Nonlinear Regression in Dynamic Environments using Particle Swarm Optimization

Cry Kuranga[1][0000-0002-2801-6974] and Nelishia Pillay[2][0000-0003-3902-5582]
Department of Computer Science University of Pretoria, Lynnwood Road, Hillcrest, Pretoria, South Africa, 0002 (Phone 012 420 5242)
[1]kurangacry@gmail.com, [2]npillay@cs.up.ac.za

Abstract. This paper extends a PSO-based nonlinear regression technique to dynamic environments whereby the induced model dynamically adjusts when an environmental change is detected. As such, this work hybridizes a PSO designed for dynamic environments with a least-squares approximation technique to induce structurally optimal nonlinear regression models. The proposed model was evaluated experimentally and compared with the dynamic PSOs, namely multi-swarm, reinitialized, and charged PSOs, to optimize the model structure and the regression parameters in the dynamic environment. The obtained results show that the proposed model was adaptive to the changing environment to yield structurally optimal models which consequently, outperformed the dynamic PSOs for the given datasets.

Keywords: Dynamic PSO, Least-squares, Nonlinear Regression, Dynamic Environments.

1 Introduction

Regression models can be classified as either linear or nonlinear. Nonlinear models can be induced by either classical techniques such as Gauss-Newton [1] and Levenberg-Marquardt methods [2] or machine learning techniques such as evolutionary computation [3], [4], [5] and artificial neural networks [6], [7].

Nonlinear regression-based prediction, in static environments, has been successfully applied in the literature where a static prediction model is constructed once and then used to predict for instances not used during training. The main objective is to minimize the nonlinear least-squares error. Therefore, nonlinear regression-based prediction can be considered as an optimization problem where the optimal parameters can be estimated by either classical or heuristic optimization techniques. However, classical techniques are usually trapped in local minima [8]. As such, metaheuristics have been considered as an alternative [5], [3], [4].

Particle swarm optimization (PSO) has been successfully applied to nonlinear regression problems, in static environments, w.r.t accuracy [9], [10]. However, the structure of the approximator needs to be optimized as well. Usually, real-world nonlinear regression problems are dynamic. As such, the objective function in a dynamic environment tends to change, which results in changes in the search space structure and

the position of optima. Therefore, the performance of the prediction model constructed using the past environment is bound to deteriorate. Thus, making a continuous adaptation of the prediction model becomes a necessity. However, a standard PSO in a changing environment is prune to outdated memory problem and diversity loss [11].

This work aims to extend a PSO-based nonlinear regression technique to dynamic environments whereby the induced model dynamically adjusts when an environmental change is detected. As such, this work hybridizes a PSO designed for dynamic environments with a least-squares approximation technique to induce structurally optimal nonlinear regression models in dynamic environments. This hybridization decreases the performance deterioration that usually results from the environmental changes and consequently, improves the algorithm's performance. Also, this work evaluates experimentally and compared with dynamic PSO algorithms, namely multi-swarm, reinitialized, and charged PSOs, to optimize the model structure and the regression parameters in the dynamic environment. Dynamic PSO algorithms are PSO variants that can adapt in dynamic environments.

This paper is structured as follows: Section 2 discusses related work. Section 3 discusses the proposed model whereas Section 4 discusses the experimental setup and present the results. Section 5 provides a conclusion to the paper.

2 Related Work

Regression analysis is a statistical-based model induction technique that models a relationship between variables (quantitative), ideal to predict a selected variable from one or more other variables [12]. Regression analysis uses equations to describe the given dataset whereby a regression model consists of the following components [13]: a response variable(s), i.e., a real-world output, y; a vector of predictor variables, i.e., an input vector $\boldsymbol{x} = x_1; x_2; \dots; x_n;$ and an unknown parameter, θ, which can be a vector or scalar.

