engineering proceedings

MDPI

Proceeding Paper

Calculation of Neural Network Weights and Biases Using Particle Swarm Optimization †

Jerin Paul Selvan * and Girish Pandurang Potdar *

Computer Engineering, Pune Institute of Computer Technology, Savitribai Phule Pune University, Pune 411043, Maharashtra, India

* Correspondence: jerinsprograms@gmail.com (J.P.S.); gppotdar@pict.edu (G.P.P.)

† Presented at the International Conference on Recent Advances in Science and Engineering, Dubai, United Arab Emirates, 4–5 October 2023.

Abstract: Various machine learning techniques and algorithms have been used to address, and are still being used to tackle, several real-world issues. One technique that has been extensively employed to address a variety of issues is the usage of neural networks. Neural networks can be used to classify data and to calculate regression coefficients. Backpropagation is the cornerstone of neural network training. The process of iteration involves changing the weight of a neural network in response to the rate of error observed in the preceding epoch. The error rates can be reduced and the applicability of the model increased, both of which will increase the model's dependability. Artificial neural networks are commonly trained using the backpropagation approach, also known as backward propagation of mistakes. This technique aids in figuring out a loss function's gradient for every network weight. The backpropagation method divides the dataset into training and testing sets. The neural network is assisted in performing exploration and exploitation using a variety of techniques. Among them are algorithms with biological inspiration. By using a different approach, bio-inspired computing can be distinguished from other traditional algorithms. Simple rules and individual life forms or swarms of individuals that adhere to those rules make up the ideology of bio-inspired computing. These living things, also referred to as agents, develop over time and advance with fundamental imperatives. This approach can be categorized as bottom-up or decentralized. In this paper, a neural network is created using weights and biases determined using the swarm's individual particles. To compare a few parameters between the particle swarm optimization and backpropagation in neural networks, the Pima Indian diabetes dataset is employed.

Keywords: neural networks; weights; biases; particle swarm optimization; bio-inspired algorithms

Citation: Selvan, J.P.; Potdar, G.P. Calculation of Neural Network Weights and Biases Using Particle Swarm Optimization. *Eng. Proc.* **2023**, *59*, 190. https://doi.org/10.3390/engproc2023059190

Academic Editors: Nithesh Naik, Rajiv Selvam, Pavan Hiremath, Suhas Kowshik CS and Ritesh Ramakrishna Bhat

Published: 18 January 2024

1. Introduction

Every connected input and output unit in a neural network (NN) has a weight set by the computer programs that use it. A neural network is made up of many of these connected input and output units. The starting values are chosen at random. The weights and biases used to connect the nodes in each individual layer of the neural network are trained using a training set, which represents a sizeable portion of the original dataset. The two error functions that are primarily utilized for backpropagation are error of mean squared in short MSE and the error of root mean squared in short RMSE. The neural network modifies the weights and biases by utilizing backpropagation. However, employing the neural network with backpropagation does have a few drawbacks. In order to generate the best discriminant for classification or the best function for regression-value prediction, it takes a lot of computational cost and data to tweak the weights and biases. Second, the model is capable of entering the local minima without being able to exit it. Another commonly observed issue is the number of epochs required for network training. For training, which frequently takes a long period of time, more data are needed.

There must be a categorization of each data point into one of the "N" classes in classification tasks. Regression issues, on the other hand, have the goal of producing a certain output value for a given input. The NN generates a discriminant function that separates the classes in order to achieve this. For instance, a network with a single linear output can resolve a two-class problem by learning a discriminant function that is larger than zero for class A and smaller than zero for class B.

The ability of a swarm to forage for food is its most noticeable feature. In computer terminology, this conduct is referred to as exploration and exploitation. The utilization of this behavior allows machine learning algorithms to break out of the local minima and reduce their search time by making use of particle communication. One such popular algorithm based on particle behavior is the particle swarm optimization (PSO) algorithm. It can be utilized for neural network optimization and calculation of the ideal weight and bias distribution. The backpropagation algorithm used with neural networks has a propensity to enter local minima, which lowers the algorithm's overall performance. The integration of PSO and NN aims to highlight the differences in NN performance between networks built using backpropagation and training data and those built using the PSO method.

