Journal of Information Technology and Computer Science
Volume 1, Number 1, 2016, pp. 28 – 37
Journal Homepage: www.jitecs.ub.ac.id

Optimizing SVR using Local Best PSO for Software Effort Estimation

Dinda Novitasari[1], Imam Cholissodin[2], Wayan Firdaus Mahmudy[3]

[1,2,3]Department of Informatics/ Computer Science, Brawijaya University, Indonesia
[1]id.dindanovitasari@gmail.com, [2]imamcs@ub.ac.id, [3]wayanfm@ub.ac.id

Received 21 February 2016; received in revised form 18 March 2016; accepted 25 March 2016

Abstract. In the software industry world, it's known to fulfill the tremendous demand. Therefore, estimating effort is needed to optimize the accuracy of the results, because it has the weakness in the personal analysis of experts who tend to be less objective. SVR is one of clever algorithm as machine learning methods that can be used. There are two problems when applying it; select features and find optimal parameter value. This paper proposed local best PSO-SVR to solve the problem. The result of experiment showed that the proposed model outperforms PSO-SVR and T-SVR in accuracy.
Keywords: *Optimization, SVR, Optimal Parameter, Feature Selection, Local Best PSO, Software Effort Estimation*

1 Introduction

Software effort estimation needs techniques to make the approximate in an attempt to improve accurately all requirements. Some projects are proved to have problems at the time of completion and costs swell, i.e. around 43% till 59% of the incident (information from Standish Group). It is heavily influenced by the strategies and considerations used in the initial process [1]. The issue is highly developed and must be resolved, because so far mostly relied on the help of an expert assessment, but the results will be very biased, because it looks less objective, which will have an impact on the value of the results of the final evaluation of the estimate is not good [2].

Clever algorithm as machine learning has many help in overcoming the problem of engineering [3]. The advantage of machine learning is able to learn from previous data patterns adaptively and provide models and the results are consistent and stable [4]. For example, SVM which well known as a robust machine learning [5]. A form of development SVM is SVR that is designed specifically for producing optimal performance machine prediction/ forecasting. The machine's main problem is the difficulty of determining the optimal parameter values and the difficulty of determining the selection of optimal features as well [6],[7],[8]. Previously, Braga has tried using GA and Hu technique with PSO to obtain the optimal SVR results [9],

[10]. The result, PSOs provide the excellent performance and effective in finding the most optimal solutions, rather than GA and others [11]. PSO improvements have been made to deal with premature convergence (local optimum), although the time it takes a little longer, but can still be tolerated and comparable to the best optimization results obtained, the name of the method is the Local best PSO utilize ring topology that illustrated in Fig. 1 [12],[13]. Thus, based on that reason, ring topology-based local best PSO-SVR is proposed in our paper.

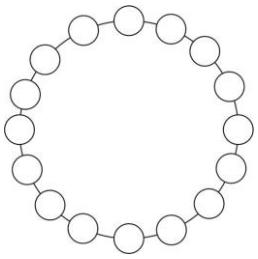

Fig. 1. Ring topology

2 Method

2.1 Support Vector Regression

Given training data $\{x_i,y_i\}$, $i = 1,...,l$; $x_i \in R^d$; $y_i \in R^d$ where x_i, y_i is input (vector) and output (scalar value as target). Other forms of alternative for bias to calculation $f(x)$ is can be build solution like bias as follows [14]:

$$\begin{aligned} b &= y_k - \varepsilon - w \bullet x_k \\ &= y_k - \varepsilon - \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) \Phi(x_i) \Phi(x_k) \\ &= y_k - \varepsilon - \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) k(x_i, x_k) \end{aligned} \quad (1)$$

x_i is support vector where $|\alpha_i - \alpha_i^*|$ isn't zero. Equation $f(x)$ can be write as follows:

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)(x \bullet x_i) + b \quad (2)$$

Lambda (λ) is scalar constant, with it's an augmented factor defined as follows [15]:

$$f(x) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)(K(x_i, x) + \lambda^2). \quad (3)$$

2.1.1 Sequential Algorithm for SVR

Vijayakumar has made tactical steps through the process of iteration to obtain the solution of optimization problems of any nature by way of a trade-off on the values of the weights x_i, or called α_i to make the results of the regression becomes closer to actual value. The step by step as follows:

1. Set initialization $\alpha_i = 0, \alpha_i^* = 0$, and get R_{ij} $[R]_{ij} = K(x_i, x_j) + \lambda^2$ (4)

for $i,j = 1,\ldots,n$

2. For each point, do looping, i=1 to n:

- $E_i = y_i - \sum_{j=1}^{n}(\alpha_j^* - \alpha_j)R_{ij}$ (5)
- $\delta\alpha_i^* = \min\{\max[\gamma(E_i - \varepsilon), -\alpha_i^*], C - \alpha_i^*\}$. (6)
- $\delta\alpha_i = \min\{\max[\gamma(-E_i - \varepsilon), -\alpha_i], C - \alpha_i\}$. (7)
- $\alpha_i^* = \alpha_i^* + \delta\alpha_i^*$. (8)
- $\alpha_i = \alpha_i + \delta\alpha_i$. (9)

3. Repeat step 2 until met stop condition.

Learning rate γ is computed from

$$\frac{\text{learning rate constant}(cLR)}{\max(\text{diagonal of kernel matrice})} \quad (10)$$

2.2 Particle Swarm Optimization

This algorithm defined solution as each particle for any problem in dimension space j. Then, it's extended by inertia weight to improves performance [16],[17]. Where x_{id}, v_{id}, y_{id} is position, velocity, and personal best position of particle i, dimension d, and $\hat{Y}_i$ is best position found in the neighborhood N_i. Each particle's neighborhood at ring topology consists of itself and its immediate two neighbours using euclidean distance.

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)\left[\hat{y}_{ij}(t) - x_{ij}(t)\right] \quad (11)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (12)$$

$v_{ij}(t)$, $x_{ij}(t)$ is velocity and position of particle i in dimension j=1,...n at time t, c_1 and c_2 are cognitive and social components, r_{1j} and r_{2j} are rand[0,1]. y_i and $\hat{y}$ is obtained by

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases} \quad (13)$$

$$\hat{y}(t+1) \in \{N_i \mid f(\hat{y}(t+1)) = \min\{f(x)\}, \ \forall x \in N_i\} \quad (14)$$

The inertia weight w, is follows equation

$$w = (w_{\max} - w_{\min})\frac{iter_{\max} - iter}{iter_{\max}} + w_{\max} \quad (15)$$

2.3.1 Binary PSO

Discrete feature space is set using binary PSO [18]. Each element of a particle can take on value 0 or 1. New velocity function is follows:

$$v'_{ij}(t) = sig\left(v_{ij}(t)\right) = \frac{1}{1+e^{-v_{ij}(t)}} \tag{16}$$

where $v_{ij}(t)$ is obtained from (11). Using (16), the position update changes to

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } r_{3j}(t) < sig\left(v_{ij}(t+1)\right) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $r_{3j}(t) \sim U(0,1)$.

2.3 Local best PSO SVR Model

2.3.1 Particle Representation

In this paper, SVR nonlinear is defined by the parameter C , ε , λ, σ , cLR. The particle is consisted of six parts: C, ε, λ, σ, cLR (continuous-valued) and features mask (discrete-valued). Table 1 shows the representation of particle i with dimension n_f+5 where n_f is the number of features.

TABLE I. PARTICLE I IS CONSISTED OF SIX PARTS: C, *E*, *Λ*, *E*, *CLR* AND FEATURE MASK

Continuous-valued					Discrete-valued
C	***ε***	***λ***	***Σ***	***cLR***	***Feature mask***
$X_{i,1}$	$X_{i,2}$	$X_{i,3}$	$X_{i,4}$	$X_{i,5}$	$X_{i,6}, X_{i,7}, \ldots, X_{i,nf}$

2.3.2 Objective Function

Objective function is used to measure how optimal the generated solution. There are two types of objective function: fitness and cost. The greater fitness value produced better solution. The lesser cost value produced better solution. In this paper, cost-typed is used as objective function because the purpose of this algorithm is to minimize the error. To design cost function, prediction accuracy and number of selected features are used as criteria. Thus, the particle with high prediction accuracy and small number of features produces a low prediction error with set weights value W_A = 95% and W_F =5% [7].

