Information and Software Technology 52 (2010) 1155–1166

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation

Adriano L.I. Oliveira *, Petronio L. Braga, Ricardo M.F. Lima, Márcio L. Cornélio

Center of Informatics, Federal University of Pernambuco, Recife 50.732-970, Brazil

ARTICLE INFO

Article history:
Received 26 August 2009
Received in revised form 11 May 2010
Accepted 14 May 2010
Available online 31 July 2010

Keywords:
Software effort estimation
Genetic algorithms
Feature selection
Support vector regression
Regression

ABSTRACT

Context: In software industry, project managers usually rely on their previous experience to estimate the number men/hours required for each software project. The accuracy of such estimates is a key factor for the efficient application of human resources. Machine learning techniques such as radial basis function (RBF) neural networks, multi-layer perceptron (MLP) neural networks, support vector regression (SVR), bagging predictors and regression-based trees have recently been applied for estimating software development effort. Some works have demonstrated that the level of accuracy in software effort estimates strongly depends on the values of the parameters of these methods. In addition, it has been shown that the selection of the input features may also have an important influence on estimation accuracy.
Objective: This paper proposes and investigates the use of a genetic algorithm method for simultaneously (1) select an optimal input feature subset and (2) optimize the parameters of machine learning methods, aiming at a higher accuracy level for the software effort estimates.
Method: Simulations are carried out using six benchmark data sets of software projects, namely, Desharnais, NASA, COCOMO, Albrecht, Kemerer and Koten and Gray. The results are compared to those obtained by methods proposed in the literature using neural networks, support vector machines, multiple additive regression trees, bagging, and Bayesian statistical models.
Results: In all data sets, the simulations have shown that the proposed GA-based method was able to improve the performance of the machine learning methods. The simulations have also demonstrated that the proposed method outperforms some recent methods reported in the recent literature for software effort estimation. Furthermore, the use of GA for feature selection considerably reduced the number of input features for five of the data sets used in our analysis.
Conclusions: The combination of input features selection and parameters optimization of machine learning methods improves the accuracy of software development effort. In addition, this reduces model complexity, which may help understanding the relevance of each input feature. Therefore, some input parameters can be ignored without loss of accuracy in the estimations.

1. Introduction

Experienced software project managers develop the ability to find the trade-off between software quality and time-to-market. The efficiency in resource allocation is one of the main aspects to find out such equilibrium point. In this context, estimating software development effort is essential.

An study published by the Standish Group's Chaos states that 66% of the software projects analyzed were delivered with delay or above the foreseen budget, or worse, they were not finished [8]. Failures rate of software projects is still very high [9,10]; it is estimated that over the last 5 years the impact of such software project failures on the US economy have cost between 25 billion and 75 billion [9,10]. In this context, both overestimates and underestimates of the software effort are harmful to software companies [7]. Indeed, one of the major causes of such failures is inaccurate estimates of effort in software projects [10]. Hence, investigate novel methods for improving the accuracy of such estimates is essential to strengthen software companies' competitive strategy.

Several methods have been investigated for software effort estimation, including traditional methods such as the constructive cost model (COCOMO) [11], and, more recently, machine learning techniques such as radial basis function (RBF) neural networks [12], MLP neural networks [26], multiple additive regression trees [30], wavelet neural networks [27], bagging predictors [13] and support vector regression (SVR) [10]. Machine learning techniques use data from past projects to build a

* Corresponding author. Tel.: +55 81 32668939.
E-mail addresses: alio@cin.ufpe.br (A.L.I. Oliveira), plb@cin.ufpe.br (P.L. Braga), rmfl@cin.ufpe.br (R.M.F. Lima), mlc2@cin.ufpe.br (M.L. Cornélio).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.05.009

regression model that is subsequently employed to predict the effort of new software projects.

Genetic algorithms (GAs) were shown to be very efficient for optimum or approximately optimum solution search in a great variety of problems. They avoid problems found in traditional optimization algorithms, such as returning the local minimum [14]. Recently, Huang and Wang proposed a genetic algorithm to simultaneously optimize the parameters and input feature subset of support vector machine (SVM) without loss of accuracy in *classification problems* [15]. Two factors substantially influence the accuracy and computation time of machine learning techniques: (1) the choice of the input feature subset and (2) the choice of the parameters values of machine learning techniques. Hence, according to Huang and Wang, the simultaneous optimization of these two factors improves the accuracy of machine learning techniques for classification problems.

Oliveira employed grid selection for optimizing SVR parameters for software effort estimation [10]. His work did not investigate feature selection methods; all input features were used for building the regression models. Huang and Wang demonstrated that the simultaneous optimization of the parameters and feature selection improves the accuracy of SVM results for classification problems [15]. Their results showed that GA-based method outperforms grid selection for SVM parameter optimization for classification problems [15]. These results motivated us to adapt the ideas of Huang and Wang for machine learning regression methods. In particular, we aim at reducing the number of input features, keeping the accuracy level for software effort estimates.

In this context, this paper adapts the method proposed by Huang and Wang [15] for feature selection and parameters optimization of machine learning methods applied to software effort estimation (a regression problem). The idea behind our method is to adapt the fitness function of the genetic algorithm and the parameters to be optimized. Notice that support vector machines for regression (SVR) has three important parameters, whereas SVM for classification has only two. Furthermore, our method generalizes the method of Huang and Wang [15], since we apply it to three different machine learning techniques (SVR, MLP neural networks and M5P model trees), whereas their method was developed and investigated solely for SVMs [15].

The main contributions of this paper are threefold: (1) to develop a novel method for software effort estimation based on genetic algorithms applied to input feature selection and parameters optimizations of machine learning methods; (2) to investigate the proposed method by applying it to three machine learning techniques, namely, (i) support vector regression (SVR), (ii) multi-layer perceptron (MLP) neural networks, and (iii) model trees; and (3) to show that our method outperforms recent methods proposed and investigated in the literature for software effort estimation [5,30,26,33,13,25,10].

This paper is organized as follows. Section 2 reviews the regression methods used in this paper and Section 3 reviews some basic genetic algorithm (GA) concepts. In Section 4, we present our GA-based method for feature selection and optimization of machine learning parameters for software effort estimation. The experiments and results are discussed in Section 5. Finally, Section 6 concludes the paper.

2. Regression methods

The goal of regression methods is to build a function $f(x)$ that adequately maps a set of independent variables $(X_1, X_2, \ldots, X_n)$ into a dependent variable Y. In our case, we aim to build regression models using a training data set to use it subsequently to predict the total effort for the development of software projects in man-months.

2.1. Support vector regression

Support vector machines (SVMs) are a set of machine learning methods used in many applications, such as classification and regression [16]. SVMs are based on structural risk minimization (SRM); instead of attempting to minimize only the empirical error, SVMs simultaneously minimize the empirical error and maximize the geometric margin [16]. This method has outperformed previous ones in many classification and regression tasks.

Support vector regression (SVR) was designed for regression problems [17]. SVR is a machine learning technique based on statistical learning theory, developed by Cortes and Vapnik [18]. The investigation of the application of SVR for software project effort estimation was originally carried out by Oliveira [10]. Oliveira showed that SVR outperforms both linear regression and radial basis functions neural networks (RBFNs) for software effort estimation in a data set of NASA software projects [10,12].

Consider a training data set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in \Re^d$ denotes an input vector and $y_i \in \Re$ its corresponding target value.

In ε-SVR, the aim is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for the training data set, and simultaneously is as flat as possible. This type of loss function defines a margin around the true outputs. The idea is that errors smaller than a certain threshold $\varepsilon > 0$ are rejected. That is, errors inside the margin are considered to be zero. On the other hand, errors caused by points outside the margin are measured by variables ξ and ξ^*.

