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

Manasa sandhya Gunda
20860532

April 27, 2020

1 Introduction

Usually, project managers estimate the measure of men/hours for each software project to be done. This is usually calculated with the instinct of the project managers. So estimation of resource allocation is one of the main factors to find the equilibrium. A report [1] released by the Standish Community Chaos tells that 66 per cent of the software projects were delivered with delay or over the estimated budget and some of them were not even completed. This tells that a wrong project estimate could fail the whole software development project, which can cause a considerable loss to the firm. The estimation done by project managers is highly dependent on the efficiency and expertise of the person.

To avoid this, recently machine learning algorithms like radial basis function (RBF) neural networks, multi-layer perceptron (MLP) neural networks, support vector regression (SVR) are used to calculate this estimation and have shown a considerable improvement in the performance. These machine learning algorithms can be used to create a regression model with previously available data of projects. The estimated regression model can now predict for new software projects data. The computation time and accuracy of machine learning algorithms are highly dependent on two factors: (1) The selection of the input feature subset; (2) The tuning of hyperparameters used.

Genetic algorithm (GA) is shown to optimize solutions in a wide range of applications. In the referred paper[17] genetic algorithm is used to optimize the machine learning algorithms. The main objective in this paper instead of using genetic algorithm, particle swarm optimization (PSO) algorithm is used. Even though the traditional genetic algorithm is well established, it has issues like the solution tends to converge to the local optima rather than global optima and the better solution is only the comparison to solutions of previous iterations. While the PSO has the memory component in the algorithm and the particles can update positions of themselves along with the velocities. When compared to GAs, PSOs are easy to implement since there is no crossover or mutations in this.

The expected outcomes from the paper can be: (1) Based on particle swarm optimization algorithm to develop a new method for the estimation of software effort and to ensure the mentioned two factors; (2) try to check the derived method by applying to different machine learning algorithms like support vector regression (SVR) and model trees for better performance; and (3) to show that the proposed method outperforms traditional GA method.

This paper comes in the category of ”Innovation by model transfer” since particle swarm optimization is used instead of a genetic algorithm for tuning the hyperparameters.

2 Literature Review

There are several methods which have worked on software effort estimation. [3] discusses the need for analysing the software development process as well the creation of new models for better estimation. To start with, the traditional algorithms, there are issues with models like COCOMO needs the data to be specific and vast [4] and function points model is hard to do and the output quality is not considered. The paper [3] evaluates few traditional algorithmic models on 15 large project datasets and also suggests the need for development of novel methods for software project estimation.

Fuzzy logic methods are a set of methods which can be used for software project estimates. [7] proposed two new methods based on fuzzy logic sizing. The size of the software is regarded as a triangular fuzzy number instead of using a single number. Malathi and Sridhar [8] suggest a novel optimization method based on linguistic quantifiers, fuzzy logic and analogy based reasoning to enhance the performance of the software project effort when the data is either categorical or numerical. The disadvantage of using a fuzzy logic system is the degrees of meaningfulness maintenance is difficult thus becomes hard to use.

The next trend in software estimation is the use of machine learning techniques to create a regression model. First started with linear regression analysis, [5] says the modelling approach generalising the linear regression functions that are built on Radial Basis Functions (RBFs) also possess excellent mathematical characteristics of best and universal approximation. [6] used Support Vector Regression algorithm using linear and RBF kernels to carry out experiments on software project datasets. Most recently, other machine learning algorithms such as wavelet neural networks, MLP neural networks, multiple additive regression trees and bagging predictors are used to do the software effort estimation.

While using machine learning algorithms, there is a need for tuning the hyperparameters so that performance is high. Manually tuning these parameters would not be the best choice. Thus optimization techniques are used to tune these parameters. In the paper [9] Engelbrecht says that genetic algorithms are better in metric evaluation that the traditional techniques like optimizing to the local minimum. Most of the literature work [17, 12, 13] is based on genetic algorithms.

Some other optimization methods also are used for parameter optimization like Grid selection method [17, 10]. The results were compared to the traditional methods and it is seen that the accuracies are better. Hybrid models with a genetic algorithm can also be used for software project estimation. A hybrid genetic algorithm [14] is an extension of genetic algorithm and is usually a genetic algorithm combined with local search or any other method.