Perceptrons, Feed Forward NN, Long and Short Term Memory, Multilayer Perceptrons, Convolutional NN, Radial Basis Functional NN, Recurrent Neural Nets, and the Feed-Forward with backpropagation are a few examples of several types of NN. This implementation takes advantage of the Feed Forward NN.

2. Materials

When creating an NN for a problem or application, picking the right NN architecture is typically a crucial step. Usually, a trial-and-error approach is used to do this, where the network is trained and tested before the number of layers is selected based on past performance. It is crucial to be aware that there are noises in the data, such as irrelevant features, redundant features, and outliers, as the architecture of NN is based on the training set. Because of this, it is crucial to pre-process the data before training the NN. One pre-processing method used to address this issue is feature selection (FS) [1].

In FS, a subset of the features is chosen to be included in a model for classification. The basic idea behind the attribute-selection process is that, since the dataset is composed of numerous attributes that are unnecessary and can be trimmed, deleting them will not cause the dataset to lose any information. This can improve the predictor's performance even more, enabling an accurate prediction. Redundancy is a difficult notion to understand because an attribute can only be redundant if it also includes a redundant feature. The crucial thing to remember is that feature selection always yields a smaller subset of features than the original attributes in the set because it only returns the best subset of attributes from a bigger attribute set. When working with datasets that have a large set of attributes or variables, attribute selection is very crucial. It has the benefit of making the predictor more accurate, quicker, and less expensive [2]. The filter model and the wrapper model are two models that can be used to classify FS techniques. Filtering models, which include statistical techniques like discriminant analysis (DA), principal-component analysis (PCA) that makes use of eigen values and eigen vectors for attribute reduction, factor analysis (FA), and independent component analysis (ICA), are mostly indirect measurements of performance that are guided by distance. Although this approach is quick, the subset of functions it produces might not be the best. After computing the classification accuracy using the classification algorithm, the wrapper model selects a subset of features using various selection techniques and assesses the outcomes [3].

Each PSO particle is responsible for selecting a subset of attributes and then figuring out the optimal neural network design based on those features. The PSO algorithms are able to carry out FS. With the PSO algorithm's exploration and exploitation methods, all PSO particles cooperate to find the optimal feature and neural network design. This has the optimal number of layers and neurons in each layer. PSO can overcome the limitation

of being restricted to local minima by utilizing particle mutations depending on mutation probability. A feature is represented as a feature vector using the input [1].

The NN architecture can also be optimized using the PSO algorithm. In a multi-modal landscape, the standard PSO algorithm struggles to converge to a local optimal result. To solve this problem, a dynamic multi-layer PSO (DMLPSO) can be utilized. Important information dynamically interacts with many swarms, increasing population variety and improving performance. Particle swarm optimization provides numerous benefits over other evolutionary computation techniques, including simplicity, quick convergence, and resilience. The PSO also excels at solving problems involving global optimization. Through cross-layer cooperation, multi-layer PSO may thoroughly explore multi-modal areas by introducing multi-layer search algorithms to add more search layers. PSO uses second-order dynamic equations, a type of discrete system, to update the position of the particles. Two techniques that can be utilized for the DMLPSO are multi-layer searching and dynamic reorganizing. The search solution space for each particle is determined by a number of factors, such as the best location for the particle itself, the best location for its lower sub-swarms, and the best location for its upper sub-swarms. For DMLPSO, the authors have used its parameters with the following values: number of layers = 4, number of initial particles = 64, SPD = (1, 4, 4, 4), $\text{Velocity}_{\max} = 3$, $\text{Velocity}_{\min} = -3$ [4].