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| \tag{18}$$

$$error = \left(w_A \times MAPE\right) + \left(w_F \times \frac{\sum_{j=1}^{n_F} f_j}{n_F} \right) \tag{19}$$

where n is number of data, A_i is actual value and F_i is prediction value for data, f_j is value of feature mask.

2.3.3 Local Best PSO SVR Algoritm

This paper proposed local best PSO-SVR algorithm to optimize SVR parameter and input feature mask simultaneously. Fig. 2 illustrates local best PSO-SVR algorithm. Details of algorithm are described as follows:

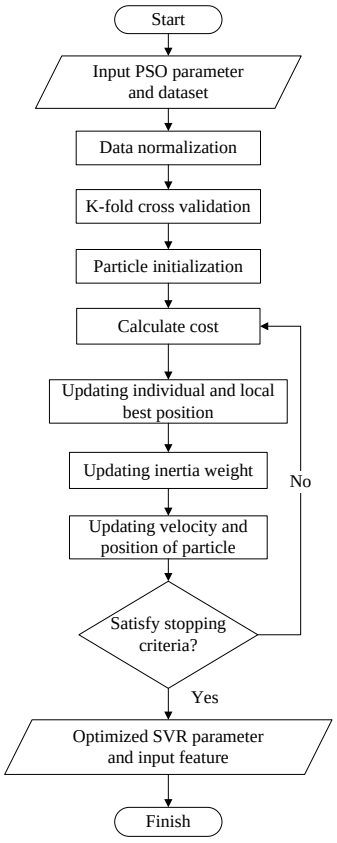

Fig. 2. Flowchart of local best PSO-SVR

1. Normalizing data using

$$x_n = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{20}$$

where x is the original data from dataset, x_{min} and x_{max} is the minimum and maximum value of original data, and x_n is normalized value.

2. Dividing data into k to determine training and testing data.
3. Initializing a population of particle.
4. Calculating cost by averaging error over k SVR training.
5. Updating individual and local best position of each particle.

6. Updating inertia weight.
7. Updating velocity and position of each particle.
8. If stopping criteria is satisfied, and then end iteration. If not, repeat step 2. In this paper, stopping criteria is a given number of iterations.

3 Application of Local Best PSO-SVR in Software Effort Estimation

3.1 Experimental settings

This study simulated 3 algorithms: local best PSO SVR, PSO SVR and T-SVR programmed using C#. For local best SVR simulation, we use the same parameters and dataset that is obtained from [19]. For software effort estimation, the inputs of SVR are Desharnais dataset [20]. The Desharnais dataset consists of 81 software projects are described by 11 variables, 9 independent variables and 2 dependent variables. For the experiment, we decide to use 77 projects due to incomplete provided features and 7 independent variables (*TeamExp*, *ManagerExp*, *Transactions*, *Entities*, *PointsAdjust*, *Envergure*, and *PointsNonAdjust*) and 1 dependent variable (*Effort*). The PSO parameters were set as in Table 2.

TABLE II. PSO PARAMETER SETTINGS

Number of fold	10
Population of particles	15
Number of iterations	40
Inertia weight(w_{max}, w_{min})	(0,6, 0,2)
Acceleration coefficient(c_1, c_2)	(1, 1,5)
Parameter searching space	C (0,1-1500), ε (0,001-0,009), σ (0,1-4), λ(0,01-3), *cLR* (0,01-1,75)

3.2 Experimental result

Fig. 3 illustrates the correlation between optimal cost and number of particle in 5 simulations. It showed that optimal cost is decreased while number of particle is being increased. From this chart, we can conclude that the more number of particles can provide more candidate solution so model can have more chance to select optimal solution. However, computing time is also increased because model spent much time to find solution among many particles and it is illustrated by Fig. 4. It happened because model must perform solution searching in many particles and it compromised computing time. In the experiment, we discovered that 20 particles could obtain the most optimal cost, but we can't use it as optimal parameter since spending much computing time. Thus, we decided to use 15 particles under consideration that it has less computing time but still obtain optimal cost.