Not only optimizing the parameters, but an optimum feature subset can also be selected without any loss of the machine learning algorithm. In particular, Huang and Wang suggested a method [11] of using genetic algorithm for feature subset selection and parameter optimization without any compromise in the performance of classification algorithms. They demonstrated that feature selection along with parameter optimization, can also improve the accuracies of the machine learning algorithms.

Particle swarm optimization (PSO) [2] is one more algorithm which can optimize the parameters of machine learning algorithms and also [15] state that particle swarm optimization is better than genetic algorithm since PSO is easier to implement, PSO does not require extra parameters like crossover rate and mutation rate which needs to be tuned. Not only these, but GA also has a limitation that most of the times the algorithm will converge to a local minimum than global minimum. However, PSO tends to converge to the global minimum. This also strengthens with [16] stating PSO used simultaneously for parameter optimization and feature subset improved the accuracies of some classification machine learning algorithms.

Even though PSO is suggested to use for software project estimation, it is not used to implement with a selected feature subset and with evaluation metrics mentioned in this paper. Hence this paper is proceeded with the novel method of using PSO for parameter optimization of machine learning techniques and with a randomly selected feature subset the techniques are evaluated and then try to increase the performance compared to traditional genetic algorithm.

3 Proposed Solution

To dive into the proposed solution first the regression methods and particle swarm optimization should be understood.

3.1 Regression methods

A regression method should be able to construct a function which will map a set of independent predictor variables to a dependent variable. The regression methods used in software project estimation are SVR with RBF kernel and model trees.

One of the methods for the regression problems in specific Support Vector Regression (SVR) is designed from SVM. Suppose we are given training data, the goal is ε-SVR [18] is to find a regression function $f(x)$ that deviates with atmost ε value from the prediction values from the training data. The idea is that errors smaller than a certain threshold $\varepsilon > 0$ are rejected. Errors inside the margin are considered to be zero. Outside the margin, errors are calculated by variables ξ and ξ^*.

Another one of the methods for classification and regression in machine learning is Decision trees. In this paper, we concentrate on model trees which is a type of regression tree. The regression tree is created explicitly for solving regression problems. A regression tree is constructed by the divide-and-conquer method. Each node is chosen by the least square error criterion. An essential difference between regression tree and model tree is that in regression trees leaves are in the form of numbers and in the model trees leaves are functions that do linear regression.

3.2 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is used for solving discrete and continuous optimization problems. PSO is a population-based stochastic approach. Particle swarm optimization falls under the techniques of swarm intelligence which are used to solve optimization problems.

In particle swarm optimization, particles which are simple software agents move in the search space of an optimization problem. In the given optimization problem, the position of a particle indicates a candidate solution. Every particle will try to update to a better position in the search space by changing the velocities. The velocity of each particle depends on the personal best position of a particle and global best position of all particles. PSO has been inspired by the social behaviour of fish schooling or bird flocking.

Each particle is described by a group of vectors denoted as $(X_i, V_i, P_i) \subset \mathbb{R}^d$ search space where X_i, V_i and P_i are the position, velocity and the personal best position of the ith particle respectively in N size population is denoted as:

$$X_i = (x_{i1}, x_{i2}, ..., x_{id}) \; for \; i = 1, 2, ..., N. \tag{1}$$

$$V_i = (v_{i1}, v_{i2}, ..., v_{id}) \; for \; i = 1, 2, ..., N. \tag{2}$$

$$P_i = (p_{i1}, p_{i2}, ..., p_{id}) \; for \; i = 1, 2, ..., N. \tag{3}$$

The best position in the entire population is denoted as:

$$P_g = (p_{g1}, p_{g2}, ..., p_{gd}) \; for \; i = 1, 2, ..., N. \tag{4}$$

From P_i and P_g the next iteration of velocity and in turn position is also calculated of the ith particle.

$$V_i(t+1) = w(t) \times V_i(t) + C_1 \times rand_1 \times (P_i(t) - X_i(t)) + C_2 \times rand_2 \times (P_g(t) - X_i(t)), \tag{5}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{6}$$

Where, $V_i(t+1)$ and $V_i(t)$ are the next and current velocity of ith particle respectively. $X_i(t+1)$ and $X_i(t)$ show the next and current position of ith particle. C_1 and C_2 are acceleration coefficients, $rand_1$ and $rand_2$ is uniformly distributed random coefficients in the interval of [0, 1] and w is inertia weight.