The PSO algorithm is a Bionic Algorithm because it is intelligent. It is a full-featured random search method with few parameters. The backpropagation (BP) network can fix some flaws and become more adaptable by optimizing its weight values and thresholds in conjunction with a BP neural network. The authors of this paper [5] suggest an immunization algorithm that, through iterations, inoculates the undesirable particles in the population while extracting the vaccine from the population's best particles. A random suspension method is used to suspend the currently optimal individual particles in a particle swarm in order to prevent premature PSO. Immunological memory refers to the immune system's ability to retain memory cells that produce antibodies in response to foreign antigens. When the same antigen invades again, associated memory cells are activated and produce a large number of antibodies. When properties like variety and immunological memory are incorporated into this PSO method, the algorithm improves its ability to conduct global searches rather than settling for local solutions. It is more likely that lower antibody concentrations will be chosen. However, the selection probability decreases with increasing antibody concentration. The authors use the following initial values for the BP-NN, (net training parameter goal = 0.0001), the count of nodes in the hidden layer nodes is five, and the learning rate net training parameter *lr* = *0.1*. In the PSO part, $v_{max} = 1.2$, $v_{min} = 0.4$, c1 = c2 = 2.05, population size pop_size = 40, evolution times max gen = 100 [3].

A thorough examination of the classification algorithm using NN is provided in the paper [6]. The study's goal was to provide a novel hybrid neural classification method for enhancing classic NN's efficacy and accuracy. According to the authors' research, the suggested algorithm outperforms the neural network algorithms of Adam, Random Forest, Gradient Boosting Machine, Lasso, Ridge, Linear, and Quadratic discriminant analysis, as well as logistic regression and Lasso regression in terms of accuracy and stability, but it also takes longer to process.

The convergence rate of the BP-NN is a little slow, yet it can capture the solution with local minima. Under these conditions, PSO with NN can be used to train a feed-forward neural network. The Nash–Sutcliffe and correlation coefficients can be used to evaluate a model's performance. The network may sometimes become stuck when there is another set of neuronal weights in the weight space whose expense function is significantly lesser than the local minimum. While modelling an NN, there are many more elements that must be taken into account, some of which are as follows:

1. Count of input nodes;
2. Count of hidden nodes;
3. Count of output nodes;
4. Learning rate;
5. Count rate;
6. Bias matrix;
7. Minimum error (function used for error estimation);
8. Activation or transfer function.

All of these factors also have an effect on the convergence of BP-NN training. In comparison to GA, the authors found the PSO approach to be simple with less iteration and fewer variables for control [7–9].

The diabetic retinopathy dataset is used by the authors of [10]. They presumptuously use 10-way cross-validation to partition the dataset into 10 equal sections. Each component will be used as test and training data. Each feature or characteristic is then chosen using the PSO approach after division. Following the selection of the ideal characteristics or traits, neural network techniques are utilized to classify the training data. On the other hand, test data with chosen characteristics are verified based on training data, and a neural network is the technique employed in this validation. The final phase is the presentation of the data-classification findings after the training and test data have been validated using the neural network.

Deciding on the ideal count of the hidden neurons in each intermediate layer and the connection weight in a neural network at the same time is considered a challenging task. This is due to the fact that modifying the hidden neurons drastically changes the topography of the network, making training more challenging and necessitating unique considerations. The authors of [11] propose a PSO with Levy flight-based multiverse optimization (MVO). PSO is a brand-new, quick algorithm that improves the harmony between development and exploration by preventing early convergence. Because of its quick convergence and simplicity of use, it is among the most important meta-heuristic algorithms.

The authors of the work [12] suggest an upgraded PSO that greatly outperforms (in terms of the global search) the original PSO algorithm and also the solution fine tuning. This improved PSO makes use of parameter automation techniques, velocity resets, crossings, and mutations. Improvements like parameters that rely on varying time and perturbations at random like velocity resetting have been proposed to help overcome these limitations of the traditional PSO. Additionally, mutation and crossover are used to create diversity. Before using a BPN to solve a problem, the authors of [13] present a technique that states that the BPN's parameter settings, such as the number of hidden layers, rate of learning, impulse term, number neurons in the hidden layer, and cycles of learning, must be established. To prevent building sub-optimal network models, which can drastically increase computing cost and decrease results, the parameterization of the network architecture should be carefully considered. The authors claim that by choosing or de-noising pertinent features, the Rate Classifier's classification accuracy can be increased. The authors use the following parameters with the respective values: $c_1 = 0.8$, $c_2 = 1.5$, $w = 0.9$, and $population_{size} = 10$ [9].