In SVR for linear regression, $f(x)$ is given by $f(x) = \langle w, x \rangle + b$, with $w \in \Re^d$, $b \in \Re$ where $\langle .,. \rangle$ denotes the dot product in $\Re^d$. For the case of nonlinear regression, $f(x) = \langle w, \phi(x) \rangle + b$, where ϕ is some nonlinear function which maps the input space to a higher dimensional feature space ($\Re^d$). In ε-SVR, the weight vector w and the threshold b are chosen to optimize the following problem [16]:

$$\begin{aligned} &\text{minimize}_{w,b,\xi,\xi^*} \quad \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &\text{subject to} \quad \begin{cases} (\langle w, \phi(x_i)\rangle + b) - y_i & \leqslant \varepsilon + \xi_i \\ y_i - (\langle w, \phi(x_i)\rangle + b) & \leqslant \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geqslant 0 \end{cases} \end{aligned} \tag{1}$$

The constant $C > 0$ determines the trade-off between the model complexity, that is, the flatness of $f(x)$ and the amount up to which deviations larger than ε are tolerated. ξ and ξ^* are called *slack variables* and measure the cost of the errors on the training points. ξ measures deviations exceeding the target value by more than ε

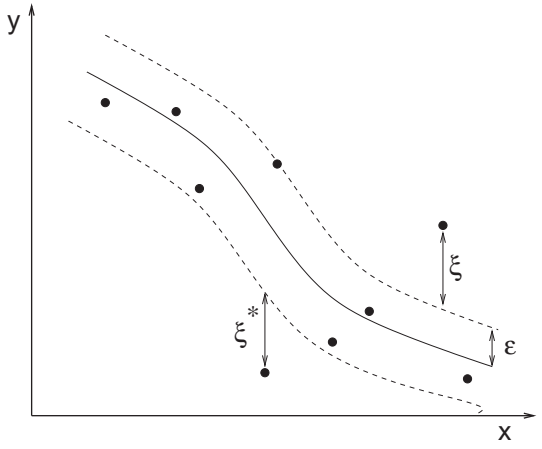

Fig. 1. Regression using ε-SVR.

and ξ^* measures deviations which are more than ε below the target value, as shown in Fig. 1. The idea of SVR is to minimize an objective function which considers both the norm of the weight vector w and the losses measured by the slack variables (see Eq. (1)). The minimization of the norm of w is one of the ways to ensure the flatness of $f(x)$ [16].

The SVR algorithm involves the use of Lagrangian multipliers, which rely solely on dot products of $\phi(x)$. This can be accomplished via *kernel functions*, defined as $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. Thus, the method avoids computing the transformation $\phi(x)$ explicitly. The details of the solution can be found in [16].

In this work, we consider SVR with both linear and RBF kernels. The polynomial kernel is computed as $K(x_i, x_j) = \langle 1 + (x_i), (x_j) \rangle^p$, whereas the RBF kernel is computed as $K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2)$, $\gamma > 0$. In most problems, the parameters C and ε significantly influence ε-SVR generalization performance. For the RBF kernel, the parameter γ must also be carefully selected. The linear kernel is the polynomial kernel with $p = 1$.

2.2. Multi-layer perceptron

The multi-layer perceptron (MLP) neural network has been applied successfully to a variety of problems such as classification, regression and time series forecasting [19,17]. MLPs are feedforward networks that consist of an input layer, one or more hidden layers and an output layer.

The number of nodes in the input layer is determined by the number of attributes of the input patterns (that is, the number of input features). In regression problems with a single independent variable there is just one output node, as shown in Fig. 2. In the project of an MLP the main parameters that influence performance are: (1) the number of hidden layers, (2) the number of neurons of each hidden layer, (3) the number of training epochs, and (4) the learning rate and (5) the momentum. The learning rate and the momentum are parameters of the back-propagation learning algorithm.

These parameters have to be carefully selected for boosting the performance of the MLP for a given problem. In most cases, they are determined using cross-validation. Tenfold cross-validation can be used to assess the generalization performance of classification and regression methods. In 10-fold cross-validation (CV), a given dataset is divided into 10 subsets. Next, a model is trained using a subset formed by joining nine of these subsets and tested by using the one left aside. This is done 10 times each employing a different subset as the test set and computing the test set error, E_i. Finally, the cross-validation error is computed as the mean over the ten errors E_i, $1 \leqslant i \leqslant 10$. The minimization of the cross-validation error is the criterion for selecting the parameters of the MLP such as the number of hidden neurons. Given the topology of the network, it is necessary to compute the weights of the connections. This is done by supervised training algorithms; the training algorithm most frequently employed for MLP networks is back-propagation [17].

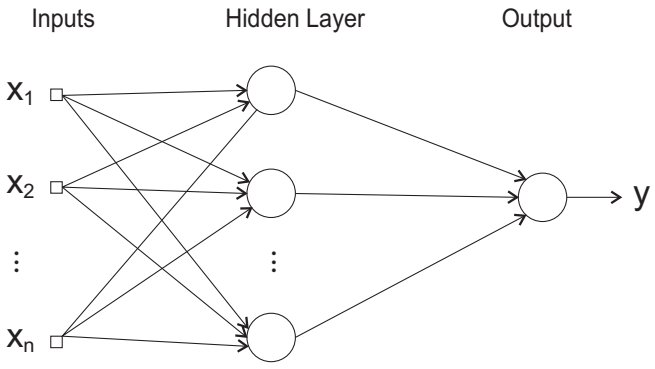

Fig. 2. An MLP with a single hidden layer and one output node.

Back-propagation is executed in two phases. The first phase is the forward phase; it involves feeding an input pattern to the input layer and propagating the signal until the output of the network to generate the prediction. The output furnished by the network is then compared to the known output and the error is computed. Next, the backward phase of the algorithm is executed. In this phase, the error is used to adjust the weights of the connections from the hidden to the output neurons The error is also backpropagated and used to adjust the weights of the connections from the input to the hidden neurons.

In the MLP network, the knowledge learned from the environment is represented by the value of the weights. The distributed nature of this knowledge makes it hard to interpret it, which is a weakness of MLPs for some practical applications [17].

2.3. Model trees

A tree, in computational terms, is a structure that represents objects (data, conditions, information, etc.) graphically. Decision trees are machine learning methods for classification and regression [20]. There are a number of types of decision trees that work with regression problems; in this paper we employ model trees.

The regression tree is a special type of decision tree developed for regression tasks [21]. This method performs induction by means of an efficient recursive partitioning algorithm, that is, the models are constructed by the divide-and-conquer method. The choice of each node of the tree is usually guided by a least squares error criterion. Binary trees consider two-way splits at each tree node.

Like the regression trees, the model trees are also decision trees. However, the main difference between regression trees and model trees is that the leaves of regression trees present numerical values, whereas the leaves of a model tree have linear functions which perform (linear regression).

One advantage of model trees over MLP neural networks is the capability of the former to produce more interpretable results, i.e., models which are much easier to understand.

In article we use the M5P algorithm [20–23]. M5P is powerful because it combines decision trees and linear regression for predicting a continuous variable [21]. Moreover, the M5P implements both regression trees and model trees [20]. Let T be a training data set. Each sample has a set of attributes (input features) that can be either numeric or discrete. The objective of M5P is to construct a model using the training set T that is able to predict values of the independent variable for novel samples.

The M5P algorithm adapts decision trees, originally developed for classification, to regression problems. These problems involve predicting a continuous numeric value instead of a discrete class. The M5P algorithm has three stages [20–22], which are outlined below:

(1) *Building a tree* – The first step consists in using a decision-tree induction algorithm to build a tree. For this induction, the splitting criterion is based on treating the standard deviation of the class values that reach a node as a measure of the error at that node. The expected reduction in error will be computed as a result of testing each attribute at that node. The attribute that maximizes the expected error reduction is chosen.
(2) *Pruning the tree* – The idea is to prune the tree back from the leaves, for as long as the expected estimated error decreases. This procedure uses an estimate of the expected error that

will take place at each node for test (unseen) data. To compute this error, the absolute difference between the predicted value and the actual class is averaged for each of the training examples that reach that node. This average will underestimate the expected error for test data, and thus the pruning procedure applies a factor to compensate [21].

(3) *Smoothing* – The smoothing process aims to compensate for the discontinuities that will take place between adjacent linear models at the leaves of the pruned tree. Smoothing works by first using the leaf model to compute the predicted value. Subsequently, that value is filtered along the path back to the root of the tree, smoothing it at each node by combining it with the value predicted by the linear model for that node. Wang and Witten have demonstrated that smoothing substantially increases the accuracy of the predictions [21].

3. Genetic algorithm

Genetic algorithm (GA) is a global optimization algorithm based on the theory of natural selection. In GA, individuals of a population having good phenotypic characteristics have greater survival and reproduction possibilities. As a result, the individuals less adapted to the environment will tend to disappear. Thus, GAs favor the combination of the individuals most apt, i.e., the candidates most promising for the solution of the problem [14]. It uses a random strategy of parallel search, directed to the search of points of high fitness, i.e., points in which the function to be minimized or maximized has relatively low or high values, respectively.