Given that $\boldsymbol{x}$ is an input vector and y is a real-world output, a nonlinear model between $\boldsymbol{x}$ and y is of the form:

$$y = f(x;\theta) + \varepsilon \qquad (1)$$

where ε is a random error. A process of fitting the best approximation to a dataset of $n = |N|$ data points $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$ is commonly referred to as least-squares approximation [14] [15]. Least-squares approximation minimizes the least-squares error, E_{ss}, to an arbitrary set of m data points:

$$E_{ss} = \sum_{i=1}^{m} [y_i - \dot{y}_i]^2$$

where $\dot{y}_i$ is the predicted output and y_i is the target value, e.g.

$$\dot{y}_i = \tau_0 + \tau_1 x_i + \cdots + \tau_{p-1} {x_i}^{p-1} + \tau_p {x_i}^p$$

$$= \sum_{j=0}^{p} \tau_j {x_i}^j \qquad (2)$$

where p is the maximum number of terms. The determination of coefficients $(\tau_{0,} \dots, \tau_n)$ is realized through solving the linear system:

$$\boldsymbol{y} \approx X\tau$$

where $\tau^T = [\,\tau_0,\ \tau_1, \dots \tau_{p-1,}\tau_p]$ and $y^T = [\,y_0,\ y_1, \dots y_{p-1,}y_p]$. Therefore, the solution to the least-squares problem can be obtained by solving the overdetermined system:

$$(X^T X)\tau = X^T \boldsymbol{y}$$

Least-squares problems are usually solved using numerous techniques such as normal equation, singular value decomposition and QR decomposition technique. The normal equation technique is very fast though the least precise whereas singular value decomposition is the most precise though the slowest. The QR decomposition technique strikes the balance between precision and computational load [15]. QR decomposition can be computed using several methods such as Givens rotations, Householder transform, and Gram-Schmidt.

The implementation of PSO, in a static environment, for regression problems has yielded favorable results [9]**,** [10]. However, since PSO in a changing environment is prune to outdated memory problem and diversity loss [11], few PSO variants that can adapt in dynamic environments were selected based on the classification given in [16]: a memory scheme - reinitialized PSO; a multi-population scheme - multiPSO; and a diversity maintenance scheme - charged PSO. A brief description of the selected algorithms is provided below.

A charged PSO proposed by Blackwell and Branke, referred to as quantum PSO (QPSO), is based on the model of an atom [17], [18], [19]. A swarm consists of both quantum particles and non-quantum particles. For quantum particles, instead of using the position equation as in the standard PSO, the position of each quantum particle is determined by a probability distribution. To preserve the swarm diversity, the quantum particles are randomized at each iteration. The non-quantum particles behave like standard PSO particles which use velocity and position equations to improve the current solution. As such, quantum particles search for new solutions [17].

Multi-swarm PSO implements multiple populations, each optimizing a single solution from the set of solutions [17]. The algorithm keeps the swarm diverse by using anti-convergence methods and repulsion. The convergence of all sub-swarms is determined by a convergence radius whereby if all sub-swarm are within r_{excl} then one subswarms, usually the one with the worst fitness, is reinitialized.

In reinitialized PSO, the swarm or part of it is randomly re-initialized within the search space when a change in the environment occurs to enhance diversity [20]. Usually, the best particles are maintained within their neighborhood to monitor the best-known positions. If a change is detected, the particle velocity is reset and the particle's current position becomes the particle's best position**.** As such, particles are discouraged to be attracted to their former position. In this work, a nonlinear regression model designed for dynamic environments is proposed that hybridize a QR decomposition, computed using Gram-Schmidt technique and a dynamic PSO algorithm.

3 Particle Swarm Optimization in Regression Analysis

The proposed dynamic PSO-based nonlinear regression model consists of a QR decomposition technique to determine the coefficients of the model and a dynamic PSO to induce an optimal model structure that can adapt whenever a change in the environment occurs.

Considering a QR decomposition technique, the predicted output, $\dot{y}_i$, in Equation (2), when the dimensionality of the input space, d, is taken into consideration, can be rewritten as [5]:

$$\dot{y}_i = \sum_{\sum_{j=1}^{d} \beta_j = 0}^{p} \left(\tau(\beta_1, \beta_2, \dots, \beta_d) \prod_{q=1}^{d} x_{i,q}^{\beta_d} \right) \qquad (3)$$

where $\tau(\beta_1, \beta_2, \dots, \beta_d)$ is a real-valued coefficient and β_d is the order of attribute $x_{i,q}$. Considering $d = 2$ and $p = 2$, Equation (3) lets the representation of function such as:

$$\dot{y}_i = \tau_{(0,0)} + \tau_{(1,0)} x_{i,1} + \tau_{(0,1)} x_{i,2} + \tau_{(1,1)} x_{i,1} x_{i,2} + \tau_{(2,0)} x_{i,1}^2 + \tau_{(0,2)} x_{i,2}^2$$

As such, QR decomposition determines the value of the coefficients, $\tau(\beta_1, \beta_2, \dots, \beta_d)$. Therefore, the dynamic PSO is tasked to determine only the optimal model structure.