In this paper, for SVR with RBF kernel, the parameters that need to be optimized are C - the complexity, γ – the kernel and ε – the extent to which deviations are tolerated. For model trees, M5P algorithm is used to implement model tree in which the parameters to be optimized are I - a minimum number of instances per leaf, S – if true, use unsmoothed predictions, P – if true, use unpruned tree.

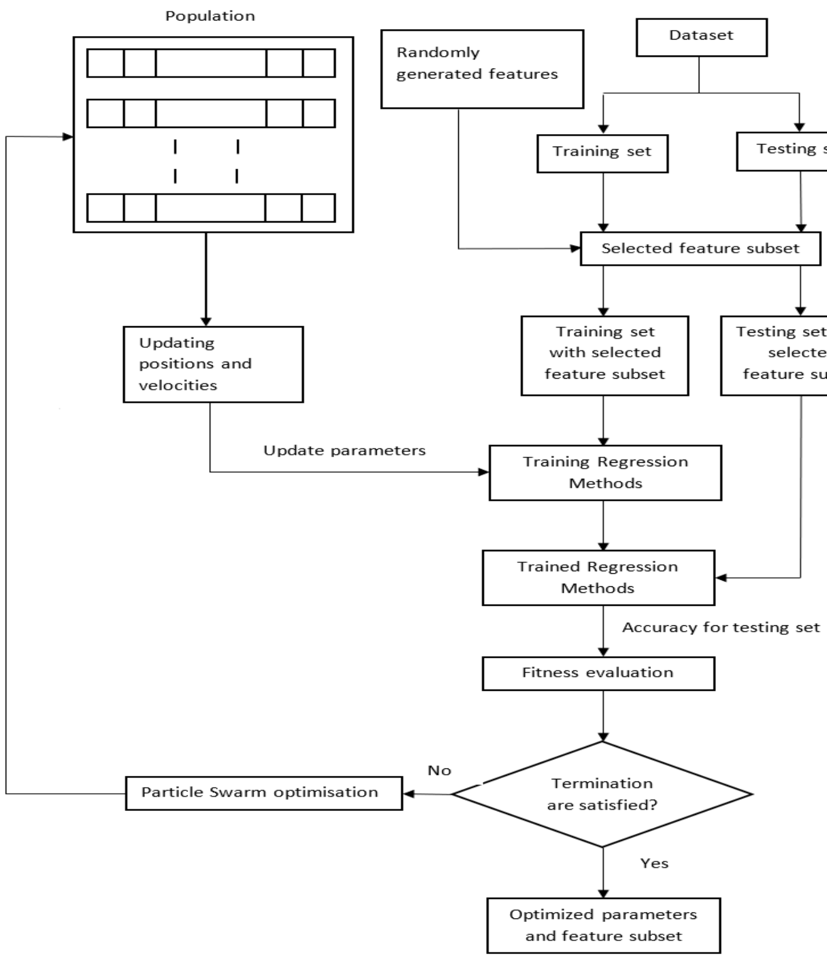
Population
Randomly generated features
Dataset
Training set
Testing set
Selected feature subset
Updating positions and velocities
Training set with selected feature subset
Testing set with selected feature subset
Update parameters
Training Regression Methods
Trained Regression Methods
Accuracy for testing set
Fitness evaluation
No
Particle Swarm optimisation
Termination are satisfied?
Yes
Optimized parameters and feature subset

Figure 1: Flowchart of proposed algorithm

3.3 System architecture for the proposed PSO-based approach

Fig. 1 is the flowchart for the proposed PSO-based approach instead of the GA. In this paper, each particle will have information of position, velocity and personal best position for all the parameters that need to be optimized.

1. Feature subset: the feature subset is randomly selected for all the population of particles. Unlike the approach mentioned in [17], these selected features are not updated with the iterations.