When GA is used for the resolution of real-world problems, a population comprised of a random set of individuals is generated (each individual of the population, called *chromosome*, normally corresponds to a possible solution of the problem). The population is evaluated during the evolution process. For each individual a rating is given, reflecting the degree of adaptation of the individual to the environment. A percentage of the most adapted individuals is kept, while that the others are discarded. The individuals kept in the selection process can suffer modifications in their basic characteristics through a mechanism of reproduction. This mechanism is applied on the current population aiming to explore the search space and to find better solutions for the problem by means of crossover and mutation operators generating new individuals for the next generation. This process, called reproduction, is repeated until a satisfactory solution is found [14].

3.1. Genetic operators

The genetic operators transform the population through successive generations, extending the search until arriving to a satisfactory or optimum result. Is very important that the population is diverse and keeps characteristics of adaptation acquired by the previous generations [14].

The intention of the selection operator is to select parents that increase the probability to reproduce members of the population that have good values of the objective function. There are some methods for the genetic selection, such as roulette wheel and tournament. The choice of an adequate mechanism of selection can help save significant computation time [14].

Crossover is the operator responsible for the recombination of characteristics of the parents during the reproduction, enabling the next generations to inherit these characteristics. It is considered the most important genetic operator [14]. The crossover operator exchanges genes between two chromosomes using the one point crossover, two-point crossover, or homologue crossover [14]. The mutation operators are necessary to guarantee a genetic diversity of the population; they do this by randomly modifying one or more components of a chosen structure. For instance, in binary coded genes, this is done by changing genes codes from 0 to 1 or vice versa [14]. Mutation makes possible the introduction of new elements in the population. Therefore, mutation assures that the probability of if arriving at any point of the search space will never be zero. The mutation operator is applied to the individuals with a probability given for mutation rate; in general a small mutation rate is used in practice.

4. GA-based feature selection and parameters optimization

4.1. Chromosome design

Machine learning techniques have parameters that almost always significantly influence the performance of these techniques. For instance, the complexity parameter, C, of support vector machines (SVMs) has an important influence on its performance and thereby needs to be carefully selected for a given data set. In this article we investigate three machine learning techniques, namely, SVR, MLP and M5P (briefly reviewed in Section 2). Table 1 shows the parameters of each of these techniques.

The chromosome is divided in two parts, as shown in Fig. 3. The first part, $G_1,\ldots,G_n$, represents the parameters of the machine learning technique that will be optimized. The second part, G_f, represents the input features that will be selected. The binary coding system was used to represent the chromosome.

In Fig. 3, the bit strings represent the genotype of the individual (parameters of the machine learning technique and input features). The transformation of genotype into phenotype is performed using Eq. (2). The precision of the parameters depend on the length of the bit strings; in this paper the length is 20. The minimum and maximum value of each parameter is decided by the user. In the chromosome, the description of feature representation is: the bit with value '1' indicates that the feature is not removed; '0' indicates that the input feature must be removed.

$$p = min_p + \frac{max_p - min_p}{2^l - 1} \times d \tag{2}$$

where p is the phenotype of bit string, min_p the minimum value of the parameter, max_p, maximum value of the parameter, d the decimal value of bit string and l is the length of bit string.

Table 1
Parameters of the machine learning techniques investigated in this article.

Technique	Parameters
SVR RBF	C – the complexity parameter ε – the extent to which deviations are tolerated γ – the kernel parameter
SVR linear	C – the complexity parameter ε – the extent to which deviations are tolerate
MLP	N – numbers for nodes in the hidden layer E – number of training epochs L – learning rate for the back-propagation algorithm M – momentum rate for the back-propagation algorithm
M5P	I – minimum number of instances per leaf P – if true, use unpruned tree S – if true, use unsmoothed predictions

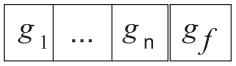

Fig. 3. The chromosome with two parts. The first representing the machine learning techniques parameters and the second representing input features data set.

4.2. Fitness function

There are several measurements used in the literature to evaluate the accuracy of prediction methods in software effort estimation tasks. In this paper we employ the two measurements that are most commonly used in the literature [24]. These measurements were also used by Shin and Goel [12], Oliveira [10], Braga et al. [13], amongst others. They are the Mean Magnitude of Relative Error (MMRE) and the PRED(25).

(1) Mean Magnitude of Relative Error (MMRE) is defined as:

$$\mathrm{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{Y_i}$$

where Y_i is the actual value in the project i and $\hat{Y}_i$ is its predicted value, and n is the number of samples. Good regression methods produce lower values of MMRE.

(2) PRED(25) is the percentage of predictions that fall within 25% of the actual value.

According to Kumar et al. [27] a software cost model is considered acceptably accurate if MMRE is at most 0.25 and PRED(25) is at least 75%.

In this paper we use two fitness functions. For five of the data sets (namely, the Desharnais, NASA, Albrecht, Kemerer and Koten and Gray [5] data sets), we aimed to find individuals (chromosomes) with high PRED(25) and a small MMRE. We solve this double criteria problem by creating a fitness function that combines the two criteria, that is, the fitness is defined as:

$$\mathrm{fitness} = (100 - \mathrm{PRED}(25)) + \mathrm{MMRE} \quad (3)$$

For the COCOMO data set we use a fitness function whose objective is to find individuals with a small *MMRE*. The choice of each fitness function is associated with a possibility of comparisons of our results to previous ones published in the literature. The comparisons were made considering results reported in [12,10,13,25,26]. For the COCOMO data set our results will be compared to a recent article by Tronto et al. which investigates the application of MLPs for software effort estimation [26]. Tronto et al. computed and attempted to minimize only the MMRE in their work, this is the reason for our selection of the fitness function for the COCOMO data set (which aims only to find the smaller MMRE).

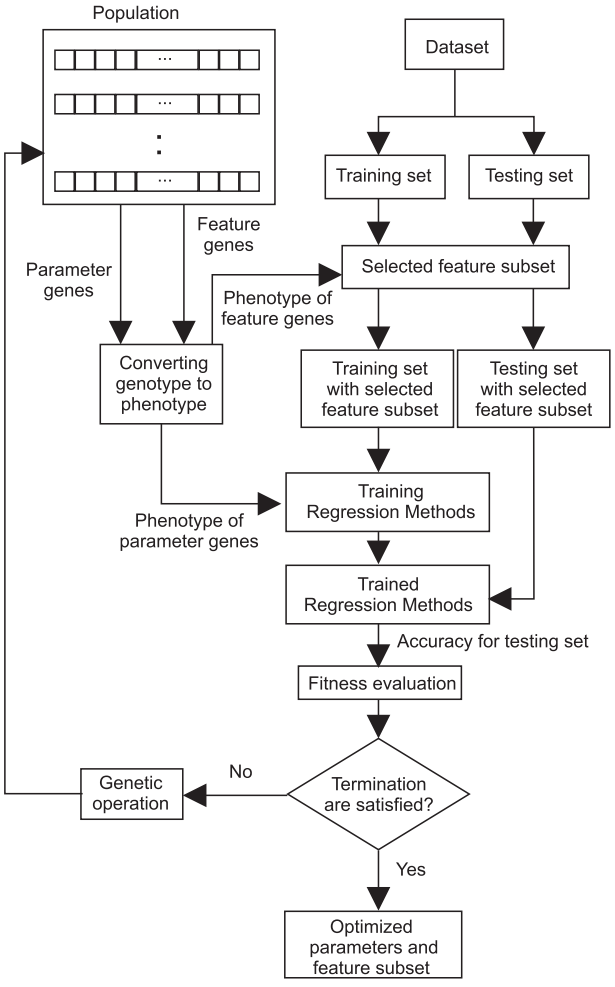

Fig. 4. System architecture of the proposed GA-based feature selection and parameters optimization for support vector regression.

4.3. System architecture for the proposed GA-based approach

Fig. 4 depicts the main steps of the proposed GA-based method. The explanation about the steps are:

(1) *Converting genotype to phenotype*: convert each part (genotype) of chromosome into a phenotype using Eq. (2).
(2) *Feature subset*: after converting each feature subset (genotype) chromosome into the phenotype, a feature subset can be selected.
(3) *Fitness evaluation*: each individual (chromosome) that represents machine learning techniques parameters and selected features is evaluated considering the fitness function.
(4) *Termination criteria*: if the termination criteria is satisfied, the process ends; otherwise, we proceed with the next generation or until the maximum number of generations defined by the user is reached.
(5) *Genetic operation*: in this step, the system searches for better solutions by applying genetic operations, including: selection, crossover, mutation, and elitism replacement.