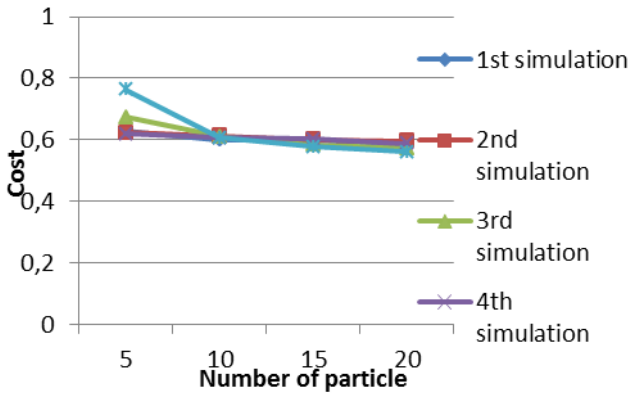

Fig. 3. Comparison of number of particle

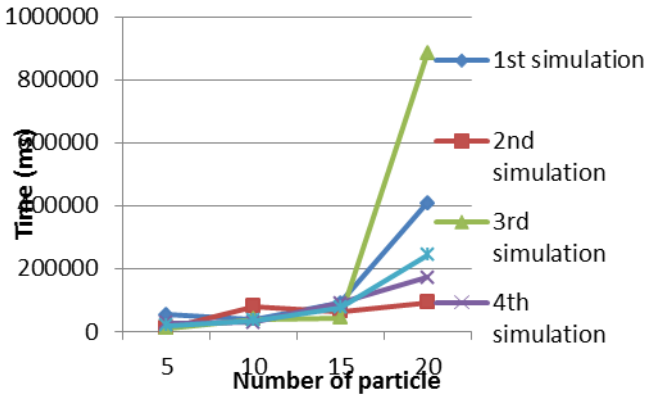

Fig. 4. Comparison of computing time

Fig. 5 illustrates the correlation between optimal cost and number of iteration in 5 simulations. It showed that optimal cost is decreased while number of iteration is being increased. For the example, in 4th simulation, optimal cost remained steady until 4th iteration and move down until 8th iteration. From 8th iteration until 40th iteration, optimal cost didn't perform any change and it means that model converged and found optimal solution.

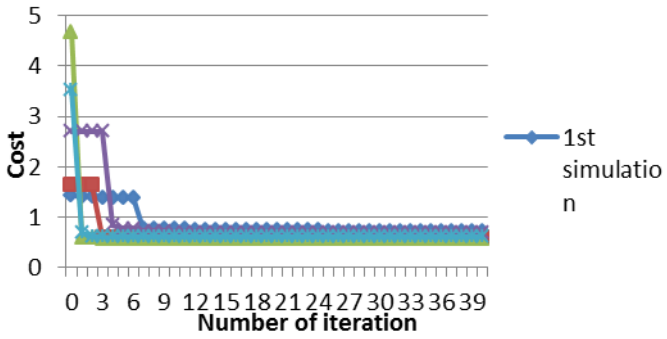

Fig. 5. Convergence during process

Table 3 showed the comparison of the experiment results. The experiments showed that the proposed model outperforms T-SVR and PSO-SVR in optimizing SVR. Local best PSO SVR obtained lowest error among three models. It is observed that PSO-SVR spent less computing time because of fast convergence. Local best PSO-SVR model has slow convergence because it finds optimal solution in its neighborhood.

TABLE III. COMPARISON OF PREDICTION RESULTS

Model	Time (ms)	Optimal (C, ε, σ, cLR, λ)	Selected features	Error
Local best PSO SVR	91.638	0,1000, 0,0063, 0,2536, 0,0100, 0,0100	2 (*Entities* and *Envergure*)	0,5161
PSO-SVR	62.610	1500, 0,09, 0,1, 0,01, 3	2 (*PointsAdjust* and *Envergure*)	0,5819
T-SVR	677.867	1055,3338, 0,0686, 0,1557, 0,1514, 0,2242	4 (*TeamExp*, *ManagerExp*, *Entities*, and *PointsAdjust*)	0,6086

4 Conclusion

This paper examined the implementation of local best particle swarm optimization for optimal feature subset selection and SVR parameters optimization in the problem of software effort estimation. In our simulations, we used Desharnais dataset. We compared our results to PSO-SVR and T-SVR. From the experiment results, using local best version can improve performance of PSO. For further research, we suggest to use different topologies e.g. Von Neumann, pyramid, wheel and four clusters, to give more perspectives about effect of social network structures to PSO for selecting optimal number of feature and optimizing SVR parameters combination in the

software effort estimation problem. Hybridizing with other heuristic algorithms such as simulated annealing becomes an option to improve the performance of PSO [19][21].