2. Fitness evaluation: each individual particle which has the parameters and randomly selected features are evaluated. The fitness functions used are explained in the next section.
3. Termination criteria: there two termination criteria for this algorithm one is it ends when best fiteness is achieved and the another is when the number of iterations are completed.
4. Particle swarm optimization operation: in this step, the system searches for better solutions by updating positions, velocities, personal best and global best positions of all the particles in the population.

4 Experimental Results

In this paper, the experiments are done on three datasets (1) Desharnais dataset, (2) Albrecht dataset and (3) Kemerer dataset which are used to evaluate the performances of new software effort prediction in numerous articles. The Desharnais dataset has information on 81 software projects described by 11 variables. The Albrecht dataset consists of 24 projects data. Finally, Kemerer dataset includes 15 huge business data-processing projects which are completed. The regression models are created using model trees and SVR with RBF kernel by optimizing their parameters using particle swarm optimization. This section contains the results of the simulations done with the PSO algorithm and is compared with the results in the referred paper [17], which used GA for optimizing parameters.

The code in the referred paper [17] was run for a set of total population, generations, crossover and mutation rate which would have taken weeks to run but the best ones were considered for the final results. Hence, the best combination mentioned is considered and all the results are calculated with 500 population, 25 epochs and 10-time simulations which will be then taken to calculate an average and a standard deviation.

4.1 Fitness functions

PSO iterations are updated on the basis of minimising the fitness of the regression method. In this paper, below are the metrics of evaluations used.

- PRED(25) is the percentage of predictions that are under 25 per cent of the actual values.
- Mean Magnitude of Relative Error (MMRE) is defined as $\frac{1}{n}\sum_{i=1}^{N}\frac{\mid Y_i - \hat{Y_i} \mid}{Y_i}$ where Y_i is the actual value of the outcome variable, $\hat{Y_i}$ is the predicted value of outcome variable and N is the total number of population.
- Sum Absolute Residual is defined as summation of the absolute residuals, $\mid Y_i - \hat{Y_i} \mid$.
- Median (Med.) Absolute Residual is defined as median of absolute residuals, $\mid Y_i - \hat{Y_i} \mid$.
- Standard deviation (SD) of Absolute Residual is defined as standard deviation of absolute residuals, $\mid Y_i - \hat{Y_i} \mid$.

PRED should be high and MMRE should be low, in order to fit both of these criteria the fitness function for which PSO updates positions in this paper is defined as

$$fitness = (100 - PRED(25)) + MMRE \tag{7}$$

4.2 Results for SVR with RBF kernel

Table 1: Results of Desharnais dataset

	PRED(25)	MMRE	Sum Ab. Res.	Med. Ab. Res.	SD Ab. Res.
with GA (results from [17])					
Average	72.22	0.4051	33320.15	938.75	2709.61
Stand. dev.	0.0	0.0712	2427.82	141.59	558.34
with PSO					
Average	33.333	1.1537	43217.5	1880.381	2733.278
Stand. dev.	0.0	0.0848	1053.8	73.28	43.404

Table 2: Results of Albrecht dataset

	PRED(25)	MMRE	Sum Ab. Res.	Med. Ab. Res.	SD Ab. Res.
with GA (results from [17])					
Average	70.42	0.4465	107.11	2.34	6.42
Stand. dev.	2.36	0.0715	39.62	0.23	6.15
with PSO					
Average	85.8333	0.1734	4.9202	4.92	0.0
Stand. dev.	14.383	0.1575	2.9647	2.97	0.0

Table 3: Results of Kemerer dataset

	PRED(25)	MMRE	Sum Ab. Res.	Med. Ab. Res.	SD Ab. Res.
with GA (results from [17])					
Average	66.67	0.3695	829.05	29.02	67.88
Stand. dev.	0.00	0.0546	122.32	4.44	21.07
with PSO					
Average	78.6667	0.2825	69.6237	69.6239	0.0
Stand. dev.	2.8109	0.0246	3.767	3.767	0.0