Each particle in dynamic PSO is a representation of Equation (3) and consists of unique, term-coefficient mappings from a set, S: [5]

$$S = \left\{ (t_0 \rightarrow \tau_o), \dots, \left(t_{n_p} \rightarrow \tau_{n_p} \right) \right\} \qquad (4)$$

where $\tau_j, j \in \{0, \dots, n_p\}$ is a real-valued coefficient and n_p is the maximum number of terms. Each term, t_j, consists of a set, T, of unique, variable-order mappings e.g.

$$T = \left\{ \left(x_{i,1} \rightarrow \beta_1 \right), \dots, \left(x_{i,d} \rightarrow \beta_d \right) \right\}$$

where $x_{i,j}, j \in \{1, \dots, n\}$ is an input variable-integer representation, n is the number of input variables and β_j is a natural-valued order. The coefficients of term-coefficient mappings are determined by reducing $\boldsymbol{y} \approx X\tau$. Algorithm 1 summarizes a dynamic PSO-based nonlinear regression technique.

Algorithm 1 Dynamic PSO-based Nonlinear Regression

BEGIN
  Initialize particles using Eqn (3)
  DO
    Run *n* iterations of the dynamic PSO algorithm
    IF an environment_change is detected
      Update the coefficients of term-coefficient mappings in each particle by reducing $\boldsymbol{y} \approx Xr$.
    END
  UNTIL termination condition satisfied;
END

3.1 Fitness Function

The adjusted coefficient of determination, R_a^2, is used to measure the fitness of each particle and is defined as:

$$R_a^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_{I,i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \times \frac{n-1}{n-k}$$

where the predicted output of particle I for the pattern i is $y_{I,i}$, the target output is y_i, n is a size of data patterns and k is the number of coefficients. The R_a^2 penalizes a model that has a larger number of coefficients, k. Thus, the objective of R_a^2 is to minimize the model's architecture, whereas maximizing the correlation between the dataset and the induced model.

3.2 Detecting Environmental Changes

For an algorithm to efficiently optimize in a dynamic environment, there is a need for the algorithm to detect an environmental change [21]. Various indicators can be used to detect an environmental change in a dynamic environment such as the time-averaged best performance or the deterioration of the population performance [22]. In this work, a simple and efficient method to detect environment change used in [23] was adopted that uses the *personal Best* position of each particle which is re-evaluated before being updated. As such, fitness deterioration implies that an environmental change had occurred.

4 Experimental Setup and Results

In this section, the multi-swarm, reinitialized and charged PSOs and the proposed model were experimentally evaluated. All experiments were implemented in MATLAB programming environment [24] on an Intel Core i7 processor (3.1 GHz) desktop with 16 GB of memory running on a Linux Centos 7 system. For reinitialized PSO, 50% of the swarm was reinitialized when an environmental change was detected. The following PSO parameters in the literature were used $c_1 = c_2 = 1.496180,\ \omega = 0.729844$, $r_{excl} = quantum\_radius = 2$, $swarm\_size = 50$ and QR decomposition: $n_p = \beta_j = 10$. For each experiment, 1000 iterations were executed for each algorithm.

The training algorithms were categorized into QR_PSOs and nonQR_PSOs. The QR_PSOs consist of dynamic PSOs namely, charged PSO (QPSO), multi-swarm PSO (mPSO) and re-initialized PSO (rePSO) that implemented the proposed Algorithm 1 and therefore, referred to as QR_QPSO, QR_mPSO and QR_rePSO respectively. To benchmark, the performance of the proposed model, nonQR_PSOs (QPSO, mPSO and rePSO) were implemented in all experiments. The initial swarm for each algorithm was initialized using Equation (3).

To simulate dynamic environments, a windowing technique was used. A sliding window of analysis was set to ws data patterns. A data pattern consists of the inputs

and target output for the given dataset. Each sliding window was split to training and generalization datasets using the ratio 4:1 respectively.

4.1 Performance Measure

The off-line performance was used to evaluate the performance of the algorithm, computed at each time step as the best fitness found so far [22]. A total of 30 independent runs were executed on each experiment for each algorithm and then averaged. The Mann-Whitney U test was performed, at a significance level of 0.05, to determine if there was a statistically significant difference between the mean fitness values of the training algorithms for each experiment [25]. A test was performed for the algorithms' mean fitness values, μ_1 and μ_2 , whereby $H_0 : \mu_1 = \mu_2$, and $H_1 : \mu_1 \neq \mu_2$. These tests were performed for every combination of algorithms and all problems. The number of wins and losses for each algorithm was determined using U-values. The overall performances were ranked based on the difference between wins and losses of each algorithm.

