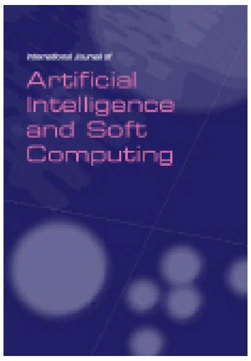
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Artificial Intelligence and Soft Computing

ISSN online: 1755-4969 - ISSN print: 1755-4950
https://www.inderscience.com/ijaisc

Optimising SVR for epidemiological predictions: a case study on COVID-19 mortality in Japan

Edward R. Sykes, Yuan Wang

DOI: 10.1504/IJAISC.2024.10068405

Article History:

Received:	27 May 2024
Last revised:	21 September 2024
Accepted:	29 October 2024
Published online:	16 December 2024

Copyright © 2024 Inderscience Enterprises Ltd.

Optimising SVR for epidemiological predictions: a case study on COVID-19 mortality in Japan

Edward R. Sykes*

School of Computer Science,
University of Guelph,
Guelph, Ontario, Canada
Email: sykes@uoguelph.ca
*Corresponding author

Yuan Wang

Centre for Applied AI,
Sheridan College,
Oakville, Ontario, Canada
Email: yuan.wang1@sheridancollege.ca

Abstract: This study enhances support vector regression machine (SVR) for COVID-19 mortality forecasting in Japan using three particle swarm optimisation (PSO) variants. Our main contributions include: 1) achieving superior model performance, notably with the fast convergence PSO-SVR variant, which outperforms existing models with an R-Squared value of 0.717; 2) demonstrating consistent and improved prediction accuracy across various PSO variants; 3) establishing the potential of our methods for broader applications beyond epidemiological modelling. Our findings, significantly advancing the accuracy and efficiency of predictive analytics in this domain, are benchmarked against prior studies, showing notable improvements in SVR hyperparameter optimisation.

Keywords: optimisation; particle swarm optimisation; PSO; support vector regression; SVR; COVID; forecasting; machine learning; Japan.

Reference to this paper should be made as follows: Sykes, E.R. and Wang, Y. (2024) 'Optimising SVR for epidemiological predictions: a case study on COVID-19 mortality in Japan', *Int. J. Artificial Intelligence and Soft Computing*, Vol. 8, No. 5, pp.1–29.

Biographical notes: Edward R. Sykes is an Assistant Professor in the School of Computer Science at the University of Guelph, specialising in AI and Health Innovation. With over a decade of research experience, he develops advanced AI models for chronic disease management, elder care, and healthcare accessibility. He previously founded and directed Sheridan's Centre for Mobile Innovation, transforming it into a multimillion-dollar research hub. He has co-authored 70+ scientific publications and led impactful collaborations with industry and academia. Passionate about mentoring, he engages students at all levels to harness AI for societal benefit.

Copyright © The Author(s) 2024. Published by Inderscience Publishers Ltd. This is an Open Access Article distributed under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)

Yuan Wang is a graduate of the Honours Bachelor of Computer Science (Mobile Computing) program from the Sheridan College in Ontario, Canada. She distinguished herself academically, excelling in all her courses and earning the prestigious Top Thesis Award in her graduating class. She is actively contributed to several research projects at Sheridan's Centre for Applied AI, collaborating with industry partners to solve real-world problems. Beyond her academic and research achievements, she enjoys exploring fine foods and discovering culinary delights in her spare time.

1 Introduction

Optimising machine learning algorithms, especially in terms of hyperparameter tuning, remains a key area of research. Traditional methods like grid search, random search, and trial-and-error, each come with their advantages and limitations (Akande et al., 2017). While grid search offers a thorough exploration of the search space, its computational intensity and inefficiency in multi-dimensional problems limit its practicality (Syarif et al., 2016). Random search, in contrast, has emerged as a more resource-efficient approach, offering a good balance between performance and computational demand (Liashchynskyi and Liashchynskyi, 2019). The trial-and-error method, although somewhat rudimentary, still holds value in quickly exploring the insights of domain experts (Akande et al., 2017).

The evolving landscape of hyperparameter optimisation has seen the rise of particle swarm optimisation (PSO) techniques. PSO, inspired by the social behaviour of birds and fish, offers a robust approach to navigating complex search spaces, making it suitable for a variety of applications, from environmental modelling to financial forecasting (Gupta et al., 2019; Siddique et al., 2020; Lu et al., 2009).

Our research extends this exploration by applying three variants of PSO – fast convergence PSO, reproduction strategy based on spawning global best PSO, and Gaussian distribution-based PSO – to the fine-tuning of support vector regression machine (SVR) hyperparameters, namely, *C*, *gamma*, and *epsilon*. We applied these techniques to the pressing challenge of predicting COVID-19 mortality, using datasets from Tokyo, Japan. This study aims to not only provide accurate and efficient models for this critical application but also to demonstrate the adaptability of PSO in handling complex, real-world datasets. By bridging the gap between theoretical optimisation methods and practical applications, we hope to contribute valuable insights to the fields of machine learning and public health analytics, especially in the context of pandemic response and management.

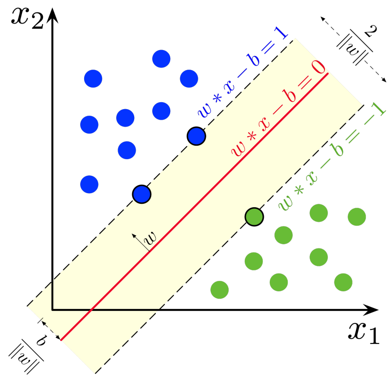
The remainder of this paper is organised as follows: Section 2 delves into the background of PSO, SVR as our primary machine learning algorithm, and a review of current COVID-19 prediction algorithms. In Section 3 (methodology), we detail our SVR-PSO experimental design. Section 4 presents our analysis and findings. The results are then interpreted in the broader context of existing research in Section 5 (discussion). Finally, Section 6 concludes the paper, summarising our contributions, addressing limitations, and suggesting future research directions.

2 Literature review

2.1 Support vector regression machines

SVR is a type of support vector machine (SVM), a popular supervised machine learning algorithm that is used for problems where forecasting is required (Drucker et al., 1997; Goodfellow et al., 2016). It consists of four main components: the *kernel*, the *hyperplane*, the *boundary line*, and the *support vectors* (Drucker et al., 1997). As illustrated in Figure 1, the red line represent the hyperplane, while the dashed lines represent the boundary lines (support vectors).

Figure 1 Support vector machine (see online version for colours)



Source: Cortes and Vapnik (1995)

The SVR is designed to process the data (x_i, y_i), where x_i is the ith sample input vector and y_i is the predicted output (Cortes and Vapnik, 1995). The optimisation function in SVR is:

$$f(x) = w^T x + b \tag{1}$$

where the coefficients, *w* and *b*, are the adjustable weight and bias respectively. To find the optimal *w* and *b* such that $f(x_i) \approx y_i$, the minimisation of regularised risk function is performed for estimation of *w* and *b*:

$$R(C) = \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} |y_i - f(x)|_\epsilon \tag{2}$$

$$|y_i - f(x)|_\epsilon = \begin{cases} 0 & |y_i - f(x)| \leq \epsilon \\ |y_i - f(x)| & \text{otherwise} \end{cases} \tag{3}$$

where C denotes the coefficient for penalty, which is considered as the trade-off tuner between model flatness/structural risk and empirical risk; while *epsilon* (ϵ) is the maximum tolerable error [equation (2)]. Moreover, both *w* and *b* are determined by the user. Equation (3) illustrates that if the predicted value is within the ϵ constraint, the loss would be zero; otherwise the loss would be the difference between actual and forecasted result. To handle cases when the error is larger than ϵ, two slack variables, ξ_i and ${\xi_i}^*$ are introduced to soften margins. Therefore, equation (2) becomes:

$$\min_{w,b,\xi,\xi^*} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} |\xi_i + {\xi_i}^*| \tag{4}$$

which is subject to:

$$s.t. = \begin{cases} y_i - (w^T x_i + b) \leq \epsilon + \xi_i \\ (w^T x_i + b) - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, \text{ where } i = 1, 2, 3, ..., n \\ C > 0 \end{cases} \tag{5}$$

By using dual formation and introducing the language multiplier a_i and a_i^*, the original function can be reformatted as:

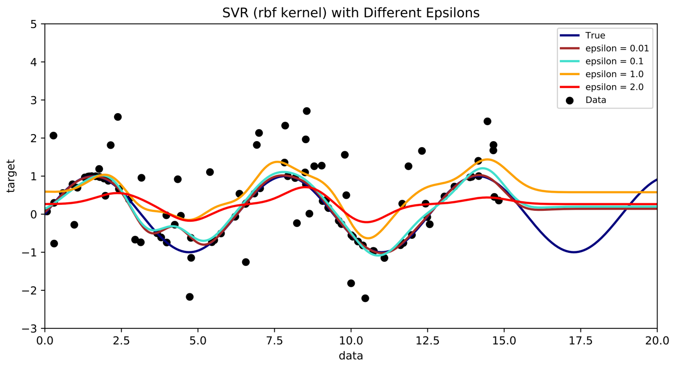
$$f(x) = \sum_{i=1}^{n} (a_i + a_i^*) K(x_i, x) + b \tag{6}$$

where the K represents the *kernel function*. The kernel function determines the type of hyperplane used to separate the data. Common types include *linear*, *polynomial*, *radial basis function* (RBF), and *sigmoid* (Drucker et al., 1997). The RBF is a popular kernel function defined as:

$$K(x_i, x) = \exp\left(-\frac{\|x_i - x\|^2}{\sigma}\right) \tag{7}$$

where σ is the kernel parameter of the function in equation (7) and controls its amplitude.