It is seen from Tables 2, 3 that in the case of Support vector regression with RBF kernel, for the datasets Albrecht and Kemerer using particle swarm optimization MMRE, Sum of Absolute Residuals, Median of Absolute Residuals, Standard deviation of Absolute Residual is heavily decreased and PRED(25) increased when compared to traditional genetic algorithm. This indicates improvement. PSO searches for global minima, unlike genetic algorithm, which sometimes searches for local minima. PSO does not have the extra parameters mutation rate and the crossover rate, which are used GA. Tuning these also highly impacts the results. Median of Absolute Residuals has a slight increased in both cases, but it is acceptable because some outliers in the outcomes can vary the medians of any data. In the case of Desharnais dataset Table 1, the results have slightly deferred and show that the fitness has not improved. This might be because of the reason the minimum and maximum of the parameters are not mentioned in the paper and the optimum parameters are chosen while simulating the models. Moreover, the other reason might be because the selected feature subset is not updated by the particle swarm optimization instead it is randomly generated. Even then, PSO has provided better results for Albrecht and Kemerer datasets.

4.3 Results for Model trees using M5P

Table 4: Results of Desharnais dataset

	PRED(25)	MMRE	Sum Ab. Res.	Med. Ab. Res.	SD Ab. Res.
with GA (results from [17])					
Average	61.11	0.5945	31819.01	639.31	2043.22
Stand. dev.	0	0.0000	0.00	0.00	0.00
with PSO					
Average	69.4444	0.2453	22010	475.5925	2009.3
Stand. dev.	4.7213	0.0286	1957.5	66.8580	351.1162

Table 5: Results of Albrecht dataset

	PRED(25)	MMRE	Sum Ab. Res.	Med. Ab. Res.	SD Ab. Res.
with GA (results from [17])					
Average	45.83	0.4700	168.19	3.56	8.51
Stand. dev.	0.00	0.0000	0.00	0.00	0.00
with PSO					
Average	98.3333	0.0248	0.6952	0.6952	0.00
Stand. dev.	2.1517	0.0163	0.4236	0.4236	0.00

Table 6: Results of Kemerer dataset

	PRED(25)	MMRE	Sum Ab. Res.	Med. Ab. Res.	SD Ab. Res.
with GA (results from [17])					
Average	46.67	0.5231	1346.90	44.30	138.33
Stand. dev.	0.00	0.0000	23.99	4.40	2.98
with PSO					
Average	93.3333	0.0866	53.9549	53.9549	0.0
Stand. dev.	0.00	0.0083	1.6901	1.6901	0.0

With model trees as a regression model with M5P algorithm, it is observed from Tables 4, 5, 6 that there is an improvement the evaluation metrics in all the datasets. The fitness evaluations using MMRE, Sum of Absolute Residuals, Median of Absolute Residuals, Standard deviation of Absolute Residuals and removed features have decreased considerably and PRED(25) has increased for all the datasets except for Median of Absolute Residuals of Keremer dataset which has slightly increased. However, the PRED(25) score has doubled for Albrecht and Keremer dataset which proves PSO has excellent improvement in converging to the optimum parameters of model trees.

5 Discussion and Conclusions

The report proposes the use of particle swarm optimization technique for optimizing the parameters of machine learning algorithms and feature selection. Even though the feature subset is not directly updated by the algorithm, these randomly generated features used in the calculation of fitness which in turn to updates the positions in PSO.

In the referred paper, a genetic algorithm is used for features selection and parameter optimization. However, genetic algorithm has some issues like converging to local minima. Accuracy is not accurate since some value is left when converting from binary to decimal and too many parameters to tune like crossover and mutation rates. Thus particles swarm optimization which converges at global minima and has fewer parameters is proposed to use instead of the traditional genetic algorithm.

For evaluation of fitness, a combination of MMRE and PRED [19, 20] are considered to satisfy the double criteria of decreasing the MMRE and increasing the PRED. But MMRE and PRED don't give any statistical analysis of the model. Thus metrics like sum, median and standard deviation of absolute residuals [19] are used to statistically analyse the fitness results.

This paper produces results for three datasets (1) Desharnais dataset, (2) Albrecht dataset and (3) Kemerer dataset and with two machine learning regression techniques (1) SVR with RBF kernel and (2) Model trees using M5P algorithm. The parameters like the number of populations, iterations and simulations are initialised to 500, 25 and 10 for the best optimum results.