4.2 Dataset

Real-world nonlinear regression datasets in dynamic environments enable to evaluate the performance of the induced model in real-world conditions. However, the existence of a real drift in the data is unknown or if the drift exists, it may be unknown when exactly it occurs. As such, it becomes difficult to have an in-depth analysis of the behavior of the predictive models. Therefore, an artificially generated dataset with induced drifts becomes favorable.

a) Benchmark Dataset

Auto-generated datasets of 10 000 patterns with 100 timesteps were used in the experiments conducted in this work on different types of change period. The datasets were generated using the benchmark nonlinear Bennett5 function computed as [26]:

$$f(x, \theta) = \theta_1 + (\theta_2 + x)^{-\frac{1}{\theta_3}}$$

The starting values for Bennett5 function were provided in the literature. An environmental change is simulated by adding drift to each parameter, $\theta_i = \theta_i + \delta\sigma$ where δ is the drift and σ is the probability of altering the direction of change. The following equation was used to simulate the drift:

$$f(\delta) = 0.6\delta^2 + 0.02\delta + 0.01$$

The impact of the drifts was smaller at the beginning and then improves along with an increase in the frequency (f).

A sliding window of size of 100 patterns slides from one timestep, which consists of 100 patterns, to the next. The change period occurs at severities and frequencies of 1 to 5. The severity of change determines the probability of altering the direction of change, σ, where a value of 5 implies that the reverse direction of change is certain at

each change period. The timestep at which the change occurs is determined by the frequency of change and was computed as:

$$changePoint = \frac{f}{10} \times T$$

where T is the total number of iteration and f is the frequency. A high value of f implies that fewer changes were occurring to the dataset.

b) Electricity Pricing

A real-world dataset, Electricity pricing that consists of 27552 data patterns, was also implemented in this work [27]. The Electricity pricing dataset was built on the electricity market in the Australian state of New South Wales. This dataset exhibits both long-term regular changes happening as a result of seasonal changes and short-term irregular changes happening as a result of weather fluctuations.

To determine the electricity price, the current electricity demand is matched with the combination of all available power stations with the least expensive electricity. The task of the proposed model was to induce a predictive model that determines the electricity price from the given parameters.

The sliding window size (ws) was set to 100 data patterns. The training algorithm requires 275 slides to traverse the complete dataset. Given that 100 iterations were executed before the sliding window slides. Therefore, for the training algorithm to traverse the complete dataset, 27 600 iterations were executed.

4.3 Results

The results were analyzed using mean squared errors (E_{MS}) and adjusted coefficient of determination (R_a^2) on training and generalization for different severities and frequencies. The p-values corresponding to the comparison of the training algorithms on $E_{MS}T$ (training) and $E_{MS}G$ (generalization) for 30 independent runs were reported in Appendix 1 where $p \leq 0.0001$ was recorded as 0.0001 for convenience. Table 1 presents the average (avgs) and standard deviation (SD at the 95% confidence interval) of Bennett5 dataset for R_a^2 and E_{MS} on training and generalization for each algorithm.

The results presented in Table 1 show that QR_QPSO obtained the best R_a^2 on both training and generalization whereas mPSO obtained the worst performance on generalization and rePSO on training. As presented in Table 1, QR_PSOs yielded improved performances compared to nonQR_PSOs on both R_a^2 and E_{MS}, except for QPSO on R_a^2. This performance improvement could have been attributed by the capability of the proposed technique to track and adapt the nonlinear regression model as the environment changes. Also, the value of R_a^2 above 0.7 suggests structurally optimal models generated by QR_PSOs. Considering QR_PSOs only, QR_QPSO exhibit superior performance whereas QR_rePSO exhibit the worst performance.