Figure 2 Support vector regression machine (SVR) with different epsilon values (see online version for colours)



Source: Ji (2023)

Figure 2 illustrates an SVR where the data was generated by a sine wave and predictions are made by the SVR using different epsilons values. As ϵ increases, the prediction becomes less sensitive to errors (Drucker et al., 1997).

In summary, SVR have several key hyperparameters. These hyperparameters are crucial for the performance of SVR models and need to be carefully optimised for the best results.

1 *Kernel:* Determines the type of hyperplane used to separate the data. Common types include linear, polynomial, radial basis function (RBF), and sigmoid.

2 *C* (regularisation parameter): Balances the trade-off between achieving a low error on the training data and minimising model complexity for better generalisation.

3 *Epsilon* (ϵ): Sets the margin of tolerance where no penalty is given to errors. Smaller ϵ-values result in more support vectors and potentially more accurate models, but with the risk of overfitting.

4 *Gamma* (γ): Used in nonlinear SVR algorithms for defining the influence of a single training example. Higher values lead to training examples having a larger influence.

5 *Degree:* The degree of the polynomial that is used in the polynomial kernel function.

2.2 Search algorithms for hyperparameter optimisation

Hyperparameter optimisation, a critical process in machine learning, significantly influences the effectiveness of prediction models (Shukla et al., 2020; Syarif et al., 2016). Common hyperparameter optimisation strategies include grid search, random search, trial and error, and genetic algorithms (Akande et al., 2017).

Grid search, in a classic Euclidean plane, exhaustively explores all possibilities within a grid, with the total iterations being the product of the grid's dimensions (e.g., *rows* $\times$ *columns*). In a 3D space, this expands to $|x| \times |y| \times |z|$. However, its performance diminishes with increasing dimensions due to exponential growth in computational complexity (Akande et al., 2017; Al-Musaylh et al., 2018).

Conversely, random search, in a 2D space, requires significantly fewer test cases than grid search (Al-Musaylh et al., 2018). It often outperforms grid search, particularly when only a subset of hyperparameters significantly impacts performance. The process begins by setting a range for each hyperparameter, followed by random selection within these bounds (Al-Musaylh et al., 2018).

Beyond the grid search and random search, there are other approaches to optimise hyperparameters. Genetic algorithms, for example, have been used in convolution neural networks to systematically fine-tune parameters, in an efficient way (Liashchynskyi and Liashchynskyi, 2019). PSO has also enjoyed the appreciation from diverse researchers in assisting the process of hyperparameter optimisation (Zhao et al., 2018; Lin et al., 2017). On the other hand, some researchers opted for bi-level programming, to combine gradient-base tuning and meta-learning, which in turn contributed positively to the hyperparameter selection of their deep learning network (Franceschi et al., 2018).

Some studies also reported on the use of evolutionary algorithms to fine-tune convolution neural networks (CNN) with significant improvements (Bochinski et al., 2017). The most recent area of exploration is to use AI to identify and construct the parameter search space, then automatically setting up strategies to find the optimal hyperparameters for the problem domain (Akcora et al., 2018).

In 2011, Polson and Scott showed that the SVM's hyperparameters can be effectively optimised using Bayesian interpretation through data augmentation. In this approach the SVM is viewed as a visual model where the parameters are connected through probability distributions. Recently, a scalable version of the Bayesian SVR was developed, enabling the application of Bayesian SVR forecasting on big data (Wenzel et al., 2017).

2.3 Particle swarm optimisation algorithm

PSO, originally inspired by biology, was proposed in 1995 by Kennedy and Eberhart (Olsson, 2011). Modelled on *swarm intelligence*, it uses a metastatic algorithm employing the strategy to explore the search space to find an optimal solution (Olsson, 2011). The following sections present the standard particle swarm optimisation and six of the most commonly used PSO variations (Olsson, 2011), namely:

1. the fast convergence PSO (FCPSO)
2. the genetic algorithm PSO (GAPSO)
3. the reproduction of spawning global best PSO
4. the lifecycle-based sub swarm PSO (LCPPSO)
5. the PSO with fitness adaptive inertia weight
6. the Gaussian distribution PSO (GDPSO).

2.3.1 The standard PSO algorithm

The standard PSO algorithm consists of the following:

- the position of the particle p at iteration k: x_p^k
- the velocity of the particle p at iteration k: v_p^k
- the best position found by p up to iteration k: b_p^k
- the global best solution up to iteration k: g_p^k.

Building on this, the predicted *next position* for particle p may be calculated as:

$$x_p^{k+1} = x_p^k + v_p^{k+1} \tag{8}$$

where the current position at time t is added with the velocity of $t + 1$ to find the next position for the particle. While *velocity* is calculated using the following formula:

$$v_p^{k+1} = wv_p^k + c_1 r_1 (b_p^k - x_p^k) + c_2 r_2 (g_p^k - x_p^k) \tag{9}$$

In this case, w represents the *inertia* and is used as a weight in the calculation. In some situations, a large w is related to a global, large search area, while a smaller inertia is referred to as a local search area. However, that is not always the case. Sometimes a large value of w is needed for the optimisation of unimodal function, which is a local search problem (Nickabadi et al., 2011). On the other hand, the variables c_1 and c_2 are considered as trust parameters, with the former representing the trust the particle has on itself, and the latter being confidence over neighbour's position. The acceleration coefficients, r_1, and r_2 are typically generated by uniform distribution with the range of $[0, 1]$. Algorithm 1 presents the pseudocode for the standard PSO.

Algorithm 1 The standard PSO algorithm with diminishing inertia

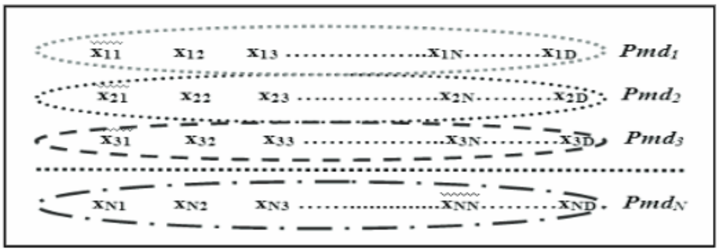
```
Require: $N > 0, D > 0, maxIte > 0$                ▷ # particles, dimensions, and max iterations
Ensure: $f(x) = \frac{1}{n}\sum_{i=1}^{N}(f(x_i) - y)^2$
  for $n \leftarrow 1, ..., N$ do                                    ▷ particles initialisation
      for $d \leftarrow 1, ..., D$ do
          $x_{n,d} \leftarrow rand(lower, upper)$          ▷ uniformly distributed locs in boundaries
          $v_{n,d} \leftarrow rand(lower - upper, upper - lower)$
      end for
      $b_i = x_i$                                                 ▷ initialise personal best
  end for
  while $iteration < maxIte$ do
      for $n \leftarrow 1, ..., N$ do                                ▷ particles initialisation
          for $d \leftarrow 1, ..., D$ do
              $r_1 \leftarrow rand(0, 1)$
              $r_2 \leftarrow rand(0, 1)$
              $v_{n,d} = wv_{n,d} + c_1r_1(b_{n,d} - x_{n,d}) + c_2r_2(g_d - x_{n,d})$
          end for
          $x_n \leftarrow x_n + v_n$
          if $f(x_n) < f(b_n)$ then
              $b_n \leftarrow x_i$
              if $f(b_n) < f(g_n)$ then
                  $g_n \leftarrow b_n$
              end if
          end if
      end for
      $w \leftarrow w_{\min} + \frac{(w_{\max} - w_{\min}) * (maxIte - iteration)}{maxIte}$
      $iteration \leftarrow iteration + 1$
  end while
```

2.4 Variation 1: fast convergence PSO

The FCPSO stands out as one of the most prominent PSO variants (Olsson, 2011). In the standard PSO, the velocity update is governed by equation (9), where the subsequent velocity is influenced by both personal and global bests, given that these values are non-zero. However, this approach can result in convergence being trapped in local minima, especially for functions with high dimensions, due to interdimensional variable restrictions (Sahu et al., 2012). To mitigate this, FCPSO introduces an additional variable, Pmd_i, into the velocity update formula. This variable is calculated as the mean of each dimension's value within the same iteration, as depicted in Figure 3. This leads to a modified velocity calculation as shown in equation (10), where c_3 represents the average best learning factor, and r_3 is a uniformly distributed value between 0 and 1.

$$v_p^{k+1} = wv_p^k + c_1r_1(b_p^k - x_p^k) + c_2r_2(g_p^k - x_p^k) + c_3r_3(Pmd_p^k - x_p^k) \quad (10)$$

Figure 3 Illustration of fast convergence mean dimension calculation



Source: Sahu et al. (2012)

2.5 Variation 2: genetic algorithm particle swarm optimisation

In 1998, researchers introduced a hybrid optimisation technique combining genetic algorithms (GA) and PSO(Ren and Bai, 2010). This approach, termed genetic algorithm PSO (GAPSO), enhances PSO performance by integrating GA's selection mechanisms. In GAPSO, less efficient particles are identified and replaced, intensifying selection pressure while reducing diversity. This method is also referred to as *selection-based PSO*, emphasising the selective replacement of particles. A high-level description of this algorithm is presented in Algorithm 2.