Furthermore, from the results, it shows that the performance is better with particle swarm optimization when compared with genetic algorithm for both machine learning algorithms and three datasets except for the case SVR with RBF kernel using Desharnais dataset. The reason for this might be because it is not mentioned in the reference paper [17] the parameters like the number of population, iterations and simulations. Only the best combination of these parmaeters are given. Also, in this paper, the selected feature subset is not updated with the PSO can deter the performance of machine learning algorithm since the GA-based approach updates the selected feature subset also. But it should be noted that except for this case all the other cases are performing exceptionally well even without updating the selected feature subset. The PRED(25) score is observed to be almost doubled for the datasets Albrecht and Kemerer. Removed features are not included in the results even though it has improvements in results with PSO based algorithm because the proposed algorithm is not directly removing the features by updating them.

The personal best and global best positions and velocities of particles are initialized to the same as that of the first particle's position and velocities for better convergence. When tried with initializing to some random values in the ranges of parameters for positions and velocities, it is seen that the convergence of parameters for best fitness is slow. PSO runs at a time for multiple parameters which can be considered as different dimensions of the particle positions in the search space.

For the future works, the proposed model can be run for more regression methods like multi layer perceptron neural networks and SVR with linear kernel. This can be extended to run on more well established software project datasets like NASA, COCOMO, Koten and Gray for better analysis. The randomness component of the selected feature subset can be removed and made as a part of the position of the particles, this will improve the accuracies compared to the model proposed.

The proposed model itself can be changed, instead of traditional particle swarm optimization a hybrid particle swarm optimization would give better fitness. And other optimization techniques can be used like harmony search optimization and ant colony optimization. The motivation for these is in the paper [22] it is shown that ant colony optimization can be used for software project optimization along with the event based scheduling. And the inspiration for harmony search algorithm is from the paper [21] which states the improvement in accuracies for COCOMO effort estimation for NASA datasets. Therefore, there is a place for improvement in the proposed algorithm and this can be studied further for future.

References

[1] Clancy, T. (1995). The standish group report. Chaos report.

[2] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

[3] Kemerer, C. (1987). An empirical validation of software cost estimation models. (technical). Communications of the ACM, 30(5), 416–429. https://doi.org/10.1145/22899.22906

[4] Edin Osmanbegović, Mirza Suljić, & Haris Agić. (2017). A REVIEW OF ESTIMATION OF SOFTWARE PRODUCTS DEVELOPMENT COSTS. Ekonomski Vjesnik, 30(1), 195–207. Retrieved from https://doaj.org/article/3a46b00c3b4d4773837f727a7e3d2c84

[5] Miyoung Shin, & Goel, A. (2000). Empirical data modeling in software engineering using radial basis functions. IEEE Transactions on Software Engineering, 26(6), 567–576. https://doi.org/10.1109/32.852743

[6] Oliveira, A. (2006). Estimation of software project effort with support vector regression.(Author abstract). Neurocomputing, 69(13 15), 1749–1753. https://doi.org/10.1016/j.neucom.2005.12.119

[7] Harish Mittal, & Pradeep Bhatia. (2007). Optimization Criteria for Effort Estimation using Fuzzy Technique. CLEI Electronic Journal, 10(1). https://doi.org/10.19153/cleiej.10.1.2

[8] Malathi, S., & Sridhar, S. (2012). Optimization of Fuzzy Analogy in Software Cost Estimation using Linguistic Variables. Procedia Engineering, 38(C), 177–190. https://doi.org/10.1016/j.proeng.2012.06.025

[9] Engelbrecht, A. (2005). Fundamentals of computational swarm intelligence . Chichester, England;: Wiley.

[10] Oliveira, A. (2006). Estimation of software project effort with support vector regression. Neurocomputing, 69(13), 1749–1753. https://doi.org/10.1016/j.neucom.2005.12.119

[11] Huang, C., & Wang, C. (2006). A GA-based feature selection and parameters optimization for support vector machines. Expert Systems With Applications, 31(2), 231–240. https://doi.org/10.1016/j.eswa.2005.09.024

[12] Huang, S. J., Chiu, N. H., & Chen, L. W. (2008). Integration of the grey relational analysis with genetic algorithm for software effort estimation. European Journal of Operational Research, 188(3), 898-909.