The averages for R_a^2 and EMS on generalization per frequency and severity are graphically illustrated in Figure 1. As illustrated in Figure 1, the performance of all training algorithms improved as the frequency increased on both R_a^2 and E_{MS}. However, QR_PSOs exhibit superior performance evident with high R_a^2 and low E_{MS}. Also,

QR_QPSO outperformed all other algorithms on both R_a^2 and E_{MS} for all frequencies and severities

Table 1. Avgs and SD for R_a^2 and E_{MS} for each algorithm

Algorithm	Training		Generalization	
	R^2	E_{MS}	R^2	E_{MS}
QR_QPSO	0.7075±0.0731	0.3106±0.0142	0.8236±0.0598	0.2183±0.0621
QPSO	0.4227±0.1782	0.3103±0.0097	0.7598±0.0699	0.4504±0.0954
QR_rePSO	0.6169±0.0901	0.3123±0.0099	0.8059±0.0762	0.4043±0.1150
rePSO	0.0769±0.0110	0.3507±0.0223	0.4237±0.0452	0.4992±0.0774
QR_mPSO	0.6407±0.0583	0.3141±0.0132	0.8121±0.0649	0.4807±0.0849
mPSO	0.0912±0.0973	0.3228±0.0192	0.4068±0.0527	0.4271±0.0326

.

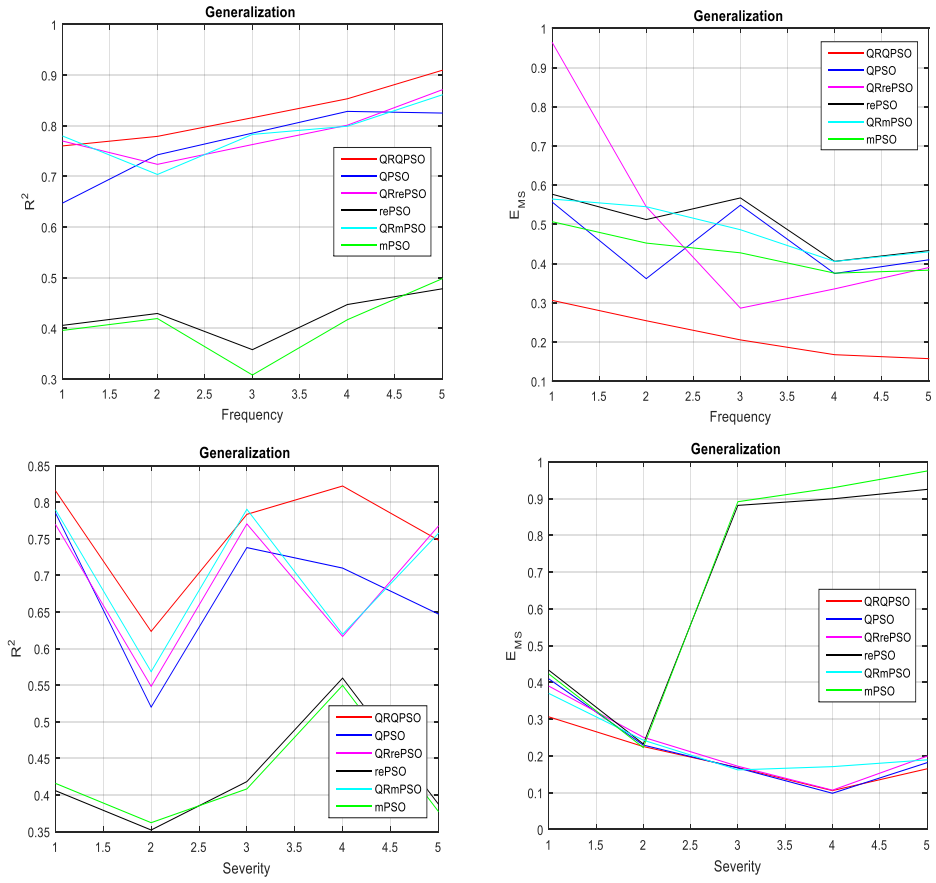

Fig. 1. Averages for R_a^2 and E_{MS} on Generalization per Frequency and Severity

The average ranks obtained on training algorithms for R_a^2 and E_{MS} on both training and generalization for the given frequencies and severities were illustrated in Figure 2 and

Figure 3. As illustrated in Figure 2, rePSO and mPSO exhibit the worst performance on both R_a^2 and E_{MS}. On training, QR_rePSO and QR_mPSO exhibit improved performance on E_{MS}. On generalization, QR_QPSO outperformed all other training algorithms on both R_a^2 and E_{MS}. Generally, as the frequency increased, the performance of nonQR_PSOs deteriorated on both R_a^2 and E_{MS} especially for QPSO.