3. Methodology

The error functions serve as the foundation for the NN's BP. In accordance with the results of the error functions, the weights and biases are modified. The mean squared error (MSE) for problems requiring real valued results (Regression) and the cross entropy for problems requiring classes (Target or labels) are the two most widely utilized error functions. The error function in this instance is the cross entropy [11]. Given two input–output pairs, the mean square error function is as follows: $U = (\vec{u_1}, \vec{v_1}), (\vec{u_2}, \vec{v_2})$.

U is the position vector indicated in two dimensions and an ANN with some parameters represented as $\boldsymbol{\theta}$ that produces output in the form $\boldsymbol{g_\theta}(\vec{\boldsymbol{u}})$; $\boldsymbol{g_\theta}(u)$ for input $\vec{\boldsymbol{u}}$ u; the error function $E(U, \boldsymbol{\theta})$ is

$$\boldsymbol{E}(\boldsymbol{U},\ \boldsymbol{\theta}) = \frac{(v_1 - g_\theta(u_1))^2}{2} + \frac{(v_2 - f_\theta(u_2))^2}{2} \tag{1}$$

One algorithm that is bio-inspired and can be used to find the best arrangement within the arrangement space is PSO. It differs from previous optimization computations in that it needs an objective function and is not dependent on the objective function's slope or other differential shape. Moreover, there are remarkably few hyper-parameters in it [14].

The optimal application of PSO is to determine the maximum or minimum of a multidimensional vector space-defined function $\boldsymbol{g_\theta}(u)$. If $\boldsymbol{g_\theta}(u)$ is a function that takes a vector parameter, like the coordinates (u, v) in a two-dimensional plane, and returns a real result that can take on almost any value in search space; for instance, $g(u)$ can calculate the altitude for any point on the plane—then PSO can be used. The parameter u that the PSO algorithm determines yields the minimal $g(u)$ that will be returned. The direction of travel is always towards the search of the worldwide optimal. Every particle dynamically modifies its travelling velocity based on its prior flying experiences as well as those of its group members [15,16]. First, each particle attempts to monitor its own best result, which it refers to as its personal best or $\mathrm{p_{best}}$.

Second, the best over the entire search space or g_{best} is the most promising value for any particle. Based on its position vector, the velocity vector of the particle at $\mathrm{T_{current}}$, the magnitude of the distance between position vector $p_{current}$ and p_{best}, and, lastly, the magnitude of the distance between its position $p_{current}$ and g_{best}, each particle adjusts its position [17].

The formula that updates the velocity of the particles is

$$\boldsymbol{V_i}^{t+1} = \mathbf{W}.\boldsymbol{V_i}^t + c_1\boldsymbol{U_1}^t(\boldsymbol{P_{b1}}^t) - \boldsymbol{P_i}^t + c_2\boldsymbol{U_2}^t(\boldsymbol{g_b}^t - \boldsymbol{P_i}^t) \tag{2}$$

where the particle or agent's velocity is denoted with $\boldsymbol{V_i}$, the population of agents is denoted with A, the inertia weight is denoted with W, the cognitive constant is denoted with c_1, c_2 the particle or agent's position is denoted with $\boldsymbol{X_i}$, the personal best is denoted with P_b, and the global best is denoted with g_b.