[13] Huang, S. J., & Chiu, N. H. (2006). Optimization of analogy weights by genetic algorithm for software effort estimation. Information and software technology, 48(11), 1034-1045.

[14] Kelly Jr, J. D., & Davis, L. (1991, August). A Hybrid Genetic Algorithm for Classification. In IJCAI (Vol. 91, pp. 645-650).

[15] Hassan, R., Cohanim, B., De Weck, O., & Venter, G. (2005, April). A comparison of particle swarm optimization and the genetic algorithm. In 46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference (p. 1897)

[16] Lin, S., Ying, K., Chen, S., & Lee, Z. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Systems With Applications, 35(4), 1817–1824. https://doi.org/10.1016/j.eswa.2007.08.088

[17] Oliveira, A., Braga, P., Lima, R., & Cornélio, M. (2010). GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Information and Software Technology, 52(11), 1155–1166. https://doi.org/10.1016/j.infsof.2010.05.009

[18] Vapnik, V. (1995). The nature of statistical learning theory . New York: Springer.

[19] Foss, T., Stensrud, E., Kitchenham, B., & Myrtveit, I. (2003). A simulation study of the model evaluation criterion MMRE. IEEE Transactions on Software Engineering, 29(11), 985–995. https://doi.org/10.1109/TSE.2003.1245300

[20] Port, D., & Korte, M. (2008). Comparative studies of the model evaluation criterions mmre and pred in software cost estimation research. Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 51–60. https://doi.org/10.1145/1414004.1414015

[21] Jafari, S. S., & Ziaaddini, F. (2016, March). Optimization of software cost estimation using harmony search algorithm. In 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC) (pp. 131-135). IEEE.

[22] Wei-Neng Chen, & Jun Zhang. (2013). Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler. IEEE Transactions on Software Engineering, 39(1), 1–17. https://doi.org/10.1109/TSE.2012.17