Algorithm 2 Genetic algorithm with particle swarm optimisation (GAPSO)

```
for each particle $i = 1, ..., n$ do
    Randomly select a subset of particles $n_{subset}$
    Evaluate and compare the performance of particle $i$ with the selected subset
end for
Rank all particles based on their performance
Maintain the personal bests but replace the lower-performing half of the particles with the
higher-performing half
```

Source: Ren and Bai (2010)

2.6 Variation 3: reproduction strategy by spawning based on global best in PSO

This PSO variation, introduced by Koay and Srinivasan (2003), enhances the selection-based PSO by focusing on generating new particles influenced by the global best performer. Unlike methods that replace underperforming particles, this approach creates new particles, potentially improving the algorithm's ability to explore the solution space. Algorithm 3 outlines the spawning process based on the global best particle.

Algorithm 3 Reproduction strategy by sprawling based on global best PSO

```
if $particle_i$ is minimum then
    $temp \leftarrow particle_i$
    while $f(particle_i) > f(temp)$ do
        $temp \leftarrow particle_i$
        for $spawn = 1$ to $N$ do
            for $var = 1$ to $num\_of\_spawn$ do
                $potential\_mutant \leftarrow temp + M(0, \sigma)$
                if $f(potential\_mutant) < f(temp)$ then
                    $temp \leftarrow potential\_mutant$
                end if
            end for
        end for
    end while
    $particle_i \leftarrow temp$
end if
```

Source: Koay and Srinivasan (2003)

2.7 Variant 4: lifecycle-based sub swarm PSO

Koay and Srinivasan (2003) proposed an innovative approach to enhance PSO's accuracy and performance by integrating the standard PSO, GAPSO, and stochastic hill climbing into a single, adaptive hybrid algorithm. This method, referred to as LCPSO, leverages the strengths of each individual technique within a lifecycle framework, as detailed in Algorithm 4.

Algorithm 4 Lifecycle-based sub swarm PSO

```
Require: $n = num\_of\_particles$
while termination condition not met do
    for each particle $i = 1$ to $n$ do
        Evaluate fitness of particle $i$
        If no improvement, switch to another lifecycle stage
    end for
    for PSO phase: particles 1 to $n$ do
        Calculate and update velocity for each particle
        Move particle according to updated velocity
    end for
    for GA phase: particles 1 to $n$ do
        Select a subset of particles for reproduction
        Perform crossover and mutation operations
    end for
    for Hill climbing phase: particles 1 to $n$ do
        Explore neighbouring solutions
        Evaluate fitness of new solutions
        Move to a new solution with probability $p$
    end for
end while
```

Source: Koay and Srinivasan (2003)

2.8 Variant 5: PSO with fitness-adaptive inertia weight

This variant of PSO introduces an innovative approach to adjust the inertia weight based on the fitness of the particles. In traditional PSO, the inertia weight is commonly updated uniformly with each iteration. However, this fitness-adaptive method, demonstrated to be effective in a study on oxygen content (Liu et al., 2013), aligns the inertia weight more closely with the performance of individual particles. The adaptive update mechanism is formalised in equation (11).

$$w = \begin{cases} w - (w - w_{\text{min}}) \cdot \left|\frac{f(x_i) - f(p_i)}{f(p_{gbest_i}) - f(p_i)}\right| & \text{if } f(x_i) < f(p_{gbest_i}) \\ w_{\text{min}} + (w_{\text{max}} - w_{\text{min}}) \cdot \frac{1}{1+\exp(-\frac{maxIte}{iteration})} & \text{if } f(p_{gbest_i}) < f(x_i) < f(p_i) \end{cases} \quad (11)$$

Here, w_{max} and w_{min} represent the maximum and minimum inertia weights, respectively. The formula adjusts w based on the fitness difference between a particle and the global best.

Additionally, a modified inertia weight calculation, influenced by the work of Shi and Eberhart (1998), was proposed to further enhance accuracy (Dai et al., 2018a). This alternative equation is presented in equation (12):

$$w = \left(w_{\text{max}} - (w_{\text{max}} - w_{\text{min}}) \cdot \frac{iteration}{maxIte}\right) \cdot \frac{f(iteration) - f(globalFitness)}{f(iteration)} \quad (12)$$

This formula integrates the iteration's progress and the relative improvement in fitness, offering a more dynamic adaptation of the inertia weight.

2.9 Variation 6: Gaussian distribution in PSO

This variation integrates the Gaussian distribution into PSO, presenting a significant modification from the standard PSO where the stochastic components r_1 and r_2 are uniformly distributed (Dai et al., 2018b). In GDPSO, these variables follow a Gaussian distribution, which could provide a more nuanced control over the particles' movement, potentially leading to more effective exploration and exploitation in the search space. Previous research has suggested that this modification may positively impact the optimisation process by adjusting the distribution of particle steps (Dai et al., 2018b).

The algorithm adheres closely to the standard PSO structure, with the key difference being that r_1 and r_2 are drawn from a Gaussian distribution. This distribution is mathematically described as follows:

$$r_i = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (13)$$

Here, r_i (representing either r_1 or r_2) is a random variable with a mean μ and a standard deviation σ, reflecting the Gaussian distribution's parameters. This variation in PSO aims to leverage the properties of the Gaussian distribution to enhance the search process by providing a diverse range of step sizes for the particles.

2.10 Related machine learning work in the COVID-19 domain

The application of machine learning in modelling and forecasting COVID-19 and its variants has been a critical area of research over the past several years. This importance stems from the potential of predictions to guide governments and local authorities in managing the pandemic more effectively and in implementing necessary infrastructure to prevent future outbreaks and reduce mortality.

A significant study compared the predictive capabilities of SVM regression against linear regression, lasso, and exponential smoothing using data from Johns Hopkins University. This research demonstrated the SVM regression model's effectiveness with an R-squared value of 0.53 (Rustam et al., 2020). Contrastingly, another study focusing on India used multiple linear regression to forecast deaths, spread, and recovery, achieving a remarkably high R-squared value of 0.9992 (Kumari et al., 2021). Additionally, a novel approach involving transfer learning with an LSTM network was applied to forecast COVID-19 related deaths. This method yielded impressive error metrics, including RMSE (1.3) and MAE (1.4) for Italy, and RMSE (1.03) and MAE (0.9) for France (Gautam, 2022).

A comprehensive review of machine learning papers on COVID-19 forecasting was completed early January 2024 which revealed a predominant use of compartment models, followed by researcher-designed models, and deep learning techniques (Rahimi et al., 2021). Notably, while SVM was the fourth most common approach, it lagged significantly behind deep learning techniques in frequency of use and was only marginally more common than the fifth-placed approach (Rahimi et al., 2021).

This overview underscores the varied efficacy and application of different machine learning models in pandemic forecasting. The high R-squared value achieved by linear regression in the Indian context suggests its potential effectiveness in specific regional scenarios, while the success of LSTM networks in European countries indicates the utility of deep learning techniques in different geographical settings. The disparity in the frequency of use between different models raises important questions about the factors influencing the choice of methodology, such as data availability, computational resources, and specific forecasting goals. This analysis not only provides insight into the current state of pandemic modelling but also guides future research directions in this rapidly evolving field.

Table 1 COVID-19 mortality predictions using SVR: research paper performance metrics

Research paper title	*Algorithm*	*MAE*	*RMSE*	*R-squared*	*Max error*
'COVID-19 future forecasting using supervised machine learning models' (Rustam et al., 2020)	SVR	23.933	11.967	0.530	NA
'Transfer learning for COVID-19 cases and deaths forecast using LSTM network' (Gautam, 2022)	SVR	1.400	1.300	NA	NA

Table 1 presents the most recent research outcomes from studies focusing on COVID-19 forecasting utilising support vector regression (SVR) machines, all employing the same dataset from Johns Hopkins University. One noteworthy study, titled 'COVID-19 future forecasting using supervised machine learning models', utilised an SVR and benchmarked its efficacy against other models such as linear regression, lasso, and exponential smoothing (Rustam et al., 2020). This study reported an R-squared value of 0.53. Another significant work, 'Transfer learning for COVID-19 cases and deaths forecast using LSTM network' (Gautam, 2022), demonstrated superior mean absolute error (MAE) and root mean square error (RMSE) results, albeit without providing R-squared values.

2.11 Past results of SVR-PSO framework in other research domains

There have been many modern approaches in utilising SVR-PSO and its variants in solving complex issues in various domains, most of which have achieved satisfactory results. One study focused on forecasting multiple-horizon electricity demand implemented a variant of SVR-PSO, achieving the lowest MAE and RMSE (Al-Musaylh et al., 2018). Another research, dealing with modelling a combined infrared radiation convection dryer for grain drying, achieved remarkable accuracy using SVR-PSO (Dai et al., 2018a). SVR-PSO has also been applied in solar radiation prediction (Olsson, 2011), real-time sensor fault analysis (Che, 2013), electrical discharge machining (EDM) modelling (Aich and Banerjee, 2014), stock price forecasting (Olsson, 2011), and financial time series modelling (Lu et al., 2009), to name just a few. All these experiments with SVR-PSO and its variants have yielded decent results in their respective research areas.