The movement of the particles is updated using the following formula:

$$\boldsymbol{P_i}^{t+1} = \boldsymbol{P_i}^t + \boldsymbol{v_i}^{t+1} \tag{3}$$

The count of neurons in the input layer, the count in the output layer, the count in the hidden layer, and the count of hidden layers determine the architecture of the neural network. Values are first randomly inserted into the weight and bias vectors. Each particle in the PSO is assigned a neuron in the NN [17,18]. The particles will then move around randomly within their own neighborhoods until they reach a better position where the sum of the squares of errors between the model predictions and actual data values are minimized. This continues until the stopping criteria, or the number of iterations in this case, is completed. At the end, an optimal weight and bias matrix is obtained. Figure 1 shows the flow of the training using PSO with NN and the pseudo code is mentioned in Algorithm 1.

The models' performance is measured using a few different performance scores. The number of times a model properly classifies a positive sample as positive is known as the true positive combination. The combination of false negative indicates the number of times a model misclassifies a positive sample as negative.

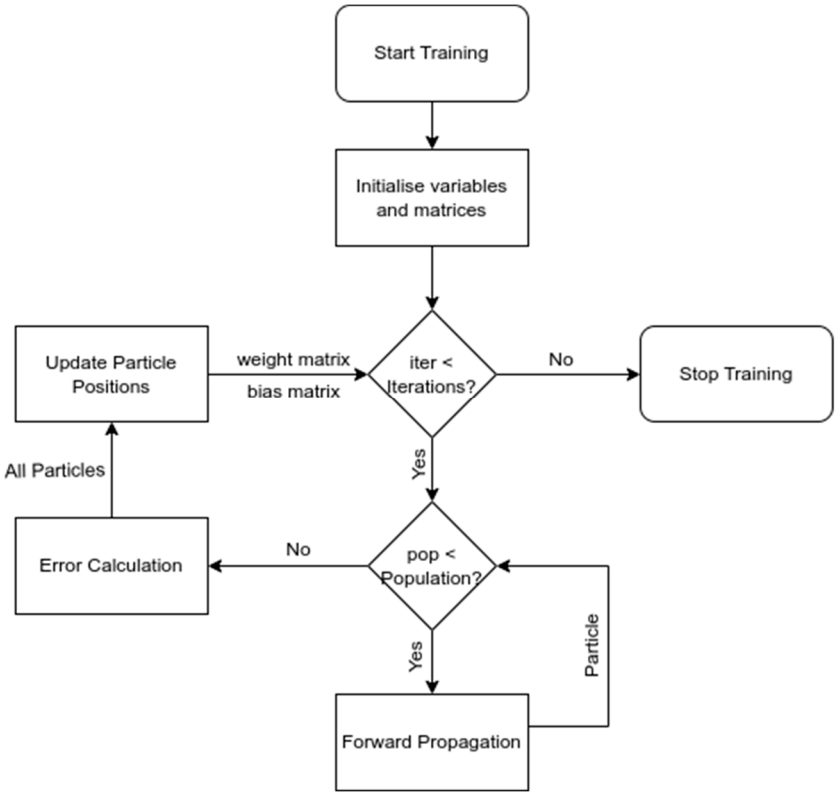

Figure 1. Flow chart of the training phase of the algorithm.

The term false positive refers to the number of times a model classifies a negative sample as positive in error. The number of times a model properly classifies a negative sample as negative is known as true negative.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F\_score = \frac{2\,.\,Precision\,.\,Recall}{Precision + Recall} \tag{6}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

The visualization of the point at which the model becomes confused when distinguishing between two classes is aided by the Confusion Matrix Table 1. A 2×2 matrix with the actual truth labels represented in the row and the predicted labels in the column provides a clear understanding of it.

Algorithm 1. Calculating weight and bias matrix using PSO.

```
c1←0.5
c2←0.3
w←0.9
n_particles←150
iterations←1000
weights←random vector
bias←random vector
while N < iterations
    popCount←0
    while popCount < nparticles
    res←forwardPropagation()
    popCount←popCount + 1
    end while
    error←errorEstimate(n_particles, res)
    n_particles←adjustParticlePosition(n_particles, error)
    N←N + 1
end while
weights←n_particles
bias←n_particles
```

Table 1. Confusion matrix.