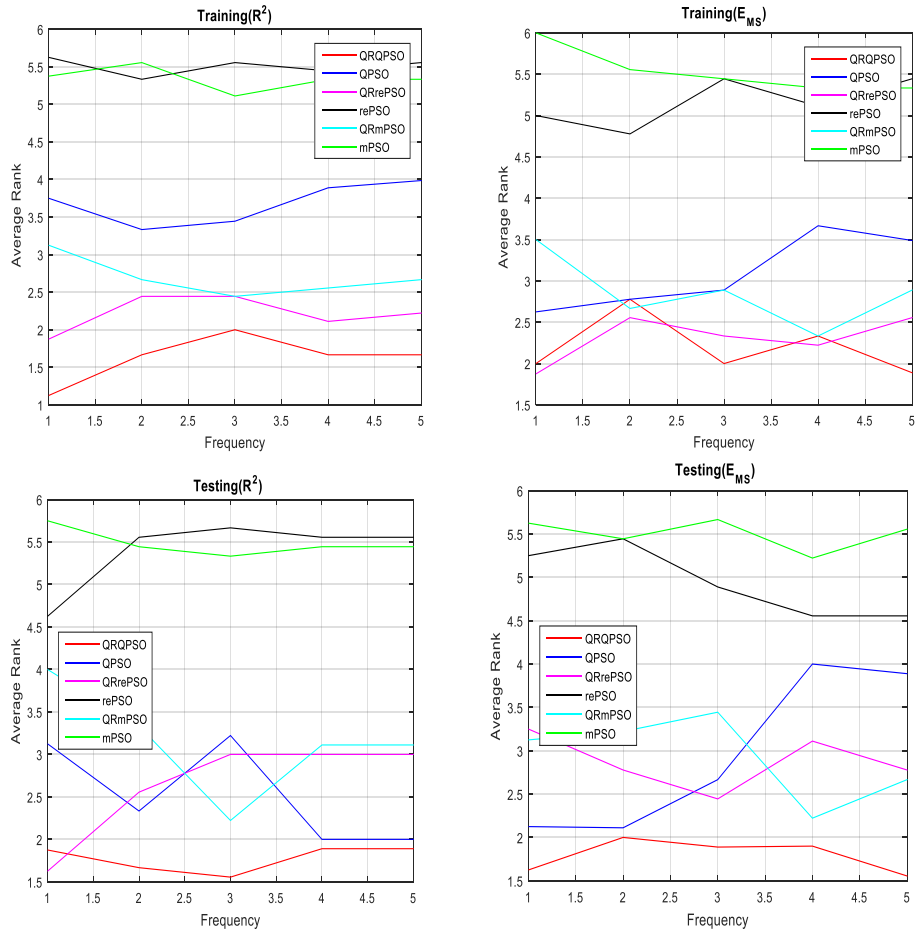

Fig. 2. Ranks of R_a^2 and E_{MS} per frequency on training and testing

As illustrated in Figure 3, QR_QPSO also outperformed all other algorithms on both R_a^2 and E_{MS} whereas rePSO and mPSO exhibit the worst performance on both R_a^2 and E_{MS}. The performance of QR_mPSO improved as the severity increased to outperform QR_rePSO on generalization on both R_a^2 and E_{MS}. This performance improvement for QR_mPSO suggests the improved adaptive traits of QR_mPSO under severe changing environment.