2.12 Summary

This literature review has established a thorough foundation in the domains of optimisation, SVR and PSO. It serves as an essential precursor to the methodology section of this study. The primary objective of this research is to develop a machine learning algorithm specifically designed for optimising hyperparameters in forecasting COVID-19 fatalities, with potential applicability in other areas. We have scrutinised leading research in this field, focusing particularly on their methodologies and the results they achieved.

A significant insight from this review is the relatively unexplored area concerning the application of various PSO variants for SVR hyperparameter optimisation, especially in predicting COVID-19 deaths. This lack of research highlights the unique contribution of our study. The forthcoming methodology section details the experimental design, the dataset used, and provides a comprehensive analysis of the PSO variants implemented in our research.

3 Methodology

This section presents the methodology that was used in this study including:

1 experiment setup (Google Colab, SVR coupled with the PSO variants)

2 dataset description and features

3 analysis techniques.

3.1 Experiment setup

The experiments were conducted using Google Colab for data preprocessing, machine learning, training, and validation. Key software and libraries included pandas (v2.1.4), numpy (v1.26.2), Matplotlib (v3.8.2), and Scikit-Learn (v1.3.2). The computational environment comprised an AMD Ryzen 5 5600G with Radeon Graphics (3.90 GHz), 32 GB RAM, running on a 64-bit Windows operating system. No additional computing resources were purchased from Google Colab; the experiments utilised the platform's free tier.

3.2 Dataset and features

Numerous datasets are available for COVID-19 research, including those from the Center for Disease Control and Prevention, Google Open Data Repository, and others. For this study, the COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University was chosen. This dataset is preferred due to its daily updates since 22 January 2020, and its data integrity (Dong et al., 2023). It has also been used in over a dozen ML research papers (Dong et al., 2023). Table 2 details the dataset's features.

Table 2 Dataset description – COVID-19 Dataset Repository

Province/state	Represented province/state, it is legacy column that was replaced by Province_State in 2021
Country/region	Represented country/region, it is legacy column that was replaced by Province_State in 2021
Last update	The time when this record was last updated
Confirmed	The number of people who were confirmed of having COVID-19
Deaths	The number of people who were deceased due to COVID-19
Recovered	The number of people who have recovered from COVID-19
Province_State	The new column that replace the previous column in keeping track of the province/state
Country_Region	The new column that replace the previous column in keeping track of the country/region
Lat	The latitude of this region
Long	The longitude of this region
Combined_Key	The concatenated name for the province and country

Source: Dong et al. (2023)

In this study, we utilised data spanning from 22 January 2020 to 19 November 2023, for training, testing, and evaluating our algorithms. This dataset was sourced from the COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (available at: https://datacatalog.med.nyu.edu/dataset/10400). The repository is updated daily, with each day's data being stored in a separate file labelled by the date of the record.

3.3 Data pre-processing

Data pre-processing involved several key steps:

1. removing rows with missing values
2. discarding irrelevant features (e.g., house size)
3. excluding records with incorrect or null values.

The pre-processed dataset was then divided into dataframes specific to various regions (e.g., Japan, USA, Canada). For this study, the Japanese data was selected for in-depth analysis. These regional dataframes were employed in training and testing the SVR-PSO model variants. Table 3 illustrate the feature sets used for Japan's *X-train* and *Y-train* data.

Table 3 Sample feature sets used for Japan's *X-train* and *Y-train* data, respectively

X-train		*Y-train*	
	Confirmed		*Deaths*
count	633.000000	count	633.000000
mean	16.541449	mean	10.223000
std	1.946977	std	1.273972
min	12.341797	min	8.224002
25%	14.933183	25%	8.842350
50%	16.931303	50%	10.812177
75%	18.504656	75%	11.394463
max	19.769316	max	11.744413

3.4 SVR-PSO algorithms

This study explored the impact of applying different PSO algorithms on SVR hyperparameter tuning by first creating the standard PSO and then exploring the following variants, *FCPSO*, *reproduction strategy by spawning based on global best PSO*, and *GDPSO*.

3.4.1 Flowchart of SVR-PSO optimisation methodology

Figure 4 illustrates a flowchart depicting the high-level functionalities and decision-making steps in our SVR-PSO optimisation process.

3.5 Dependent and independent variables

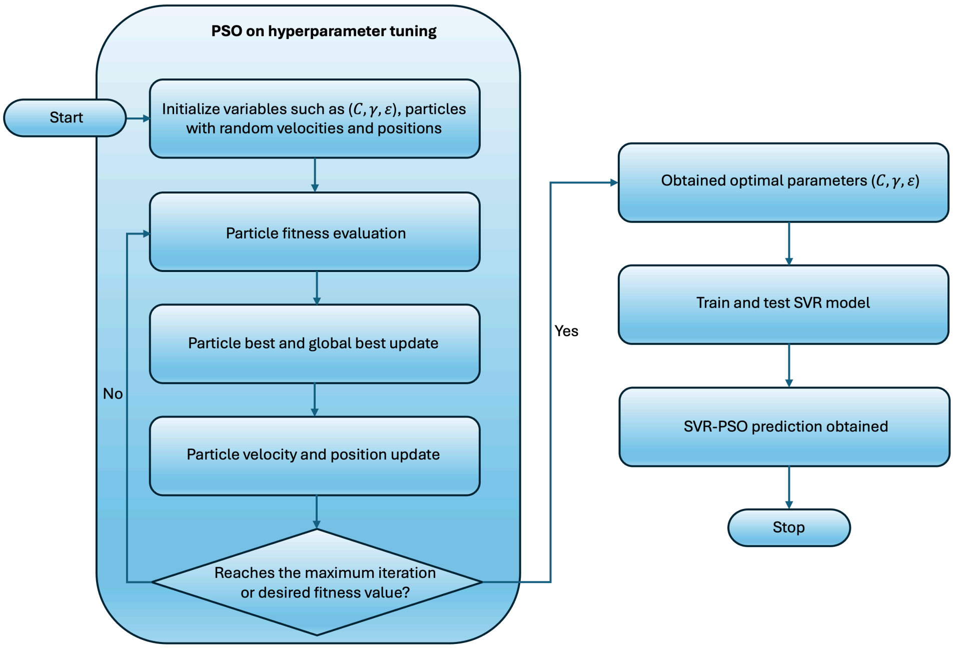
The independent variables in this experiment included:

- the PSO algorithm,
- the number of particles, (N),

- the number of dimensions, (D),
- the number of iterations, $(MaxIte)$, and
- the hardware configuration (CPU, GPU, memory, etc.).

The dependent variables were the SVR hyperparameters (C, ϵ, γ). The key performance metrics commonly used in regression algorithms were then collected, namely mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and R-squared (R^2). These metrics are further elaborated in the following section.

Figure 4 Flow diagram of optimisation methodology for SVR hyperparameters (see online version for colours)


PSO on hyperparameter tuning
Start
Initialize variables such as (C, γ, ε), particles with random velocities and positions
Particle fitness evaluation
Particle best and global best update
Particle velocity and position update
Reaches the maximum iteration or desired fitness value?
No
Yes
Obtained optimal parameters (C, γ, ε)
Train and test SVR model
SVR-PSO prediction obtained
Stop

3.6 Performance evaluation metrics

We employed several metrics to assess the performance of our SVR-PSO models, commonly used in evaluating ML regression algorithms (Goodfellow et al., 2016; Pouyanfar et al., 2019).

The *MAE* measures the average magnitude of errors in a set of predictions, without considering their direction. It is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. A lower MAE indicates better model performance. MAE does not have an upper limit, and its interpretability depends on the context and scale of the data (Goodfellow et al., 2016). MAE is defined as:

$$MAE = \frac{\sum_{i=1}^{n} |\hat{Y}_i - Y_i|}{n} \tag{14}$$

where Y_i represents the actual value and $\hat{Y}_i$ the predicted value.

RMSE is similar to MAE but gives more weight to larger errors, as it involves squaring the individual differences. This means RMSE is more sensitive to outliers than MAE. Like MAE, a lower RMSE indicates a better fit, and its value depends on the scale of the data. RMSE is more commonly used when large errors are particularly undesirable. RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}} \tag{15}$$

R-squared, also known as the coefficient of determination, is a statistical measure in regression analysis that represents the proportion of the variance for a dependent variable that is explained by an independent variable or variables in a regression model. R^2 is defined as:

$$R^2 = 1 - \frac{SS_{reg}}{SS_{tot}} \tag{16}$$

where SS_{reg} is the sum of squares due to regression (explained sum of squares); and SS_{tot} is the total sum of squares. The R^2 value ranges from 0 to 1. A value of 0 indicates that the model does not explain any of the variability of the response data around its mean. A value of 1 indicates that the model explains all the variability of the response data around its mean. In terms of performance:

- an R^2 close to 1 is usually considered excellent, indicating a high level of correlation between the observed and predicted values
- a moderate R^2 (e.g., 0.5-0.7) might be considered good, depending on the context and domain of the study
- a low R^2 (closer to 0) is generally viewed as poor, suggesting that the model fails to accurately capture the variance in the dependent variable.