	Positive	**Negative**
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

4. Results and Discussion

The dataset used for the study is a benchmark dataset called Pima Indians Diabetes Dataset. The datasets contains various medical-related predictors or variables that are independent and one class or variable that is dependent. The target is labeled as "Outcome". The independent variables include the following: "number of pregnancies", "glucose", "blood pressure", "skin thickness", "insulin level", "BMI", "DiabetesPedigreeFunction", and "Age". The correlation coefficient is between −1 and 1, where 1 denotes a very positive correlation and -1 a significantly negative correlation. Both of these extremes indicate that the features are redundant. For instance, in this case, the feature "pregnancies" and "age" are positively correlated with the score of 0.54. However, if the FS pre-processing step is applied, depending on the threshold (usually kept greater than 0.8) either of the features can be removed. This dataset has been taken from the UCI Machine Learning repository and contains 9 features and 768 instances [19].

The neural network is trained by making use of the BP algorithm and the particle swarm optimization algorithm. Various scoring methods are used for comparing the performance of the two algorithms.

Tables 2 and 3 show various scores of the NN with BP and with the particle swarm optimization algorithm over 1000 and 1500 epochs, respectively.

Table 2. Accuracy of NN with PSO over multiple iterations.

Initial Population	**Iteration**	**Accuracy**
100	5000	65%
150	1000	80%
200	1000	75%
150	2000	80%
150	500	75%

Table 3. Accuracy of NN with BP over multiple iterations.

Iteration	Accuracy
500	64.99%
1000	65%
1500	75%
3000	69.99%
5000	65%

5. Conclusions

The PSO algorithm uses exploration along with exploitation making it capable of coming out of the local minima. This helps in solving problem of the NN not being able to explore the search space when stuck in local minima. The BP is not effective in distributing optimal weights and biases between the nodes as compared to the PSO algorithm. The PSO algorithm has performed better than the NN with backpropagation in terms of accuracy when using it to calculate the weight and bias matrix for the NN architecture. The quality of the performance of PSO is much better than NN with BP, since the precision of PSO with NN is better. The PSO algorithm requires only 1000 iterations with 150 as the initial population to give an accuracy of 80%, whereas NN with BP requires 1500 iterations to give the best result, which is 75%. This indicated that the NN with PSO shows fast convergence. As part of future scope, the NEAT algorithm with PSO can be used as a hybrid algorithm to create a topology for the NN along with the calculation of optimal weights and biases for the NN. This can help in improving the performance of the NN.

Author Contributions: Conceptualization, J.P.S. and G.P.P.; methodology, J.P.S. and G.P.P.; software, J.P.S. and G.P.P.; validation, J.P.S. and G.P.P.; formal analysis, J.P.S. and G.P.P.; investigation, J.P.S. and G.P.P.; resources, J.P.S.; data curation, J.P.S.; writing—original draft preparation, J.P.S. and G.P.P.; writing—review and editing, J.P.S. and G.P.P.; supervision, G.P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the data used are made available in the present work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shee, B.K.; Vipsita, S.; Rath, S.K. Protein Feature Classification Using Particle Swarm Optimization and Artificial Neural Networks. In Proceedings of the 2011 International Conference on Communication, Computing & Security, Odisha, India, 12–14 February 2011; pp. 313–318. [CrossRef]
2. Brito, R.; Fong, S.; Zhuang, Y.; Wu, Y. Generating Neural Networks with Optimal Features through Particle Swarm Optimiza-tion. In Proceedings of the International Conference on Big Data and Internet of Thing, London, UK, 20–22 December 2017; pp. 96–101. [CrossRef]
3. Tianhan, T.; Junsheng, L.; Kun, W.; Renhui, S. Target Threat Assessment Using Particle Swarm Optimization and BP Neural Network. In Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference, Guangzhou, China, 22–24 June 2019; pp. 181–184. [CrossRef]
4. Cai, M.; Liu, P.S.; Wang, Y.B. Optimizing Floating Centroids Method Neural Network Classifier Using Dynamic Multi-layer Particle Swarm Optimization. In Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 15–19 July 2018; pp. 394–401. [CrossRef]
5. Li, S. Robot Path Planning Algorithm Based on Particle Swarm Optimization and Feedforward Neural Network in Network Environment. In Proceedings of the 2020 4th International Conference on Artificial Intelligence and Virtual Reality, Kumamoto, Japan, 23–25 October 2020; pp. 66–68. [CrossRef]
6. Yang, H.L.; Li, B.-Y. A Hybrid Neural Network based on Particle Swarm Optimization for Predicting the Diabetes. In Proceedings of the 2021 10th International Conference on Software and Computer Applications, Kuala Lumpur, Malaysia, 23–26 February 2021; pp. 302–306. [CrossRef]