References

[1] T. Standish Group, "Chaos Manifesto 2013 Think Big, Act Small," 2013.

[2] R. Agarwal, M. Kumar, Yogesh, S. Mallick, R. M. Bharadwaj, and D. Anantwar, "Estimating Software Projects," ACM SIGSOFT Software Engineering Notes, vol. 26, no. 4, pp. 60–67, 2001.

[3] D. Zhang and J. J. Tsai, "Machine Learning and Software Engineering," Software Quality Journal, vol. 11, no. 2, pp. 87–119, 2003.

[4] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort," IEEE Transactions on Software Engineering, vol. 21, no. 2, pp. 126–137, 1995.

[5] V. N. Vapnik, "An Overview of Statistical Learning Theory," IEEE Transactions on Neural Networks, vol. 10, no. 5, pp. 988–999, 1999.

[6] H. Frohlich, O. Chapelle, and B. Scholkopf, "Feature Selection for Support Vector Machines by Means of Genetic Algorithm," in Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence, 2003, pp. 142–148.

[7] Y. Guo, "An Integrated PSO for Parameter Determination and Feature Selection of SVR and Its Application in STLF," in Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, 12-15 July 2009, 2009, no. July, pp. 12–15.

[8] W. Wang, Z. Xu, W. Lu, and X. Zhang, "Determination of The Spread Parameter in the Gaussian Kernel For Classification and Regression," Neurocomputing, vol. 55, pp. 643–663, 2003.

[9] P. Braga, A. Oliveira, and S. Meira, "A GA-based Feature Selection and Parameters Optimization for Support Vector Regression Applied to Software Effort Estimation," in Proceedings of the 2008 ACM Symposium on Applied Computing, 2008, pp. 1788–1792.

[10] G. Hu, L. Hu, H. Li, K. Li, and W. Liu, "Grid Resources Prediction With Support Vector Regression and Particle Swarm Optimization," 3rd International Joint Conference on Computational Sciences and Optimization, CSO 2010: Theoretical Development and Engineering Practice, vol. 1, pp. 417–422, 2010.

[11] M. Jiang, S. Jiang, L. Zhu, Y. Wang, W. Huang, and H. Zhang, "Study on Parameter Optimization for Support Vector Regression in Solving the Inverse ECG Problem," Computational and Mathematical Methods in Medicine, vol. 2013, pp. 1–9, 2013.

[12] A. P. Engelbrecht, Computational Intelligence: An Introduction, 2nd ed. West Sussex: John Wiley & Sons Ltd, 2007.

[13] R. Mendes, J. Kennedy, and J. Neves, "The Fully Informed Particle Swarm: Simpler, Maybe better," IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 204–210, 2004.

[14] A. J. Smola and B. Scholkopf, "A Tutorial on Support Vector Regression," Statistics and Computing, vol. 14, no. 3, pp. 199–222, 2004.

[15] S. Vijayakumar and S. Wu, "Sequential Support Vector Classifiers and Regression," in Proceedings of International Conference on Soft Computing (SOCO '99), 1999, vol. 619, pp. 610–619.

[16] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in Neural Networks, 1995. Proceedings., IEEE International Conference on, 1995, vol. 4, pp. 1942–1948.

[17] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," in 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), 1998, pp. 69–73.

[18] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the World Miulticonference on Systemics, Cybernetics and Informatics*, 1997, pp. 4104–4109.

[19] D. Novitasari, I. Cholissodin, and W. F. Mahmudy, "Hybridizing PSO with SA for Optimizing SVR Applied to Software Effort Estimation", *Telkomnika (Telecommunication Computing Electronics and Control)*, 2015, vol. 14, no. 1, pp. 245-253.

[20] J. Sayyad Shirabad and T. J. Menzies, "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository. [Accessed: 05-Mar-2015].

[21] Mahmudy, WF 2014, 'Improved simulated annealing for optimization of vehicle routing problem with time windows (VRPTW)', *Kursor*, vol. 7, no. 3, pp. 109-116.