Max error in regression analysis is a metric that identifies the largest single error between the predicted and actual values in the dataset. It focuses on the worst-case scenario, providing insight into the maximum deviation in the model's predictions. Unlike MAE and RMSE, max error is not influenced by the average performance across the entire dataset but is solely concerned with the largest error. Max error is defined as:

$$Max\ error = \max(|y_i - \hat{y}_i|) \tag{17}$$

where y_i represents the actual value and $\hat{y}_i$ the predicted value. While there is no standard range for good or bad values, a lower max error is generally preferable as it indicates less deviation in the worst-case prediction.

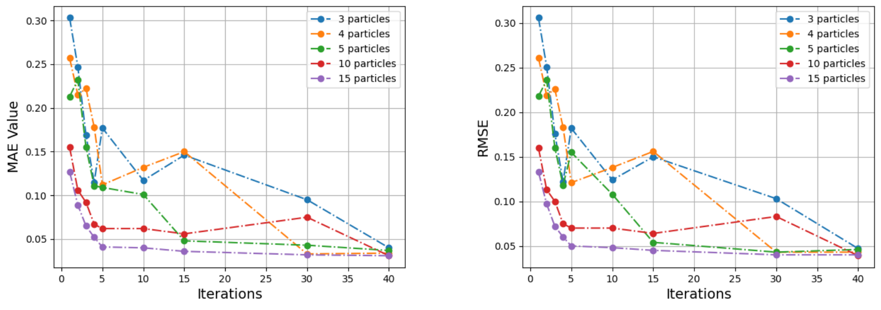
4 Findings (analysis and evaluation)

This section presents the main findings obtained through the experiment and displayed through plots. There are four types of graphs for each variant of PSO-SVR for Japan related data, and they are positioned in the order of MSE, RMSE, R-squared value and max error. The order of PSO-SVR is as follows: fast convergence PSO-SVR configuration, the spawning global best PSO-SVR, and, the Gaussian distribution PSO-SVR configuration.

4.1 Fast convergence PSO configuration

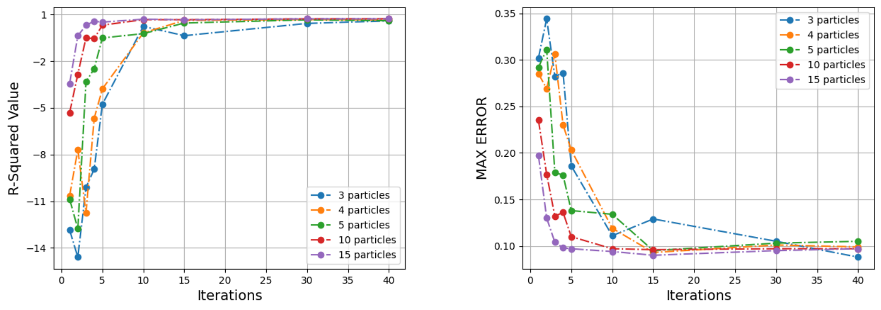
Figure 5 present the MAE and RMSE results for the fast convergence PSO-SVR on Japanese COVID data for 3–15 particles and 1–40 iterations. Figure 6 presents the R-squared and max error results on the FCPSO. As shown in Figures 5 and 6(b), there is a clear pattern of improvement in the SVR predictions with increased number of particles, where the higher particle count, generally yielded better results. Furthermore, there is also an improvement with increased number of iterations. The best prediction performance under this configuration was achieved for MAE = 0.031 (15 particles, 40 iterations); RMSE = 0.039 (10 particles, 40 iterations); R-squared = 0.715 (15 particles, 40 iterations); and max error = 0.090 (15 particles, 15 iterations).

Figure 5 Fast convergence PSO: 3–15 particles; 0–40 iterations, (a) MSE and (b) RMSE results (see online version for colours)



(a) (b)

Figure 6 Fast convergence PSO: 3–15 particles; 0–40 iterations, (a) R^2 and (b) max error results (see online version for colours)



(a) (b)

4.2 *Spawning global best PSO-SVR configuration*

Figures 7 and 8 present the MAE and RMSE, and R-squared and max error results, respectively, for the spawning global best PSO-SVR configuration for 3-15 particles and 1-40 iterations. Figures 7 and 8(b) indicate that there is a distinct improvement in the SVR's prediction capability with increased number of particles. Furthermore, there is also an improvement when the number of iterations is increased. The best prediction performance for the spawning global best PSO-SVR was achieved for MAE = 0.036 (15 particles, 40 iterations); RMSE = 0.042 (15 particles, 30 iterations); R-squared = 0.713 (15 particles, 40 iterations); and max error = 0.032 (15 particles, 40 iterations).

Figure 7 Spawning global best PSO: 3–15 particles; 0–40 iterations, (a) MAE and (b) RMSE results (see online version for colours)

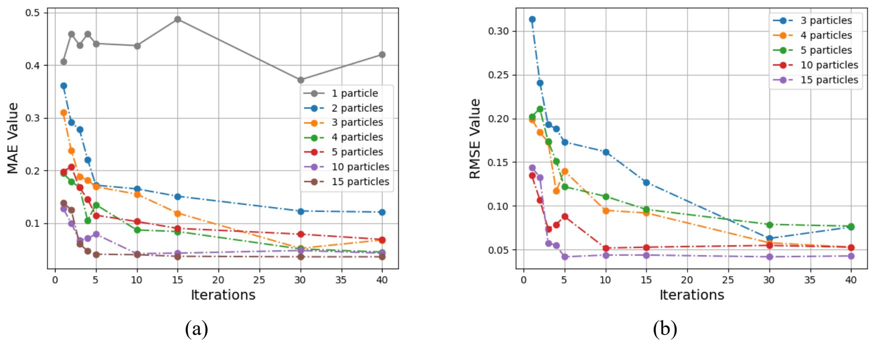
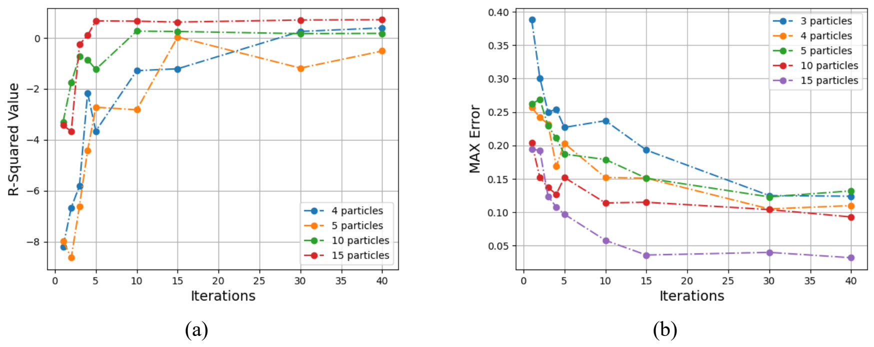


Figure 8 Spawning global best PSO: 3–15 particles; 0–40 iterations, (a) R-squared and (b) max error results (see online version for colours)



4.3 *Gaussian distribution SVR-PSO configuration*

Figures 9 and 10 present the MAE and RMSE, and R-squared and max error results, respectively, for the Gaussian PSO-SVR configuration for 3-15 particles and 1-40 iterations. Figures 9 and 10(b) present a similar pattern indicating that there is a significant improvement in the SVR's predictions as the number of particles increase.

This is also true as the number of iterations is increased. It was discovered that the SVR produced the best predictions for MAE = 0.035 (15 particles, 40 iterations); RMSE = 0.042 (15 particles, 40 iterations); R-squared = 0.704 (15 particles, 30 iterations); and max error = 0.034 (15 particles, 15 iterations).

Figure 9 Gaussian PSO: 3–15 particles; 0–40 iterations, (a) MAE and (b) RMSE results (see online version for colours)

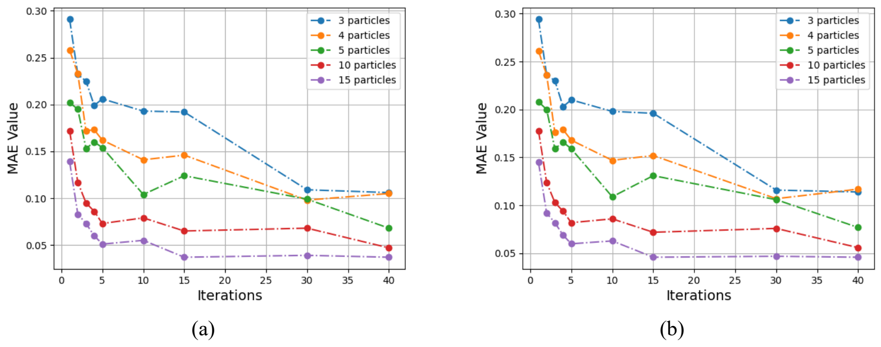
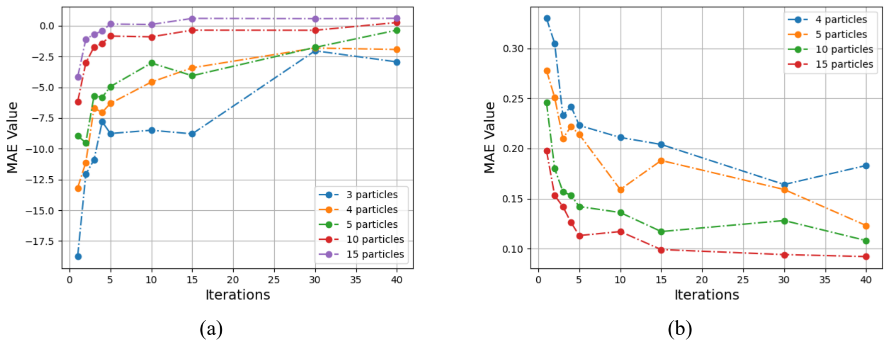