5. Experiments

In this article we use six benchmark software effort data sets to evaluate the proposed method: (1) Desharnais data set [24,25], (2) a data set from NASA [12,10], (3) the COCOMO data set [26], (4) Albrecht data set [4,6], (5) Kemerer data set [11], and (6) a data set used by Koten and Gray [5], which contains data from database-oriented software systems that were developed using a specific 4GL toolsuite. These data sets are used in many articles to evaluate the performance of novel software effort prediction methods.

For each data set we have performed 10 simulations, since the results depend on the population randomly generated by the GA algorithm. This was done for all the machine learning techniques used in this paper. For all metrics, we report the average and standard deviation considering the 10 simulations performed.

In all simulations, the GA used two-point crossover randomly selected, roulette wheel selection and elitism replacement. There are many alternatives for the crossover and selection operators in genetic algorithms. In the future we may investigate other alternative and evaluate their influence in performance (computation time and accuracy).

For each data set, we have carried out simulations using several combinations of GA parameters. The values for each parameter were: (1) population size = {50, 100, 200, 300, 500}, (2) number of generations = {20, 25, 50, 100, 200}, (3) crossover rate = {0.5, 0.6, 0.65, 0.7, 0.8}, (4) mutation rate = {0.05, 0.1, 0.2, 0.3}.

The performance was influenced by the combination of GA parameters used, yet, in general, the best combination for all data sets was: population size = 500, number of generations = 25, crossover rate = 0.8, mutation rate = 0.3. For some data sets, another combination of parameters gives slightly better results. However, we decided to report the results for this combination for all data sets, since the difference in accuracy was not statistically significant.

In order to test if there is significant difference between the models generated by the GA-based method, Mann–Whitney U statistical tests with 95% confidence level (also called the Mann–Whitney–Wilcoxon (MWW), Wilcoxon rank-sum test, or Wilcoxon–Mann–Whitney test) were performed [2]. Mann–Whitney U is a non-parametric test for assessing whether two independent samples of observations have equally large values and is one of the best-known non-parametric significance tests. However, statistical analysis comparing our results to those of other papers was not possible because they do not report detailed results needed to perform such tests [5,30,12,26].

5.1. Performance metrics

Several works [3,1,31,32] argue that, despite the MMRE be probably the most widely used metric for assessing the accuracy of prediction models applied for software effort estimations, this metric is not consistent. They demonstrated that, in some cases, such metric does not aways indicates the best model. The same is observed for the PRED evaluation criterion.

We want to evaluate our method against results reported in scientific articles: [30,12,10,13,25,26]. Unfortunately, these papers' results are described in terms of MMRE and PRED metrics. Thus, in this paper, we decided to use these evaluation criteria, despite the problems mentioned.

In addition, in this paper we use metrics based on the absolute residuals [5], since such metrics are considered better for model comparison than MMRE or PRED(25) [31], in particular the standard deviation of the absolute residuals [5]. The absolute value of residual is given by:

$$Ab.Res. = |\text{actual value} - \text{predicted value}| \quad (4)$$

The metrics based on the absolute residuals used in this paper are: (i) the sum of the absolute residuals (Sum Ab.Res.), (ii) the median of the absolute residuals (Med.Ab. Res.), and (iii) the standard deviation of the absolute residuals (SD Ab.Res.). The Sum Ab.Res. measures the total residuals over the dataset. The Med.Ab.Res. measures the central tendency of the residual distribution. The SD Ab.Res. measures the dispersion of the residual distribution. Many other authors [30,12,10,13,25,26] do not adopt the metrics based on residuals. Therefore in five of the data sets our comparisons with other papers will involve only the MMRE and PRED(25). The comparison of metrics based on the absolute residuals will be possible only in the Koten and Gray data set, since these metrics are reported in their paper [5].

5.2. Desharnais data set

The Desharnais data set consists of 81 software projects described by 11 variables, nine independent and two dependent [24,25]. The categorical variable was coded using dummy variables, as in [1]. The first dependent variable is the effort. The second dependent variable (length) was ignored because our aim is to predict the total effort to develop software projects. For our simulations we divided the data set into two disjoint sets: the training set and the test set. To form the test set we randomly selected 18 out of the 81 projects [24,25]. The remaining 63 projects were used to form the training set. This procedure was also used by Burgess and Lefley [24] and Braga et al. [25]. The division of the data set into training and test sets was exactly the same employed by Braga et al. [25]. This makes possible to compare the results of the proposed method to the results reported by Braga et al. [25].

Table 2 shows the results of the simulations using the Desharnais data set. The results reported are the average from 10 simulations and the corresponding standard deviation. The Wilcoxon rank-sum tests indicate that both SVRs and MLP have equivalent performance regarding PRED(25) and that these methods outperform M5P in terms of PRED(25). The results also indicate that MLP obtained the best MMRE and the statistical tests indicate that the difference for the remaining methods is significant. In contrast, the M5P was the best method in terms of SD Ab.Res. and the differences to the other methods is statistically significant according to the Wilcoxon rank-sum tests.

It is also important to note that for this data set, the proposed method removed, for each method the number of input variables listed below (out of nine input variables):

(1) SVR RBF – 4.4 input features on average.
(2) SVR linear – 3.3 input features on average.
(3) MLP – 2.7 input features on average.
(4) M5P – 4.9 input features on average.

Table 3 compares the results obtained by our GA-based method to previous results obtained by Braga et al. [25]. These results show that the proposed method improved substantially the performance of SVR RBF and SVR-linear and improved the performance MLP and M5P, in terms of both MMRE and PRED(25). Notice also that the results of our method are better than those obtained by bagging MLP neural networks (see Table 3).

Table 2
Experimental results for Desharnais data set using GA-based approach.

	PRED(25)	MMRE	Sum Ab.Res.	Med. Ab.Res	SD Ab.Res.	Removed features
SVR with RBF Kernel						
Average	72.22	0.4051	33320.15	938.75	2709.61	4.40
Stand. dev.	0.00	0.0712	2427.82	141.59	558.34	0.52
SVR with linear Kernel						
Average	66.67	0.3685	30638.25	632.68	2238.69	3.30
Stand. dev.	0.00	0.0114	1472.51	37.95	110.21	0.48
Multi-layer perceptron						
Average	72.22	0.3154	26255.62	643.07	2251.30	2.70
Stand. dev.	0.00	0.0304	1287.93	146.03	245.33	1.16
M5P						
Average	61.11	0.5945	31819.01	639.31	2043.22	4.90
Stand. dev.	0.00	0.0000	0.00	0.00	0.00	0.88

Table 3
Comparison of the GA-based method to other methods on the Desharnais data set.

Method/parameters	PRED(25)	MMRE
SVR RBF ($C = 10$, $\varepsilon = 10^{-3}$, $\gamma = 10^{-2}$) (results from [25])	55.56	0.4736
MLP ($N = 9$, $E = 2000$, $L = 0.01$, $M = 0.3$) (results from [25])	66.67	0.4069
M5P ($I = 8$, $P =$ true, $S =$ false)/model trees (results from [25])	55.56	0.6135
Bagging ($S = 90$, $I = 10$, $C =$ MLP ($N = 12$, $E = 2500$, $L = 0.01$, $M = 0.3$)) (results from [25])	66.67	0.3991
Bagging ($S = 70$, $I = 10$, $C =$ M5P ($I = 8$, $P =$ true, $S =$ false))/model trees (results from [25])	55.56	0.6054
GA-based with SVR RBF	**72.22**	0.4051
GA-based with SVR-linear	66.67	0.3685
GA-based with MLP	**72.22**	**0.3154**
GA-based with M5P	61.11	0.5945

5.3. NASA data set

The NASA data set consists of two independent variables, Developed Lines (DL) and Methodology (ME), and one dependent variable, Effort (Y). DL is a measurement of the size of the software, given in KLOC (thousands of lines of code). ME is the methodology applied in the development of each software project. Y is the total effort for project development, and is given in man–months. For this data set we have employed leave-one-out cross-validation (LOOCV) [10] to estimate the generalization error; this procedure was also used by Shin and Goel [12], Oliveira [10], Elish [30] and Braga et al. [13].