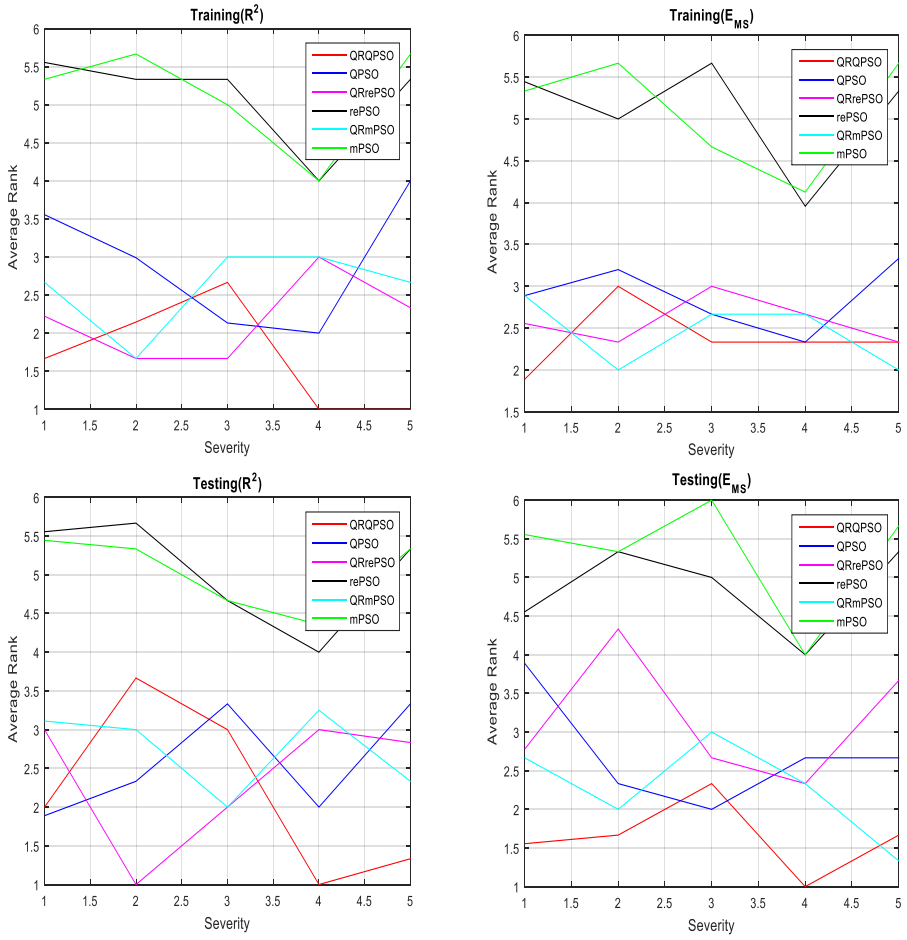

Fig. 3. Ranks of R_a^2 and E_{MS} per severity on training and testing

Generally, the performance of nonQR_PSO on both R_a^2 and E_{MS} deteriorated as the severity increased whereas QR_PSOs exhibit an improved performance as the severity increased which suggests improved adaptive traits to a changing environment.

Table 2 presents the average (avgs) and standard deviation (SD at the 95% confidence interval) for R_a^2 and E_{MS} on training and generalization for each algorithm on Electricity pricing dataset.

As observed in Table 1, the results presented in Table 2 show that QR_QPSO obtained the best R_a^2 on both training and generalization whereas rePSO obtained the worst performance on generalization and training. Also, QR_PSOs yielded improved performances compared to nonQR_PSOs on both R_a^2 and E_{MS}. As already explained, the performance improvement could have been attributed by the capability of the proposed technique to track and adapt the nonlinear regression model as the environment changes.

Table 2. Avgs and SD for R_a^2 and E_{MS} for Electricity Pricing Dataset

Algorithm	Training		Generalization	
	R^2	E_{MS}	R^2	E_{MS}
QR_QPSO	0.8874±0.1794	0.0001±0.0004	0.8097±0.2979	0.0011±0.0005
QPSO	0.8646±0.1630	0.0001±0.0004	0.7316±0.3697	0.0041±0.0010
QR_rePSO	0.8671±0.1496	0.0001±0.0004	0.7744±0.3384	0.0032±0.0011
rePSO	0.6670±0.1931	0.0003±0.0012	0.7259±0.3234	0.0043±0.0007
QR_mPSO	0.8694±0.1712	0.0001±0.0003	0.7837±0.2649	0.0028±0.0008
mPSO	0.7108±0.1253	0.0002±0.0009	0.7305±0.3234	0.0042±0.0009

5 Conclusion

The obtained results suggest the capability of the proposed dynamic PSO-based nonlinear regression technique to track and adapt the induced model as the environment changes. Therefore, yield improved performance and consequently, outperformed the dynamic PSOs on the given datasets. The hybridization of the dynamic PSOs with QR decomposition technique indeed decreased the performance deterioration of the induced model that resulted from the environmental changes. The obtained values of R_a^2 suggests that the dynamic PSO-based nonlinear regression technique induced structurally optimal nonlinear regression models.

Future work could extend the dynamic PSO-based nonlinear regression technique to induce the nonlinear regression models using the most recent and relevant data points from the presented dataset by excluding the irrelevant data points provided in the data window. Also, to perform a comparative study of dynamic PSO-based nonlinear regression technique with non-PSO machine learning approaches and neural networks.