Figure 10 Gaussian PSO: 3–15 particles; 0–40 iterations, (a) R-squared and (b) max error results (see online version for colours)



5 Discussion

5.1 Analysis of results

Our findings, detailed in Section 4, reveals a consistent improvement in SVR predictions with increased particles in all PSO variants. Notably, this enhancement exhibits diminishing returns beyond certain thresholds of particles and iterations, following a logarithmic pattern. After a sharp initial improvement, performance gains plateau. The following observations have been gleaned from the experiments conducted on each of the PSO variants:

For the fast convergence PSO:

- the best performance for MAE (≈0.037), RMSE (≈0.390) and R^2 (≈0.7) was achieved at 40 iterations (where all 3-, 4-, 5-, 10- and 15-particles converged)
- with 15 particles, the R^2 reached 0.7 within four iterations. From this point, there was no statistically significant improvement with increased iterations.

For the spawning global best PSO:

- RMSE stabilised after approximately 20 iterations, regardless of the number of particles
- there was no statistically significant difference in MAE, RMSE or R^2 results between 5 and 40 iterations using 15 particles.

For the Gaussian distribution PSO:

- the performance of this PSO was more impacted by the number of particles than other PSO algorithms tested; the best performance was achieved with 15 particles (the maximum explored)
- there was no statistically significant difference in MAE, RMSE or R^2 results between 15 and 40 iterations using 15 particles.

In cases where only one particle is used in PSO, there is no significant improvement in the optimisation process. This is because, with a single particle, the global best position and velocity are identical to that of the particle itself. This scenario leads to ineffective movement and velocity calculations, and thus does not effectively guide the optimisation process. Essentially, the lack of diversity in potential solutions restricts the algorithm's ability to explore and converge on the most optimal solution.

In our study, we observed a strong correlation among MAE, RMSE, R^2, and max error across all PSO variants. Lower errors in MAE and RMSE typically indicated a closer approach to the optimal R^2 value. However, max error's relationship with R^2 was not as straightforward. For example, at a particle count of 15, max error remained constant from the 5^{th} to the 40^{th} iteration, while R^2 values showed slight but continuous improvements. In the context of mortality forecasting in Japan using fast convergence PSO, we noted that the R^2 value had an upper limit of approximately 0.715, unaffected by changes in the number of particles or iterations.

Overall, a proportional relationship was discovered: *the number of particles* $\propto$ *rate of convergence*; and *the number of iterations* $\propto$ *rate of convergence*. However, there are diminishing returns as the relationship of both variables' impact on SVR's performance is logarithmic in nature. This characteristic was confirmed through additional experimentation. We also explored what happens when we significantly increase the number of particles and iterations. We experimented with up to 50 particles for the FCPSO and went up to 50 iterations. The summary results are shown in Table 4 which presents the R-squared values using 50 particles for the fast convergence PSO using 20 and 50 iterations. The best we were able to achieve was an R^2 of 0.717 which was with 50 particles and 50 iterations; this is only a 0.001 improvement over the 50 particles and 20 iteration case, and a 0.002 improvement over the 15 particles and 40 iterations case (the base maximum for all experiments conducted in this study). To achieve these results took considerable GPU processing resources and time.

Table 4 R^2 results using large particle count and high number of iterations (fast convergence PSO)

Iteration count	20	50
R-squared results (using 50 particles)	0.716	0.717

These findings demonstrate that substantially increasing the particle count and iteration number beyond 15 particles and 40 iterations offers little to no enhancement in SVR's predictive performance. This observation highlights the importance of optimising resource usage in PSO applications, avoiding unnecessary computational expenses for minimal gains in accuracy.

Our results indicate diminishing returns when increasing the number of particles and iterations in PSO optimisation. Specifically, extending beyond 15 particles and 30 iterations did not yield significant improvements in performance. The relationship between particle/iteration count and the R^2 value follows a logarithmic pattern, with rapid initial improvements tapering off over time.

Similarly, the reproduction of spawning global best method (variant 2) mirrored the fast convergence PSO in terms of performance. In both cases, increasing particles and iterations beyond 15 particles at 15 iterations did not significantly enhance the R^2 value, highlighting an optimisation threshold in these PSO variants.

5.2 Comparison with previous studies

As shown in Table 5, our work shows a significant improvement over previous research. For the paper that utilised SVM regression against linear regression, lasso and exponential smoothing (Rustam et al., 2020), they have achieved an R-squared value of 0.53. Even though the R-squared value is missing in the second paper (Gautam, 2022), it has a higher MAE and RMSE value using the dataset from John Hopkins University. Comparing with the past papers, it not only demonstrates that this experiment provided additional insight and contribution to the COVID-19 death forecasts using SVR, but it also showcased the effectiveness of PSO in fine-tuning of the SVR hyperparameters, namely *C*, *gamma* and *epsilon*.

Our research has demonstrated that utilising various PSO techniques with linear SVR for COVID-19 mortality forecasting yields promising results, achieving an R-squared value greater than 0.7. This is accompanied by satisfactory MSE, RMSE, and max error values, which improved with increasing iterations and particles. Compared to previous studies, such as Rustam et al. (2020) which reported an R-squared value of 0.53 using SVR, our findings underscore the enhanced effectiveness of PSO variants in precisely fine-tuning SVR hyperparameters.

Table 5 COVID-19 mortality predictions using SVR: research papers and our algorithm

Research paper title	*Algorithm*	*MAE*	*RMSE*	*R-squared*	*Max error*
'COVID-19 future forecasting using supervised machine learning models' (Rustam et al., 2020)	SVR	23.933	11.967	0.530	NA
'Transfer learning for COVID-19 cases and deaths forecast using LSTM network' (Gautam, 2022)	SVR	1.400	1.300	NA	NA
Our fast convergence PSO-SVR algorithm	SVR	0.031	0.040	0.717	0.091

6 Conclusions

6.1 Summary

This study advanced the application of PSO variants for optimising the hyperparameters of linear SVR in the context of COVID-19 mortality forecasting. Conducted on Google Colab, our research focused on the efficacy of various PSO-SVR configurations. The primary contributions of our work include:

- The fast convergence PSO-SVR variant excelled, surpassing all previous studies in this domain with an R-squared value of 0.717 and corresponding MSE and RMSE values around 0.03 and 0.04, respectively. This represents a significant leap forward in the field of COVID-19 mortality prediction.
- Our results demonstrate improved accuracy in forecasting COVID-19 mortality using SVR and PSO, underscoring the potential of these methods for precise epidemiological analysis.
- Consistency was observed across different PSO variants in terms of the number of particles and iterations needed to achieve optimal R-squared values, indicating the robustness of our PSO-SVR methodology.
- When compared to previous studies, notably Rustam et al. (2020) which reported an R-squared value of 0.53, our approach shows considerable improvements, as detailed in Table 5. This achievement underscores our contributions in enhancing accuracy and fine-tuning the SVR hyperparameters, specifically *C*, *gamma*, and *epsilon*.
- The versatility of the SVR and PSO optimisation techniques developed in this study suggests their potential for effective application across various domains beyond epidemiological forecasting.

6.2 Limitations

Our study effectively implemented, assessed, and evaluated three specific PSO variants in combination with SVR: the fast convergence PSO, the spawning global best PSO, and the Gaussian distribution PSO. While these variants yielded valuable insights, our research scope was limited and did not encompass other promising PSO variations that might offer further enhancements. Potential future explorations could excitingly encompass:

1 the GAPSO, merging genetic algorithm principles with PSO for enhanced optimisation

2 the LCPPSO, promising novel insights with its unique sub-swarm dynamics approach

3 the PSO with fitness adaptive inertia weight, aiming for a more dynamic PSO adaptation process.

Investigating these additional variants could significantly broaden our understanding and efficacy of PSO in complex predictive modelling tasks, such as COVID-19 mortality forecasting.

6.3 Future work

Future research should focus on thoroughly exploring and evaluating the remaining PSO-SVR variants outlined in our methodology. This endeavour is crucial to identify any particular variant that might significantly excel in forecasting COVID-19 mortality. While our study has implemented and analysed three of the six proposed PSO-SVR configurations, revealing similar trends in hyperparameter fine-tuning of linear SVR, it is imperative to conduct a comprehensive examination of all variants for an in-depth understanding and enhanced optimisation.

An essential aspect of future research will be the evaluation of computational efficiency in these models. Considering the resource-intensive nature of these optimisations, it is vital to meticulously measure and analyse processing time, with a special emphasis on CPU and GPU resource utilisation throughout the modelling process.