The LOOCV technique consists in partitioning the data sets into n mutually exclusive partitions, where each partition has only one sample and n is the number of samples. Training takes place n times. For each training and test phase a difference sample i is chosen as the test sample. The remaining samples form the training set. After each training and test phase, the individual error $e_i = |Y_i - \hat{Y}_i|$ is computed, where Y_i is the actual effort of project i and $\hat{Y}_i$ is its predicted value. The individual errors are subsequently used to compute some important metrics which are used to evaluate the results.

Table 4 shows the results of the proposed GA-based method together with the amount of removed features using the NASA data set. The results reported are the average from 10 simulations and the corresponding standard deviation. They indicate that the best method for this data set is SVR with linear kernel. This is because the Wilcoxon rank-sum tests indicate that both SVRs and MLP have equivalent performance regarding PRED(25) and that these methods outperform M5P in terms of PRED(25). The statistical tests also indicate that both SVRs have equivalent performance regarding SD Ab.Res. However, SVR with linear kernel significantly outperforms the other methods regarding the remaining metrics (namely, MMRE, Sum Ab.Res. and Med. Ab.Res.).

Table 4
Experimental results for NASA data set using GA-based approach.

	PRED(25)	MMRE	Sum Ab.Res.	Med. Ab.Res	SD Ab.Res.	Removed features
SVR with RBF Kernel						
Average	94.44	0.1778	85.28	3.15	5.44	0.00
Stand. dev.	0.00	0.0148	13.84	0.39	0.76	0.00
SVR with linear Kernel						
Average	94.44	0.1650	75.58	2.01	5.46	0.00
Stand. dev.	0.00	0.0013	0.52	0.04	0.08	0.00
Multi-layer perceptron						
Average	94.44	0.1950	99.93	2.62	5.99	0.00
Stand. dev.	0.00	0.0021	0.74	0.08	0.05	0.00
M5P						
Average	83.33	0.1838	108.55	3.81	7.60	0.00
Stand. dev.	0.00	0.0000	0.00	0.00	0.00	0.00

Table 5
Comparison of the GA-based method with other methods on the NASA data set.

Method/parameters	PRED(25)	MMRE
MART (Huber-M) (results from [30])	83.33	0.1139
MART (least absolute deviation) (results from [30])	88.89	**0.0897**
MART (least squares) (results from [30])	83.33	0.1034
RBF-SG ($\delta = 0.1\%$, $\sigma = 0.85$, $m = 7$) (results from [12])	72.22	0.1907
SVR-linear ($C = 10$, $\varepsilon = 10^{-4}$) (results from [10])	88.89	0.1650
MLP ($N = 6$, $E = 2000$, $L = 0.01$, $M = 0.6$) (results from [25])	**94.44**	0.2030
M5P ($I = 7$, $P =$ true, $S =$ false)/model trees (results from [13])	83.33	0.1778
Bagging ($S = 90$, $I = 10$, $C =$ MLP ($N = 6$, $E = 2000$, $L = 0.01$, $M = 0.6$)) (results from [25])	**94.44**	0.1771
Bagging ($S = 100$, $I = 10$, $C =$ M5P ($I = 6$, $P =$ true, $S =$ false))/model trees (results from [13])	88.89	0.1639
GA-based with SVR RBF	**94.44**	0.1778
GA-based with SVR-linear	**94.44**	0.1650
GA-based with MLP	**94.44**	0.1950
GA-based with M5P	83.33	0.1838

Table 6
Variables of the COCOMO data set.

Number	Variable	Description
1	RELY	Required software reliability
2	DATA	Database size
3	CPLX	Process complexity
4	TIME	Time constraint for cpu
5	STOR	Main memory constraint
6	VIRT	Machine volatility
7	TURN	Turnaround time
8	ACAP	Analysts capability
9	AEXP	Application experience
10	PCAP	Programmers capability
11	VEXP	Virtual machine experience
12	LEXP	Language experience
13	MODP	Modern programing practices
14	TOOL	Use of software tools
15	SCHED	Schedule constraint
16	RVOL	Requirements volatility
17	MODE	Development model used (organic, semidetached e embedded)
18	ADJKDSI	Software size
19	EFFORT	Effort to develop the project

For this data set, no input feature was removed in the simulations. This demonstrates that the two independent input features are important for the prediction.

Table 5 compares the results obtained by the proposed method with previous results obtained by Elish [30], Shin and Goel [12], Oliveira [10], Braga et al. [13] and [25] for the NASA data set. The results given by the GA-based method are averages over 10 executions of the method (results reported in Table 4).

The results of Table 5 show that the proposed GA-based method improved the performance of the following methods: SVR RBF (*MMRE* and *PRED*(25)), SVR-linear (*MMRE* and *PRED*(25)) and MLP (*MMRE*). Notice that these are the best results obtained for the NASA data set regarding the PRED(25) (in comparison with the results obtained by Elish [30], Shin and Goel [12], Oliveira [10], Braga et al. [13] and [25]). Furthermore, notice that for this data set all but one method of Table 5 is accurate according to the criterion stated by Kumar et al. [27]. All methods obtained MMRE $\leqslant$0.25 and PRED(25) $\geqslant$75%, except the RBF-SG proposed by Shin and Goel [12] (which obtained PRED(25) = 72.22%).

These results also show that the MART methods [30] are superior regarding the MMRE, yet our proposed method achieves much better PRED(25). Foss et al. state that the MMRE is not recommended to evaluate and compare prediction models [31]. More recently, Port and Korte [32] observed that MMRE is sensitive to outliers; they are also uncertain about the true value of the error parameter estimated by MMRE. In contrast, they found that PRED is a reliable and robust estimator [32]. Therefore our model is more advantageous in practice in comparison to the MART methods, since it is much better regarding the PRED(25) estimator.

5.4. COCOMO data set

The COCOMO data set has 63 software projects [28], including business, scientific and system projects, described by 19 variables (18 input variables and 1 output variable), showed in Table 6. Each attribute of the project is categorized according with level of impact on effort estimation. The levels are: Very Low, Low, Nominal, High, Very High and Extra High. Each one of these levels has a numerical value associated. Tronto et al. [26] improved significantly the prediction performance in terms of MMRE obtained in previous works by using a preprocessing of the data set. The numerical value, for each level, was adjusted to attain a better discernibility power. For more details on the preprocessing, see Tronto et al. [26].

After preprocessing the data set, we divided it into six different pairs of training sets and test sets, showed in Table 7. This procedure was also used by Tronto et al. [26]. Next, we used these six pairs of training and test sets for training and testing the proposed method and to compare the results obtained with those of Tronto et al. [26]. Notice that for these data sets, our fitness function consists solely of the MMRE, since we aim to compare our results with those of Tronto et al. [26], which reports only the MMRE (and attempts to minimize only this measurement). The comparison is shown in Table 10.

Tables 8 and 9 show the average and standard deviation results of the proposed GA-based method together with the amount of removed features for the COCOMO data set. For each data set, we have performed 10 simulations and Tables 8 and 9 report the average and standard deviation of the results obtained. Notice that SVR with linear kernel outperformed SVR with RBF kernel for five of the data sets (in terms of both MMRE and PRED(25)). In data sets 1, 2, 3, 5, 6 the differences of PRED(25) for the SVR methods were statistically significant according to Wilcoxon rank-sum tests. The comparison of SVR with linear kernel to MLP shows that the former achieved significantly better results in only two of the data sets (data sets 2 and 5). SVR with linear kernel also obtained the best MMRE for five data sets (1, 3, 4, 5 and 6). However, the Wilcoxon rank-sum tests indicate that SVR with linear kernel performance for MMRE is statistically equivalent to that of MLP for four data sets (1, 2, 3, 5) and to SVR with RBF kernel in one data set (data set 4). The statistical tests also indicate that the results of SVR with linear kernel were significantly better than the results of M5P for all data sets (in terms of both MMRE and PRED(25)).

The best results for metrics based on the absolute residuals depend on the data set. For instance, for SD Ab.Res., SVR with linear kernel obtained the best result for data set 5; SVR with RBF kernel was the best in data sets 1 and 4; MLP was the best in data sets 3 and 6; finally, M5P achieved the best result for data set 2. Recall that in these experiments our fitness function was the MMRE, because we wanted to compare our results to a paper that reported only this metric [26]. This may explain the superiority of SVR with linear kernel in most data sets in terms of MMRE. In future works we may investigate a fitness function aimed to minimize the SD Ab.Res., which may help to improve SD Ab.Res. and indicate the superiority of one machine learning technique in this metric.