Appendices

A PSO algorithm implementation in matlab

```
function [best_fitness, MMRE_best, PRED_25_best, Med_Ab_Res_best,...
         Sum_Ab_Res_best, SD_Ab_Res_best, best_rf] = ...
          PSO(population_size, number_of_iterations, regression_method,...
          X_train, Y_train, X_test, Y_test, dataset_name)


if strcmp(regression_method, 'SVR with RBF kernel')
    parameters_number = 3;
    range_of_parmaters{1} = [80, 150];    %--> C
    range_of_parmaters{2} = [0.01, 1]; %--> epsilon
    range_of_parmaters{3} = [0.01, 1]; %--> gamma

elseif strcmp(regression_method, 'M5P')
    parameters_number = 3;
    range_of_parmaters{1} = [2, 20];    %--> I
    range_of_parmaters{2} = [0, 1]; %--> P
    range_of_parmaters{3} = [0, 1]; %--> S
end

w=0.2;                % Inertia Weight
wdamp=1;              % Inertia Weight Damping Ratio
c1=0.7;               % Personal Learning Coefficient
c2=1.0;               % Global Learning Coefficient


number_of_features = size(X_train, 2);
dimension = 2*(parameters_number) + number_of_features;

%%%% initialize population:
particles = zeros(population_size, dimension);
for particle_index = 1:population_size
    while true
        particles(particle_index, :) = round(rand(1, dimension));
        selected_features = particles(particle_index,...
            end-number_of_features+1 : end);
        if ~sum(selected_features) == 0 %--> not all features are removed
            break
        end
    end
end
best_parameters_personal= zeros(population_size,parameters_number);
for parameter_index = 1:parameters_number
    VarMin = range_of_parmaters{parameter_index}(1);
    VarMax = range_of_parmaters{parameter_index}(2);
    VelMax = 0.1*(VarMax−VarMin);
    VelMin = −VelMax;
    particles(:,2*(parameter_index−1) + 1) =...
        VarMin+(VarMax−VarMin)*rand(population_size,1);
    best_parameters_personal(:,parameter_index)=...
        particles(:,2*(parameter_index−1) + 1);
    particles(:,2*(parameter_index−1) + 2) =...
        VelMin+(VelMax−VelMin)*rand(population_size,1);

end


%% PSO Main Loop


best_fitness_personal= Inf(population_size,1);
best_fitness= Inf;
best_parameters_global= best_parameters_personal(1,:);
for iteration = 1:number_of_iterations

    fitness = zeros(population_size, 1);
    MMRE = zeros(population_size, 1);
    PRED_25 = zeros(population_size, 1);
    Sum_Ab_Res = zeros(population_size, 1);
    Med_Ab_Res = zeros(population_size, 1);
    SD_Ab_Res = zeros(population_size, 1);
    for particle_index = 1:population_size

        particle = particles(particle_index, :);
        %%%% extract parameters out of particles:
        parameters = zeros(parameters_number, 1);
        velocities = zeros(parameters_number, 1);
        for parameter_index = 1:parameters_number
            parameters(parameter_index) =...
                particle(2*(parameter_index−1) + 1);
            velocities(parameter_index) =...
                particle(2*(parameter_index−1) + 2);
        end


        for parameter_index = 1:parameters_number
            VarMin = range_of_parmaters{parameter_index}(1);
            VarMax = range_of_parmaters{parameter_index}(2);
            VelMax = 0.1*(VarMax−VarMin);
            VelMin = −VelMax;
            r1 = rand;
            r2 = rand;
            velocity = velocities(parameter_index);
            position = parameters(parameter_index);
            % Update Velocity
            velocity = w*velocity ...
                +c1*r1*(best_parameters_personal...
                (particle_index, parameter_index)-position) ...
                +c2*r2*(best_parameters_global(parameter_index)-position);

            % Apply Velocity Limits
            velocity = max(velocity,VelMin);
            velocity = min(velocity,VelMax);

            % Update Position
            position = position + velocity;

            % Velocity Mirror Effect
            IsOutside=(position<VarMin | position>VarMax);
            velocity(IsOutside)=-velocity(IsOutside);

            % Apply Position Limits
            position = max(position,VarMin);
            position = min(position,VarMax);

          %  disp(position);
          %  disp(velocity);

            particle(2*(parameter_index-1) + 1) = position;
            particle(2*(parameter_index-1) + 2) = velocity;

        end
        particles(particle_index, :) = particle;
        %disp(['The result is: [' num2str(parameters(:).') ']']) ;
        %%%% extract selected features out of chromosome:

        selected_features = particle(end-number_of_features+1 : end);
        if sum(selected_features) == 0
            a_feature_to_be_mutated =...
                round(1 + (number_of_features - 1) * rand);
            particles(particle_index,...
                end-number_of_features+a_feature_to_be_mutated) = 1;
            selected_features =...
                particles(particle_index, end-number_of_features+1 : end);
        end

        %%%% evaluate the fitness of particles using PRED(25) and MMRE:
        [fitness(particle_index), MMRE(particle_index),...
            PRED_25(particle_index), Sum_Ab_Res(particle_index),...
            Med_Ab_Res(particle_index), SD_Ab_Res(particle_index)] = ...
```

```
            calculate_fitness(X_train, Y_train, X_test, Y_test,...
            regression_method, parameters, selected_features, dataset_name);

            % Update Personal Best
            if (fitness(particle_index)< best_fitness_personal(particle_index))
                best_fitness_personal(particle_index)= fitness(particle_index);
              % disp(best_fitness_personal);
                best_particle_index_personal = particle_index;
                best_parameters_personal(particle_index,:) = parameters;
                if best_fitness_personal(particle_index) < best_fitness
                    best_fitness = best_fitness_personal(particle_index);
                    best_particle_index = particle_index;
                    best_parameters_global = parameters;
                    best_particle = particles(best_particle_index, :);
                    best_selected_features =...
                        best_particle(end-number_of_features+1 : end);
                    best_rf = sum(best_selected_features == 0);
                    MMRE_best = MMRE(best_particle_index);
                    PRED_25_best = PRED_25(best_particle_index);
                    Sum_Ab_Res_best = Sum_Ab_Res(best_particle_index);
                    Med_Ab_Res_best = Med_Ab_Res(best_particle_index);
                    SD_Ab_Res_best = SD_Ab_Res(best_particle_index);
                end
            end


        end



        w=w*wdamp;

    end
```