A detailed evaluation of these efficiency metrics will illuminate the computational requirements of each PSO-SVR variant and further inform their practical utility and scalability in a range of real-world applications.

Acknowledgements

We would like to express our gratitude to the advisory committee members for their invaluable support and guidance in developing the thesis that served as the foundation for this work.

References

Aich, U. and Banerjee, S. (2014) 'Modeling of EDM responses by support vector machine regression with parameters selected by particle swarm optimization', *Applied Mathematical Modelling*, Vol. 38, No. 11, pp.2800–2818.

Akande, K.O., Owolabi, T.O., Olatunji, S.O. and AbdulRaheem, A. (2017) 'A hybrid particle swarm optimization and support vector regression model for modelling permeability prediction of hydrocarbon reservoir', *Journal of Petroleum Science and Engineering*, Vol. 150, pp.43–53.

Akcora, C.G., Dey, A.K., Gel, Y.R. and Kantarcioglu, M. (2018) 'Forecasting bitcoin price with graph chainlets', in Phung, D., Tseng, V.S., Webb, G.I., Ho, B., Ganji, M. and Rashidi, L. (Eds.): *Advances in Knowledge Discovery and Data Mining*, pp.765–776, Springer International Publishing, Cham.

Al-Musaylh, M.S., Deo, R.C., Li, Y. and Adamowski, J.F. (2018) 'Two-phase particle swarm optimized-support vector regression hybrid model integrated with improved empirical mode decomposition with adaptive noise for multiple-horizon electricity demand forecasting', *Applied Energy*, Vol. 217, pp.422–439.

Bochinski, E., Senst, T. and Sikora, T. (2017) 'Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms', *2017 IEEE International Conference on Image Processing (ICIP)*, pp.3924–3928.

Che, J. (2013) 'Support vector regression based on optimal training subset and adaptive particle swarm optimization algorithm', *Applied Soft Computing*, Vol. 13, p.3473–3481.

Cortes, C. and Vapnik, V. (1995) 'Support-vector networks', *Machine Learning*, Vol. 20, No. 3, pp.273–297.

Dai, A., Zhou, X., Dang, H., Sun, M. and Wu, Z. (2018a) 'Intelligent modeling method for a combined radiation-convection grain dryer: a support vector regression algorithm based on an improved particle swarm optimization algorithm', *IEEE Access*, Vol. 6, pp.14285–14297.

Dai, H-P., Chen, D-D. and Zheng, Z-S. (2018b) 'Effects of random values for particle swarm optimization algorithm', *Algorithms*, Vol. 11, p.23.

Dong, E., Du, H. and Gardner, L. (2023) *Johns Hopkins University COVID-19 Data Repository* [online] https://datacatalog.med.nyu.edu/dataset/10400 (accessed 3 January 2023).

Drucker, H., Burges, C.C., Kaufman, L., Smola, A.J. and Vapnik, V.N. (1997) 'Support vector regression machines', *Advances in Neural Information Processing Systems*, Vol. 9, p.155–161.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R. and Pontil, M. (2018) 'Bilevel programming for hyperparameter optimization and meta-learning', in Dy, J. and Krause, A. (Eds.): *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp.1568–1577.

Gautam, Y. (2022) 'Transfer learning for COVID-19 cases and deaths forecast using LSTM network', *ISA Transactions*, Vol. 124, pp.41–56.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*, MIT Press, Cambridge, AM, USA.

Gupta, D., Pratama, M., Ma, Z., Li, J. and Prasad, M. (2019) 'Financial time series forecasting using twin support vector regression', *PLOS ONE*, Vol. 14, No. 3, pp.1–27.

Ji, S. (2023) *Support Vector Regression (SVR) (RBF Kernel) with Different Epsilons* [online] https://upload.wikimedia.org/wikipedia/commons/7/7a/Svr_epsilons_demo.svg (accessed 2 January 2023).

Koay, C.A. and Srinivasan, D. (2003) 'Particle swarm optimization-based approach for generator maintenance scheduling', *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*, IEEE, pp.167–173.

Kumari, R., Kumar, S., Poonia, R.C., Singh, V., Raja, L., Bhatnagar, V. and Agarwal, P. (2021) ‘Analysis and predictions of spread, recovery, and death caused by COVID-19 in India’, *Big Data Mining and Analytics*, Vol. 4, No. 2, pp.65–75.

Liashchynskyi, P. and Liashchynskyi, P. (2019) *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*, Computing Research Repository [online] http://arxiv.org/abs/1912.06059 (accessed 2 November 2022).

Lin, W-C., Yin, Y., Cheng, S-R., Cheng, T., Wu, C-H. and Wu, C-C. (2017) ‘Particle swarm optimization and opposite-based particle swarm optimization for two-agent multi-facility customer order scheduling with ready times’, *Applied Soft Computing*, Vol. 52, pp.877–884.

Liu, S., Xu, L., Li, D., Li, Q., Jiang, Y., Tai, H. and Zeng, L. (2013) ‘Prediction of dissolved oxygen content in river crab culture based on least squares support vector regression optimized by improved particle swarm optimization’, *Computers and Electronics in Agriculture*, Vol. 95, pp.82–91.

Lu, C-J., Lee, T-S. and Chiu, C-C. (2009) ‘Financial time series forecasting using independent component analysis and support vector regression’, *Decision Support Systems*, Vol. 47, No. 2, pp.115–125.

Nickabadi, A., Ebadzadeh, M.M. and Safabakhsh, R. (2011) ‘A novel particle swarm optimization algorithm with adaptive inertia weight’, *Applied Soft Computing*, Vol. 11, No. 4, pp.3658–3670.

Olsson, A.E. (2011) ‘Engineering tools, techniques and tables’, *Particle Swarm Optimization: Theory, Techniques and Applications*, Nova Science Publishers, Inc., Hauppauge, NY.

Polson, N.G. and Scott, S.L. (2011) ‘Data augmentation for support vector machines’, *Bayesian Analysis*, Vol. 6, No. 1, p.1–23.

Pouyanfar, S., Sadiq, S., Yan, B., Tian, H., Tao, Y., Reyes M.P., Shyu, M-L., Chen, S-C. and Iyengar, S.S. (2019) ‘A survey on deep learning: algorithms, techniques, and applications’, *ACM Comput. Surv.*, Vol. 51, No. 5, pp.1–36.

Rahimi, I., Chen, F. and Gandomi, A. (2021) ‘A review on COVID-19 forecasting models’, *Neural Computing and Applications*, Vol. 35, pp.23671–23681.

Ren, Y. and Bai, G. (2010) ‘Determination of optimal SVM parameters by using GA/PSO, *J. Comput.*, Vol. 5, No. 8, pp.1160–1168.

Rustam, F., Reshi, A.A., Mehmood, A., Ullah, S., On, B-W., Aslam, W. and Choi, G.S. (2020) ‘COVID-19 future forecasting using supervised machine learning models’, *IEEE Access*, Vol. 8, pp.101489–101499.

Sahu, A., Panigrahi, S.K. and Pattnaik, S. (2012) ‘Fast convergence particle swarm optimization for functions optimization’, *2nd International Conference on Computer, Communication, Control and Information Technology (C3IT-2012), Procedia Technology*, 25–26 February, Vol. 4, pp.319–324.

Shi, Y. and Eberhart, R. (1998) ‘A modified particle swarm optimizer’, *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)*, pp.69–73.

Shukla, R., Agrawal, J., Sharma, S., Chaudhari, N. and Shukla, K. (2020) *Social Networking and Computational Intelligence: Proceedings of SCI-2018. Lecture Notes in Networks and Systems*, Springer, Singapore.

Siddique, M., Mohanta, D. and Mishra, S. (2020) ‘A hybrid model of artificial neural network and particle swarm optimization for forecasting of stock price of Tata Motors’, *Spectrochim Acta A Mol. Biomol. Spectrosc.*, Vol. 10, pp.18299–18305.

Syarif, I., Prugel-Bennett, A. and Wills, G. (2016) ‘SVM parameter optimization using grid search and genetic algorithm to improve classification performance’, *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, Vol. 14, p.1502.

Wenzel, F., Galy-Fajou, T., Deutsch, M. and Kloft, M. (2017) ‘Bayesian nonlinear support vector machines for big data’, *Machine Learning and Knowledge Discovery in Databases. Lecture Notes in Computer Science*, Vol. 10534, p.307–322.

Zhao, W., Luan, Z. and Wang, C. (2018) 'Parametric optimization of novel electric-hydraulic hybrid steering system based on a shuffled particle swarm optimization algorithm', *Journal of Cleaner Production*, Vol. 186, pp.865–876.

Appendix

This appendix contains the supplementary tables of results from our experiments on the *Fast Convergence PSO*, *Reproduction of Spawning Global Best PSO*, and *Gaussian Distribution PSO*. The findings presented in Section 4 and the subsequent analysis in Section 5 are based on the data detailed in these tables. These tables provide the empirical foundation for our research conclusions and facilitate a comprehensive understanding of our study's outcomes.