Tables 8 and 9 also show that a considerable number of input features were removed in all cases. For instance, for SVR with linear kernel, our proposed method removed on average (considering the six data sets) 6.72 input features (out of 18 input features).

Table 7
COCOMO data set: pairs of training sets and test sets.

Data set	Training set (projects)	Test set (projects)
1	All projects, except: 1, 7, 13, 19, 25, 31, 37, 43, 49, 55 and 61	1, 7, 13, 19, 25, 31, 37, 43, 49, 55 and 61
2	All projects, except: 2, 8, 14, 20, 26, 32, 38, 44, 50, 56 and 62	2, 8, 14, 20, 26, 32, 38, 44, 50, 56 and 62
3	All projects, except: 3, 9, 15, 21, 27, 33, 39, 45, 51, 57 and 63	3, 9, 15, 21, 27, 33, 39, 45, 51, 57 and 63
4	All projects, except: 4, 10, 16, 22, 28, 34, 40, 46, 52 and 58	4, 10, 16, 22, 28, 34, 40, 46, 52 and 58
5	All projects, except: 5, 11, 17, 23, 29, 35, 41, 47, 53 and 59	5, 11, 17, 23, 29, 35, 41, 47, 53 and 59
6	All projects, except: 6, 12, 18, 24, 30, 36, 42, 48, 54 and 60	6, 12, 18, 24, 30, 36, 42, 48, 54 and 60

Table 8
Average and standard deviation of experimental results for COCOMO data set using GA-based approach.

Data set		PRED(25)	MMRE	Sum Ab.Res.	Med. Ab.Res	SD Ab.Res.	Removed features
SVR with RBF Kernel							
1	Average	77.27	0.2120	1550.61	20.01	263.35	7.60
	Stand. dev.	4.79	0.0341	486.02	9.57	120.68	2.22
2	Average	71.82	0.2698	3725.58	48.86	791.98	6.50
	Stand. dev.	6.71	0.0464	2311.79	21.62	601.56	1.65
3	Average	70.91	0.3748	1176.43	28.23	217.16	9.20
	Stand. dev.	3.83	0.1799	563.75	12.54	159.74	2.15
4	Average	81.00	0.2076	416.37	11.72	68.94	8.70
	Stand. dev.	3.16	0.0101	58.31	5.30	6.81	1.42
5	Average	71.00	0.2323	269.55	6.34	47.99	6.30
	Stand. dev.	3.16	0.0334	74.96	3.09	13.94	1.16
6	Average	74.00	0.2295	5748.71	11.88	1656.21	8.30
	Stand. dev.	5.16	0.0360	3445.25	4.68	1140.54	1.57
SVR with linear Kernel							
1	Average	82.73	0.1781	1777.00	13.04	326.39	5.90
	Stand. dev.	2.88	0.0187	737.49	4.02	159.20	1.52
2	Average	79.09	0.3327	1707.29	51.43	273.25	6.20
	Stand. dev.	4.39	0.0795	482.33	8.79	118.59	2.00
3	Average	82.73	0.1840	1379.17	17.66	323.43	7.00
	Stand. dev.	2.87	0.0219	382.76	6.45	119.68	1.83
4	Average	81.00	0.2027	414.99	13.52	69.23	6.60
	Stand. dev.	3.16	0.0120	37.18	4.19	5.63	0.84
5	Average	85.00	0.1755	200.33	5.83	33.74	6.70
	Stand. dev.	5.27	0.0177	93.48	2.76	21.06	1.89
6	Average	80.00	0.1728	4398.11	11.58	1273.87	7.90
	Stand. dev.	0.00	0.0243	639.52	1.20	639.52	1.20

Table 9
Average and standard deviation of experimental results for COCOMO data set using GA-based approach.

Data set		PRED(25)	MMRE	Sum Ab.Res.	Med. Ab.Res.	SD Ab.Res.	Removed features
MLP							
1	Average	81.82	0.1942	1840.99	17.51	353.26	7.40
	Stand. dev.	0.00	0.0318	610.11	5.74	188.62	2.01
2	Average	70.91	0.2750	1609.91	53.81	230.14	6.90
	Stand. dev.	5.75	0.0494	594.39	26.62	108.56	1.60
3	Average	82.73	0.1697	683.91	18.64	125.71	8.30
	Stand. dev.	5.16	0.0229	161.14	6.25	57.54	1.95
4	Average	79.00	0.2379	478.44	16.17	71.74	9.20
	Stand. dev.	3.16	0.0255	70.10	4.20	6.92	1.03
5	Average	79.00	0.2352	222.90	4.97	39.63	5.60
	Stand. dev.	3.16	0.1018	50.45	1.72	14.58	1.71
6	Average	79.00	0.2044	2915.23	14.28	805.43	8.50
	Stand. dev.	3.16	0.0347	1480.68	8.90	451.32	0.97
M5P							
1	Average	76.36	0.2486	1715.71	23.93	272.26	6.70
	Stand. dev.	6.36	0.0888	363.14	12.88	92.04	1.50
2	Average	63.64	0.3037	1487.75	47.41	180.90	6.10
	Stand. dev.	0.00	0.0226	248.84	17.12	48.77	1.52
3	Average	67.27	0.3196	1048.65	23.92	173.44	8.80
	Stand. dev.	4.69	0.0776	213.95	8.26	65.02	0.79
4	Average	76.00	0.2654	458.09	16.55	71.62	8.40
	Stand. dev.	5.16	0.0152	64.33	5.54	7.63	2.22
5	Average	74.00	0.3807	287.76	8.67	48.75	7.90
	Stand. dev.	8.43	0.1849	155.18	3.65	30.03	1.45
6	Average	73.00	0.2189	7677.73	9.84	2317.45	8.70
	Stand. dev.	4.83	0.0273	456.20	1.25	456.20	1.25

Table 10
Comparison of the GA-based method with MLP [26] on the COCOMO data set.

Data set	ANN (MLP) [26]	GA-based SVR RBF	GA-based SVR linear	GA-based MLP	GA-based M5P
1	0.3647	0.2120	**0.1781**	0.1942	0.2486
2	0.4652	**0.2698**	0.3327	0.2750	0.3037
3	0.3795	0.3748	0.1840	**0.1697**	0.3196
4	0.2519	0.2076	**0.2027**	0.2379	0.2654
5	0.3987	0.2323	**0.1755**	0.2352	0.3807
6	0.6317	0.2295	**0.1728**	0.2044	0.2189

Table 10 compares the results obtained by our GA-based approach with previous results obtained by Tronto et al. [26]. The results given by our GA-based method are averages over 10 executions of the method. These results show that our proposed method improved substantially the results obtained by Tronto et al. [26] in all the six COCOMO data sets. Notice that we can only compare the MMRE, since Tronto et al. [26] do not report the PRED(25) obtained by their method. In addition, notice that the neural networks models investigated by Tronto et al. are not considered accurate according to the criterion used by Kumar et al. [27], since MMRE >0.25 for all COCOMO data sets (reported in [26] and reproduced here in Table 10). In contrast, our method applied to SVR and MLP obtained MMRE <0.25 for all but one COCOMO data set (see Table 10). If we consider also the PRED(25) metric and observe the results of Tables 8 and 9 we can see that both GA with SVR (linear kernel) and GA with MLP were able to satisfy the accuracy criterion (MMRE $\leqslant$0.25 and PRED(25) $\geqslant$75% [27]) for five out of six data sets (for data set 2, GA with SVR (linear kernel) and GA with MLP obtained MMRE $\geqslant$0.25; for data set 2, GA with MLP obtained PRED(25) <75%).

5.5. Albrecht data set

The Albrecht data set includes 24 projects developed by third generation languages [4,6]. This data set has eighteen projects written in COBOL, four written in PL1, and two written in DMS languages. There are five independent features: Inpcout (input count), Outcount (output count), Quecount (query count), Filcount (file count), and SLOC (lines of source code). The Effort is recorded in 1000 person hours and is the targeting feature of effort estimation. For this data set we have employed leave-one-out cross-validation (LOOCV) to estimate the generalization error [10], since this is a small data set.