7. Kuok, K.K.; Harun, S.; Shamsuddin, S.M. Particle swarm optimization feedforward neural network for modeling runoff. *Int. J. Environ. Sci. Technol.* **2010**, *7*, 67–78. [CrossRef]
8. Wu, J.; Chen, J.; Gu, L. Alternative Combination of Improved Particle Swarm and Back Propagation Neural Network. In Proceedings of the 2010 Sixth International Conference on Natural Computation, ICNC, Yantai, China, 10–12 August 2010; pp. 75–78.
9. Lin, S.W.; Chen, S.C.; Wu, W.J.; Chen, C.H. Parameter determination and feature selection for back-propagation network by particle swarm optimization. *Knowl. Inf. Syst.* **2009**, *21*, 249–266. [CrossRef]
10. Herliana, A.; Arifin, T.; Susanti, S.; Hikmah, A.B. Feature Selection of Diabetic Retinopathy Disease Using Particle Swarm Op-timization and Neural Network. In Proceedings of the 6th International Conference on Cyber and IT Service Management CITSM, Parapat, Indonesia, 7–9 August 2018; pp. 196–199. [CrossRef]
11. Bousmaha, R.; Hamou, R.M. A Amine, Automatic selection of hidden neurons and weights in neural networks for data classification using hybrid particle swarm optimization, multi-verse optimization based on Lévy flight. *Evol. Intell.* **2021**, *15*, 1695–1714. [CrossRef]
12. Yu, J.; Xi, L.; Wang, S. An Improved Particle Swarm Optimization for Evolving Feedforward Artificial Neural Networks. *Neural Process. Lett.* **2007**, *26*, 217–230. [CrossRef]
13. Garro, B.A.; Vázquez, R.A. Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms. *Comput. Intell. Neurosci.* **2015**, *20*, 61. [CrossRef] [PubMed]
14. Wang, Q.; Yan, K.; Wan, X.; Yuan, M. Application of Particle Swarm Optimization in Fussy Neural Networks. In Proceedings of the 2009 Fifth Inter-national Conference on Information Assurance and Security, Xi'an, China, 18–20 August 2009; pp. 158–161. [CrossRef]
15. Hajare, P.R.; Bawane, D.N.G. Feed Forward Neural Network optimization by Particle Swarm Intelligence. In Proceedings of the 2015 7th International Conference on Emerging Trends in Engineering & Technology (ICETET), Kobe, Japan, 18–20 November 2015; pp. 40–45. [CrossRef]
16. Smith, J.W.; Everhart, J.E.; Dickson, W.C.; Knowler, W.C.; Johannes, R.S. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care, Washington, DC, USA, 6–9 November 1988; IEEE Computer Society Press: Washington, DC, USA, 1988; pp. 261–265.
17. Kumar, G.; Singh, U.P.; Jain, S. An adaptive particle swarm optimization-based hybrid long short-term memory model for stock price time series forecasting. *Soft Comput.* **2022**, *26*, 12115–12135. [CrossRef] [PubMed]
18. Kuo, R.J.; Hong, S.Y.; Huang, Y.C. Integration of particle swarm optimization-based fuzzy neural network and artificial neural network for supplier selection. *Appl. Math. Model.* **2010**, *34*, 3976–3990. [CrossRef]
19. UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 3 October 2023).