Table 6 Mean absolute error for fast convergence PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.417	0.390	0.471	0.382	0.405	0.413	0.434	0.469	0.430
2 particles	0.365	0.338	0.283	0.230	0.186	0.170	0.136	0.088	0.132
3 particles	0.303	0.247	0.169	0.115	0.177	0.117	0.146	0.095	0.040
4 particles	0.257	0.215	0.222	0.178	0.112	0.132	0.150	0.033	0.034
5 particles	0.213	0.232	0.155	0.111	0.109	0.101	0.048	0.043	0.037
10 particles	0.155	0.106	0.092	0.067	0.062	0.062	0.056	0.075	0.031
15 particles	0.127	0.089	0.065	0.052	0.041	0.040	0.036	0.032	0.031

Table 7 Root mean square error for fast convergence PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.420	0.392	0.474	0.385	0.408	0.415	0.426	0.471	0.432
2 particles	0.3684	0.3417	0.2873	0.2356	0.1912	0.175	0.143	0.094	0.1368
3 particles	0.306	0.251	0.176	0.122	0.182	0.124	0.150	0.103	0.047
4 particles	0.261	0.219	0.226	0.183	0.121	0.138	0.156	0.043	0.043
5 particles	0.218	0.236	0.160	0.118	0.115	0.108	0.054	0.043	0.046
10 particles	0.160	0.1132	0.100	0.075	0.070	0.704	0.642	0.083	0.039
15 particles	0.133	0.097	0.728	0.060	0.050	0.048	0.045	0.040	0.040

Table 8 R-squared results for fast convergence PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	–45.947	–51.884	–34.215	–40.035	–35.151	–29.479	–52.875	–46.463	–40.314
2 particles	–21.843	–28.575	–17.739	–7.668	–10.242	–7.810	–1.228	0.323	0.683
3 particles	–12.857	–14.594	–10.118	–8.942	–4.774	0.228	–0.367	0.422	0.584
4 particles	–10.656	–7.691	–11.740	–5.674	–3.784	–0.189	0.616	0.666	0.646
5 particles	–10.870	–12.770	–3.329	–2.488	–0.513	–0.236	0.446	0.652	0.593
10 particles	–5.315	–2.847	–0.508	–0.552	0.307	0.664	0.659	0.712	0.712
15 particles	–3.441	–0.378	0.302	0.563	0.504	0.693	0.655	0.713	0.715

Table 9 Max error for fast convergence PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.529	0.549	0.442	0.513	0.474	0.458	0.559	0.524	0.507
2 particles	0.389	0.433	0.357	0.252	0.288	0.261	0.151	0.097	0.099
3 particles	0.302	0.344	0.282	0.286	0.186	0.111	0.129	0.105	0.088
4 particles	0.285	0.269	0.306	0.230	0.203	0.119	0.093	0.101	0.099
5 particles	0.292	0.311	0.179	0.176	0.138	0.134	0.095	0.103	0.105
10 particles	0.235	0.177	0.132	0.136	0.110	0.097	0.096	0.097	0.097
15 particles	0.197	0.130	0.104	0.098	0.097	0.094	0.090	0.095	0.097

Table 10 Mean absolute error (MAE) for reproduction of spawning global best PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.407	0.459	0.438	0.459	0.441	0.437	0.487	0.372	0.420
2 particles	0.362	0.291	0.278	0.221	0.172	0.165	0.151	0.123	0.121
3 particles	0.310	0.238	0.188	0.182	0.169	0.155	0.119	0.052	0.068
4 particles	0.195	0.179	0.168	0.105	0.135	0.087	0.084	0.051	0.045
5 particles	0.198	0.207	0.168	0.145	0.115	0.103	0.090	0.079	0.069
10 particles	0.128	0.100	0.067	0.071	0.079	0.042	0.043	0.048	0.043
15 particles	0.139	0.125	0.061	0.047	0.041	0.040	0.037	0.036	0.036

Table 11 Root mean square error (RMSE) for reproduction of spawning global best PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.409	0.461	0.440	0.461	0.443	0.439	0.488	0.376	0.420
2 particles	0.364	0.293	0.280	0.225	0.177	0.172	0.157	0.129	0.128
3 particles	0.313	0.241	0.193	0.188	0.173	0.162	0.127	0.063	0.076
4 particles	0.199	0.184	0.173	0.117	0.140	0.095	0.092	0.058	0.053
5 particles	0.202	0.211	0.174	0.151	0.122	0.111	0.096	0.079	0.077
10 particles	0.135	0.107	0.074	0.079	0.088	0.052	0.053	0.055	0.053
15 particles	0.144	0.133	0.068	0.055	0.042	0.044	0.044	0.042	0.043

Table 12 R-squared value for reproduction of spawning global best PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	–34.577	–47.199	–40.388	–44.932	–43.494	–41.990	–35.341	–31.841	–39.400
2 particles	–28.064	–16.542	–16.990	–10.760	–7.055	–6.183	–5.081	–3.626	–3.688
3 particles	–19.198	–11.939	–8.149	–8.181	–5.824	–5.412	–3.333	0.156	–0.636
4 particles	–8.207	–6.663	–5.815	–2.157	–3.657	–1.286	–1.212	0.255	0.394
5 particles	–7.978	–8.633	–6.606	–4.409	–2.721	–2.822	0.0374	–1.184	–0.513
10 particles	–3.305	–1.758	–0.727	–0.860	–1.233	0.267	0.253	0.169	0.179
15 particles	–3.443	–3.669	–0.258	0.102	0.671	0.658	0.623	0.704	0.713

Table 13 Max error for reproduction of spawning global best PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.476	0.518	0.512	0.527	0.510	0.510	0.551	0.442	0.488
2 particles	0.423	0.356	0.332	0.284	0.237	0.240	0.224	0.183	0.185
3 particles	0.388	0.301	0.250	0.254	0.227	0.237	0.193	0.125	0.124
4 particles	0.257	0.242	0.232	0.169	0.203	0.152	0.151	0.105	0.110
5 particles	0.262	0.269	0.229	0.211	0.187	0.179	0.151	0.123	0.132
10 particles	0.204	0.152	0.137	0.127	0.152	0.114	0.115	0.104	0.093
15 particles	0.195	0.192	0.124	0.108	0.097	0.058	0.036	0.040	0.032

Table 14 Mean absolute error (MAE) for Gaussian distribution PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.421	0.425	0.426	0.463	0.439	0.427	0.489	0.392	0.419
2 particles	0.355	0.299	0.278	0.222	0.174	0.167	0.152	0.122	0.122
3 particles	0.315	0.236	0.188	0.184	0.167	0.154	0.117	0.051	0.069
4 particles	0.196	0.175	0.168	0.105	0.137	0.088	0.085	0.052	0.046
5 particles	0.199	0.206	0.168	0.147	0.113	0.106	0.091	0.080	0.070
10 particles	0.129	0.103	0.067	0.074	0.077	0.046	0.045	0.045	0.038
15 particles	0.133	0.128	0.061	0.048	0.045	0.045	0.036	0.038	0.035

Table 15 Root mean square error (RMSE) for Gaussian distribution PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.405	0.467	0.453	0.445	0.455	0.436	0.487	0.375	0.422
2 particles	0.367	0.296	0.282	0.230	0.178	0.175	0.155	0.131	0.130
3 particles	0.314	0.243	0.193	0.190	0.176	0.166	0.123	0.067	0.075
4 particles	0.200	0.186	0.176	0.118	0.143	0.097	0.095	0.061	0.055
5 particles	0.205	0.213	0.176	0.153	0.126	0.115	0.098	0.081	0.073
10 particles	0.134	0.108	0.076	0.080	0.085	0.056	0.056	0.056	0.052
15 particles	0.147	0.134	0.069	0.056	0.046	0.046	0.043	0.046	0.042

Table 16 R-squared value for Gaussian distribution PSO

Iterations	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	–37.366	–46.635	–42.388	–44.396	–43.467	–41.963	–35.653	–31.094	–39.524
2 particles	–23.275	–15.367	–15.990	–10.635	–7.050	–6.375	–5.363	–3.652	–3.649
3 particles	–15.737	–11.958	–7.149	–8.463	–5.396	–5.265	–3.958	0.299	–0.592
4 particles	–7.643	–7.959	–5.456	–2.986	–3.602	–1.725	–1.172	0.359	0.375
5 particles	–5.745	–6.745	–4.653	–4.479	–2.647	–2.753	0.649	–1.764	–0.554
10 particles	–3.468	–1.635	–0.367	–0.470	–1.374	0.548	0.595	0.619	0.696
15 particles	–2.637	–3.636	–0.648	0.163	0.583	0.456	0.623	0.704	0.702

Table 17 Max error for Gaussian distribution PSO

7 *Iterations*	*1*	*2*	*3*	*4*	*5*	*10*	*15*	*30*	*40*
1 particle	0.475	0.563	0.512	0.526	0.512	0.516	0.556	0.424	0.488
2 particles	0.436	0.363	0.326	0.286	0.239	0.243	0.227	0.163	0.181
3 particles	0.373	0.363	0.253	0.252	0.225	0.236	0.193	0.135	0.126
4 particles	0.235	0.276	0.236	0.165	0.206	0.155	0.175	0.115	0.114
5 particles	0.286	0.222	0.231	0.212	0.182	0.172	0.124	0.126	0.135
10 particles	0.263	0.152	0.138	0.126	0.157	0.116	0.110	0.103	0.095
15 particles	0.174	0.136	0.128	0.109	0.093	0.056	0.034	0.041	0.036