Table 11 shows the results (average and standard deviations) of the proposed GA-based approach (MMRE, PRED(25) and SD), together with the average and standard deviation of the amount of removed features using the Albrecht data set. The results of Table 11 show that SVR with RBF kernel outperformed the other machine learning methods regarding MMRE, PRED(25), Sum Ab.Res. and SD Ab.Res. SVR with linear kernel obtained the best results in terms of Med. Ab.Res. The Wilcoxon rank-sum tests for the PRED(25), Sum Ab.Res., Med. Ab.Res, and SD Ab.Res.results have shown that there is significant difference from the results of SVR with RBF kernel to the other methods of Table 11. The statistical tests also have shown that there is significant difference from the results of MMRE from the SVR with RBF kernel to all other methods, except M5P. Notice also that SVR with both RBF and linear kernel remove around two input features, whereas MLP and M5P remove more than three input features.

We have not found research reported in the literature applying effort estimation to the Albrecht data set using LOOCV. Yet, recently Li et al. applied the Nonlinear adjustment to Analogy Based Estimation (NABE) and evaluated the method using threefold cross-validation [6]. The best results obtained by NABE were *PRED*(25) = 36, *MMRE* = 0.41. The best results reported in Table 11 (SVR with RBF kernel) are much better in terms of PRED(25) (*PRED*(25) = 70.41). Yet, NABE had slightly better performance regarding the MMRE (*MMRE* = 0.41) than our method (*MMRE* = 0.4465).

5.6. Kemerer data set

This section reports on experiments carried out using the Kemerer data set, which is a company-specific data from Kemerer's empirical work [11,33]. This data set contains data from 15 large

Table 11
Average and standard deviation of experimental results for Albrecht data set using GA-based approach.

	PRED(25)	MMRE	Sum Ab.Res.	Med. Ab.Res	SD Ab.Res.	Removed features
SVR with RBF Kernel						
Average	70.42	0.4465	107.11	2.34	6.42	2.20
Stand. dev.	2.36	0.0715	39.62	0.23	6.15	1.14
SVR with linear Kernel						
Average	56.25	0.6628	141.81	1.92	10.83	2.00
Stand. dev.	2.20	0.0277	7.55	0.41	0.21	0.00
Multi-layer perceptron						
Average	61.67	0.6863	135.27	2.68	7.83	3.80
Stand. dev.	1.76	0.2023	18.44	0.20	2.43	1.03
M5P						
Average	45.83	0.4700	168.19	3.56	8.51	3.00
Stand. dev.	0.00	0.0000	0.00	0.00	0.00	0.00

Table 12
Average and standard deviation of experimental results for Kemerer data set using GA-based approach.

	PRED(25)	MMRE	Sum Ab.Res.	Med. Ab.Res	SD Ab.Res.	Removed features
SVR with RBF Kernel						
Average	66.67	0.3695	829.05	29.02	67.88	1.70
Stand. dev.	0.00	0.0546	122.32	4.44	21.07	0.48
SVR with linear Kernel						
Average	60.00	0.4373	1295.73	51.98	136.50	2.20
Stand. dev.	0.00	0.0448	157.51	6.08	31.46	0.42
Multi-layer perceptron						
Average	64.00	0.3349	885.85	31.12	65.51	1.90
Stand. dev.	3.44	0.0191	102.20	7.03	8.17	0.32
M5P						
Average	46.67	0.5231	1346.90	44.30	138.33	2.90
Stand. dev.	0.00	0.0000	23.99	4.40	2.98	0.32

Table 13
Average and standard deviation of experimental results for Koten and Gray data set using GA-based approach.

	PRED(25)	MMRE	Sum Ab.Res.	Med. Ab.Res	SD Ab.Res.	Removed features
SVR with RBF Kernel						
Average	93.53	0.0947	626.96	25.56	37.27	2.70
Stand. dev.	1.86	0.0072	49.44	5.28	2.70	0.48
SVR with linear Kernel						
Average	94.12	0.0895	600.30	27.98	37.05	2.00
Stand. dev.	0.00	0.0024	14.74	2.97	0.47	0.00
Multi-layer perceptron						
Average	92.94	0.1219	802.68	38.13	39.24	2.90
Stand. dev.	2.48	0.0124	78.37	3.50	6.46	0.99
M5P						
Average	88.24	0.1164	751.28	37.09	34.12	3.00
Stand. dev.	0.00	0.0000	0.00	0.00	0.00	0.00

completed business data-processing projects of the same company. Each project has six input features: (i) programming language, (ii) hardware, (iii) duration, (iv) KSLOC, (v) AdjFP (adjusted function points), and (vi) RAWFP (raw function points). We used LOOCV in the experiments because the data set is small and to enable comparison to recent results reported by Li et al. using the same experimental setup [33].

Table 12 shows the average and standard deviation for each metric as well as for the number of removed features for the Kemerer data set. The results given by the GA-based method for each machine learning method are averages over 10 executions of the GA method. The results of Table 12 shows that the best machine learning method for this data set is the SVR with RBF kernel. The Wilcoxon rank-sum statistical tests indicate that both SVR with RBF kernel and MLP have equivalent performance regarding MMRE, Sum Ab.Res., Med. Ab.Res, and SD Ab.Res. Yet the statistical tests show that SVR with RBF kernel significantly outperforms all the other methods in terms of PRED(25).

Li et al. [33] do not reported PRED(25) nor the metrics based on absolute residuals. The best MMRE obtained by Li et al. [33] was MMRE = 0.636 (with their Optimal Linear Combining (OLC) method). Notice that our results reported in Table 12 are much better those of Li et al. (the average MMRE was from 0.3349 to 0.5231). In particular, the average MMRE obtained by our GA-based method applied to MLP and SVR with RBF kernel is approximately 50% smaller than that of the OLC method. And the standard deviation of our results are smaller than 0.06.

5.7. Koten and Gray data set

This data set has five input variables (features) and one target variable (the effort) [5]. The input variables include four specification-based software size metrics plus a metric that measures the productivity of a development team. The data set has a total of 17 software projects (small or medium sized TPSs and/or MISs, which were developed using Oracle's Designer 6i and Developer 6i. of a development team [5]).

The simulations were carried out using LOOCV to enable a direct comparison with the results reported by Koten and Gray [5]. Table 13 shows the results of the proposed GA-based method together with the amount of removed features for the Koten and Gray data set. For each metric Table 13 reports the average and standard deviation over 10 executions of the GA-based method. For the PRED(25) the best results were obtained by the SVR methods. The Wilcoxon rank-sum tests for the PRED(25) results have shown that there is significant difference only between the results of M5P and of each of the other methods of Table 13. The best results for MMRE were obtained by the SVRs and the statistical tests have shown that there is significant difference between the SVR results and those of MLP and M5P. However, there is no significant difference between the SVR with linear kernel and the SVR with RBF kernel. The results for the SD of the absolute residuals show that the best model is the M5P (with significant statistical differences for other models). On the other hand, SVR models outperform M5P regarding both Sum Ab.Res. and Med. Ab.Res. For this data set, the GA-based method removed an average of two input features (for SVR with linear kernel) to three input features (for M5P) out of five input features.

Table 14
Comparison of the proposed GA-based method in the Koten and Gray data set to the methods of Koten and Gray [5].

MMRE	Sum Ab.Res.	Med. Ab.Res	SD Ab.Res.
Reduced Bayesian regression [5]			
0.135	865.2	36.4	37.9
Reduced regression [5]			
0.136	861.1	42.6	37.0
SVR with RBF Kernel			
0.0947	626.96	25.56	37.27
SVR with linear Kernel			
0.0895	600.30	27.98	37.05
Multi-layer perceptron			
0.1219	802.68	38.13	39.24
M5P			
0.1164	751.28	37.09	34.12

Table 14 compares the best results of our method (reported in Table 13) to the results obtained by Koten and Gray [5]. It is not possible to apply statistical tests because Koten and Gray report only the average of each metric. However, the results of Table 14 indicate that our method achieved much better results than those of Koten and Gray regarding the five metrics considered here [5].

6. Conclusion

This article proposed and investigated a novel method for software effort estimation. The proposed method applies a genetic algorithm to simultaneously select the optimal input feature subset and the parameters of a machine learning technique used for regression.

Although very popular in the literature, as discussed in the papers [1,31,32] the metrics MMRE and PRED(25) do not present enough support for adequate statistical analysis. However, most of the recently published articles on software effort estimation adopt these metrics to evaluate the performance of their results. Thus,

since the most simulations results on publicly available data sets are presented in terms of MMRE and PRED(25), we decided to use them to compare our results with others recently reported in the literature. However, in order to strengthen the statistical analysis of our results, we also adopted three metrics based on the absolute residual, including the standard deviation of the absolute residual [31,5].

