4_11
22. Fortin, F.A.: DEAP: evolutionary algorithms made easy. J. Mach. Learn. Res. **13**, 2171–2175 (2012)