References

1. R.A. Gray, P.D. Docherty, L.M. Fisk, and R. Murray, "A modified approach to objective surface generation within the Gauss-Newton parameter identification to ignore outlier data points," *Biomedical Signal Processing and Control,* vol. 30, p. 162-169, (2016).
2. "Basics on Continuous Optimization," November (2019). [Online]. Available: http://www.brnt.eu/phd/node10.html.
3. P. Erdoğmuş and S. Ekiz "Nonlinear Regression using Particle Swarm Optimization and Genetic Algorithm," *International Journal of Computer Applications,* vol. 153, no.6, (2016).
4. G. Potgieter and A.P. Engelbrecht, "Genetic Algorithm for Structurally Optimisation of Learned Polynomial Expressions," *Applied Mathematics and Computation,* vol. 186, p. 1441-1466, (2007).
5. T. Özel, and Y. Karpat, "Identification of constitutive material model parameters for high-strain rate metal cutting conditions using evolutionary computational algorithms," *Materials and Manufacturing Processes,* vol. 22, no. 5, p. 659-667, (2007).
6. K. Hornik, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, p. 359–366, (1989).

7. U. Atici, "Prediction of the strength of mineral admixture concrete using multivariable regression analysis and an artificial neural network.," *Expert Systems with Applications,* vol. 38, no. 8, p. 9609-9618, (2011).
8. Z. Lu, C. Yang, D. Qin, Y. Luo, and M. Momayez, "Estimating ultrasonic time-of-flight through echo signal envelope and modified Gauss-Newton method," *Measurement,* vol. 94, p. 355-363, (2016).
9. S. Cheng, C. Zhao, J. Wu, and Y. Shi, "Particle Swarm Optimization in Regression Analysis: A Case Study," *Lecture Notes in Computer Science ,*(2013).
10. S.M. Abdullah, A. I. M. Yassin, and N. M. Tahir, "Particle Swarm Optimization and Least Squares Estimation of NARMAX," *ARPN Journal of Engineering and Applied Sciences,* vol. 10, no. 22, (2015).
11. T.M Blackwell, "Particle Swarm Optimization in Dynamic Environments," *Evolutionary Computation in Dynamic and Uncertain Environments,* vol. 51, p. 29-49, (2007).
12. N.R. Draper, and H. Smith, Applied Regression Analysis, vol.326, John Wiley & Sons, (1998).
13. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed., Springer Series in Statistics, Springer, (2009).
14. J.B. Fraleigh, and R.A. Beauregard, Linear Algebra, 3. edition, Ed., Upper Saddle River, NJ: Addison-Wesley Publishing Company, (1995).
15. S. M. Stigler, "Gauss and the Invention of Least Squares," *Ann. Stat.,* vol. 9, no. 3, p. 465–474, (1981).
16. S. Yang and X. Yao, "A Comparative Study on Particle Swarm Optimization in Dynamic Environments," in *Evolutionary Computation for DOPs*, Berlin Heidelberg, Springer-Verlag, pp. 109-136, (2013).
17. T. Blackwell and J. Branke, "Multi-swarm Optimisation in Dynamic Environments," *Applications of Evolutionary Computing,* vol. 3005, pp. 489-500, (2004).
18. T.M. Blackwell and P.J. Bentley, "Don't Push Me! Collision-Avoidance Swarms," *Proceedings of the IEE Congress on Evolutionary Computation,* vol. 2, p. 1691-1696, (2002).
19. T.M. Blackwell and P.J. Bentley, "Dynamic Search with Charged Swarms," *Proceedings of the Genetic and Evolutionary Computation Conference,* vol. 2, p. 19-26, 2002.
20. T.M. Blackwell, and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments.," *IEEE Transactions on Evolutionary Computation,* vol. 10, no. 4, p. 459–472, (2006).
21. H. Richter, "Detecting change in dynamic fitness landscapes," *in Proc. Congr. Evol. Comput,* p. 1613–1620 ,(2009).
22. J. Branke, Evolutionary Optimization in Dynamic Environments, Norwell, MA: Kluwer, (2002).
23. D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Trans. Evol. Comput.,* vol. 10, no. 4, p. 440–458, (2006).
24. "Mathworks," MATLAB, [Online]. Available: www.mathworks.com.
25. P.E. McKnight, and J. Najab, "Issues with Performance Measures for Dynamic Multi-objective Optimization," *Proceedings of IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments,* p. 17-24, (2013).
26. L. Bennett, L. Swartzendruber, and H. Brown, "Superconductivity Magnetization Modeling," *NIST, (*1994).
27. M. Harries, "Splice-2 comparative evaluation: Electricity pricing. Technical Report UNSW-CSE-TR-9905," Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sydney 2052, Australia, (1999).