In this article we investigated three machine learning techniques used together with our method, namely: (i) support vector regression (SVR), (ii) MLP neural networks and (iii) decision trees (M5P). In our simulations, we used six benchmark data sets of software projects: Desharnais, NASA, COCOMO, Albrecht, Kemerer and Koten and Gray.

We compared our results to previous ones published in [33,5,30,12,10,13,25,26]. In all the data sets, our method achieved the best performance in terms of PRED(25). In the NASA dataset, the MART models [30] outperformed our model regarding the MMRE. In the Desharnais and COCOMO datasets, our method was superior to all other in terms of both PRED(25) and MMRE. We also compared our results in the Koten and Gray data set to the original article considering the metrics based on the absolute residuals. The comparison has shown that our proposed method outperforms the methods of Koten and Gray [5] in terms of the three metrics, namely, Sum Ab.Res., Med. Ab.Res., and SD Ab.Res.

In particular, the method proposed in this paper remarkably outperformed the results recently reported by Tronto et al. in the COCOMO data sets [26]. Our simulations have shown that the GA-based method was able to improve performance of the three machine learning techniques considered in the problem of software effort estimation. Furthermore, the use of GA for feature selection has resulted in significant reduction of the number of input features for two of the data sets considered (the Desharnais and the COCOMO data sets).

For future work we propose to investigate the use of another optimization method, namely, particle swarm optimization (PSO), for simultaneous feature selection and parameter optimization for software effort estimation. The motivation for such an investigation is that PSO was recently used for feature selection and SVM parameter optimization for *classification problems* [29]. Lin et al. have compared their method with that of Huang and Wang [15] and have shown that PSO applied to SVM outperforms GA applied to SVM in many classification problems [29]. Therefore, we aim to investigate in the future whether PSO is better than GA for feature selection and parameter optimization of machine learning methods applied to a regression problem (software effort estimation).

Acknowledgements

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08 and by Petrobrás.

References

[1] B. Kitchenham, E. Mendes, Why comparative effort prediction studies may be invalid, in: PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering, 2009, pp. 1–5. < http://doi.acm.org/10.1145/1540438.1540444>.
[2] W. Conover, Practical Nonparametric Statistics, Wiley Series in Probability and Statistics, 1998.
[3] B.A. Kitchenham, E. Mendes, G.H. Travassos, Cross versus within-company cost estimation studies: a systematic review, IEEE Transactions on Software Engineering (2007) 316–329.
[4] A. Albrecht, J. Gaffney Jr., Software function, source lines of code, and development effort prediction: a software science validation, IEEE Transactions on Software Engineering 9 (6) (1983) 639–648. <http://dx.doi.org/10.1109/TSE.1983.235271>.
[5] C. van Koten, A. Gray, Bayesian statistical effort prediction models for data-centred 4GL software development, Information & Software Technology 48 (11) (2006) 1056–1067. <http://dx.doi.org/10.1016/j.infsof.2006.01.001>.
[6] Y.F. Li, Min Xie, T.N. Goh, A study of the non-linear adjustment for analogy based software cost estimation, Empirical Software Engineering 14 (6) (2009) 603–643. <http://dx.doi.org/10.1007/s10664-008-9104-6>.
[7] R. Agarwal, M. Kumar, Yogesh, S. Mallick, R.M. Bharadwaj, D. Anantwar, Estimating software projects, SIGSOFT Software Engineering Notes 26 (4) (2001) 60–67. <http://doi.acm.org/10.1145/505482.505491>.
[8] Standish, Project Success Rates Improved Over 10 Years, Tech. Rep., Standish Group, 2004. <http//www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish>.
[9] R.N. Charette, Why software fails, IEEE Spectrum 42 (9) (2005) 42–49. <http://www.spectrum.ieee.org/sep05/1685>.
[10] A.L.I. Oliveira, Estimation of software project effort with support vector regression, Neurocomputing 69 (13–15) (2006) 1749–1753. <http://dx.doi.org/10.1016/j.neucom.2005.12.119>.
[11] C.F. Kemerer, An empirical validation of software cost estimation models, Communications of the ACM 30 (5) (1987) 416–429. <http://doi.acm.org/10.1145/22899.22906>.
[12] M. Shin, A.L. Goel, Empirical data modeling in software engineering using radial basis functions, IEEE Transactions on Software Engineering 26 (6) (2000) 567–576. <http://www.computer.org:80/tse/ts2000/e0567abs.htm>.
[13] P.L. Braga, A.L.I. Oliveira, G.H.T. Ribeiro, S.R.L. Meira, Bagging predictors for estimation of software project effort, IEEE International Joint Conference on Neural Networks (IJCNN2007) (2007) 1595–1600. <http://dx.doi.org/10.1109/IJCNN.2007.4371196>.
[14] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Wiley & Sons, NJ, 2006.
[15] C.-L. Huang, C.-J. Wang, A GA-based feature selection and parameters optimization for support vector machines, Expert Systems with Applications 31 (2) (2006) 231–240. <http://dx.doi.org/10.1016/j.eswa.2005.09.024>.
[16] A. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (3) (2004) 199–222. <http://eprints.pascal-network.org/archive/00002057/01/SmoSch03b.pdf>.
[17] S. Haykin, Neural Networks: A Comprehensive Introduction, Prentice-Hall, 1999.
[18] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297. <http://citeseer.ist.psu.edu/cortes95supportvector.html>.
[19] M.H. Hassoun, Fundamentals of Artificial Neural Networks, MIT Press, 1995.
[20] E.F. Ian, H. Witten, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufman, 2005.
[21] R.J. Quinlan, Learning with continuous classes, in: 5th Australian Joint Conference on Artificial Intelligence, World Scientific, Singapore, 1992, pp. 343–348. <http://citeseer.ist.psu.edu/quinlan92learning.html>.
[22] Y. Wang, I.H. Witten, Inducing Model Trees for Continuous Classes, 1997. <http://www.cs.waikato.ac.nz/ml/publications/1997/Wang-Witten-Induct.ps>.
[23] T. Menzies, Z. Chen, J. Hihn, K.T. Lum, Selecting best practices for effort estimation, IEEE Transactions on Software Engineering 32 (11) (2006) 883–895. <http://doi.ieeecomputersociety.org/10.1109/TSE.2006.114>.
[24] C.J. Burgess, M. Lefley, Can genetic programming improve software effort estimation? A comparative evaluation, Information & Software Technology 43 (14) (2001) 863–873.
[25] P.L. Braga, A.L.I. Oliveira, G.H.T. Ribeiro, S.R.L. Meira, Software effort estimation using machine learning techniques with robust confidence intervals, in: IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), vol. 1, 2007, pp. 181–185. <http://dx.doi.org/10.1109/ICTAI.2007.80>.
[26] I.F.B. Tronto, J.D.S. Silva, N. SantAnna, An investigation of artificial neural networks based prediction systems in software project management, Journal of Systems and Software 81 (3) (2008) 356–367. <http://dx.doi.org/10.1016/j.jss.2007.05.011>.
[27] K.V. Kumar et al., Software development cost estimation using wavelet neural networks, Journal of Systems and Software 81 (11) (2008) 1853–1867. <http://dx.doi.org/10.1016/j.jss.2007.12.793>.
[28] B. Boehm, Software Engineering Economics, Prentice-Hall, 1981.
[29] Shih-Wei Ling et al., Particle swarm optimization for parameter determination and feature selection of support vector machines, Expert Systems with Applications 35 (4) (2008) 1817–1824. <http://dx.doi.org/10.1016/j.eswa.2007.08.088>.
[30] M.O. Elish, Improved estimation of software project effort using multiple additive regression trees, Expert Systems with Applications 36 (7) (2009) 10774–10778. <http://dx.doi.org/10.1016/j.eswa.2009.02.013>.
[31] T. Foss et al., A simulation study of the model evaluation criterion MMRE, IEEE Transactions on Software Engineering 29 (11) (2003) 985–995. <http://doi.ieeecomputersociety.org/10.1109/TSE.2003.1245300>.
[32] D. Port, M. Korte, Comparative studies of the model evaluation criterions MMRE and PRED in software cost estimation research, in: Proc. of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2008, pp. 51–60. <http://doi.acm.org/10.1145/1414004.1414015>.
[33] Q. Li et al., Reducing biases in individual software effort estimations: a combining approach, in: Proc. of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2008, pp. 223–232. <http://doi.acm.org/10.1145/1414004.1414041>.