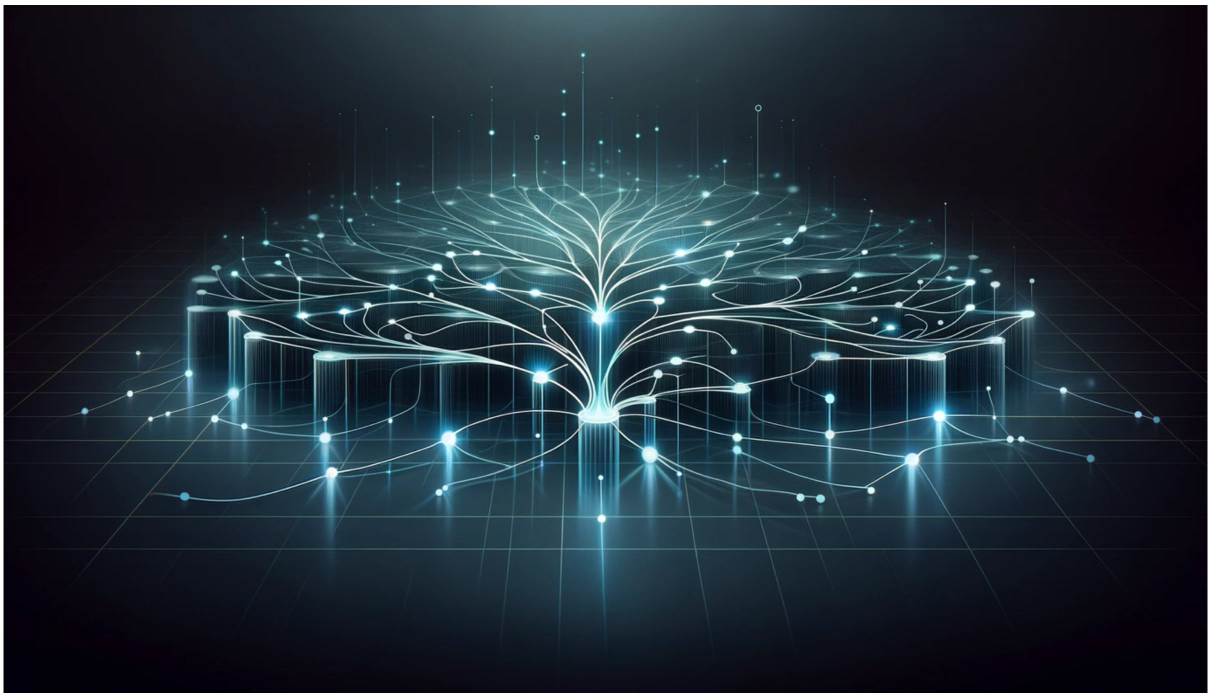
SCHOOL OF MATHEMATICS & COMPUTER SCIENCE

Artificial Neural Network with Particle Swarm Optimization (ANN and PSO)



Prepared by:
Akshay Garg (H00338776)
Irfan Syed (H00389591)

Prepared on:
22th November 2023

Course Name:
Biologically-Inspired Computation (F20BC)

IMPLEMENTATION

Note*: We have two files for ANN with PSO: MainWithInputs and MainWithoutInputs. For testing purposes, please run MainWithInputs as it allows the user to give inputs of their choice, whereas MainWithoutInputs has predefined input values set so that the user can test whether the algorithm is working or not.

In this section, we will deep dive into the practical aspects of our project, where we have developed an Artificial Neural Network (ANN) integrated with Particle Swarm Optimization (PSO) from scratch. Our approach involved the development of six fundamental classes where each class is designed to represent different entities in ANN. These classes are Activation, Loss, Layer, NeuralNetwork, ANNBuilder, Particle, and PSO each playing a key role in the ANN's learning and optimization process. Here, we will summarize the purpose and structure of these classes, providing insights into how they collectively form an effective ANN model and are optimized using PSO techniques.

Loss Class: The `Loss` class is used for evaluating the performance of the optimizer or the developed through different loss functions. These functions calculate the difference between predicted and actual values. Also, functions are defined as static to enable direct application without the need to instantiate the `Loss` class. The functions implemented in this class are Mean Absolute Error (MAE), Mean Squared Error (MSE), Binary Cross Entropy, Categorical Cross Entropy, and Hinge Loss.

Activation Class: The `Activation` class is a collection of activation functions which are essential in the output of neural networks. Activation functions help to learn complex patterns. These functions are implemented as static methods which helps us in direct usage without the need for class instantiation. This includes the following activation functions: Sigmoid, ReLU (Rectified Linear Unit), Hyperbolic Tangent (tanh), and Softmax.

Layer Class: The Layer class is important for creating neural network layers. Its constructor initializes the layer with specified inputs, number of nodes, and activation function, and sets a random seed. During a forward pass, this class calculates the weighted sum of inputs, weights, and bias, applying the selected activation function to obtain activated node values. Also, this class provides functions to flatten and return weights and bias for external usage. The update function facilitates the sequential update of weights and biases across layers during the training process.

Neural Network Class: The NeuralNetwork class is used for the building and evaluation of neural networks. The constructor of the class initializes the network with an empty list of layers, and the append function is used to add new layers. The threshold_function applies a threshold to determine activated nodes. During a forward pass, the forward function computes the network's output as the evaluate function calculates accuracy and loss. The weightBiasList function retrieves parameters. The update function updates weights and biases for each layer in the neural network. This class contains functions/methods for creating dynamic networks.

Ann Builder Class: The ANNBuilder class is used to build an artificial neural network (ANN) with different layers, the number of nodes for each layer and their activation functions. The build function initializes the network and appends layers based on the parameters passed in the function.

Particle Class: The Particle class is used in the Particle Swarm Optimization (PSO) algorithm which represents the behaviour of individual particles within the optimization process. The constructor is used to initialize variables like position, velocity, boundaries, personal best position, personal best fitness, fitness, and informants. The set boundary function sets the exploration boundary, and the initialize function sets the initial position, velocity, and personal best position within these bounds.
One of the important functions is updateVelocity in which we dynamically adjust the particle's velocity by using cognitive, social, and global components, controlling its movement within the exploration boundary. The updated position function ensures the particle's position is updated and stays within specified bounds. The setInformants function randomly selects informants from a provided list.

PSO Class: This class combines the Particle Swarm Optimization (PSO) algorithm with neural network training. The constructor is used to initialize variables like population size, particles, boundaries, neural network, and the number of informants. The initialize function creates particles and sets exploration boundaries and informants for the optimizer. The `train` function handles the training process and tracks important metrics like loss, accuracy, fitness, positions, etc through each epoch. It updates particles, evaluates fitness, and dynamically changes global and personal best positions. Updates for Velocity and position are based on optimization parameters. Also, it updates informants and the neural network with the best global position.

EXPERIMENTAL INVESTIGATION

During the development of an artificial neural network (ANN) using Particle Swarm Optimization (PSO), selecting hyperparameters is important for optimizing the accuracy of the network. Our experiment was focused on important hyperparameters that impact the efficiency and accuracy of the ANN.

Different Parameters to Investigate

1. **Network Structure:** The network structure includes the number of layers, neurons in each layer, and the activation functions of neurons. The structure of the network directly affects its ability to learn and predict from data.

2. **PSO Parameters:** These include population size, the number of informants per particle, and coefficients. These parameters are vital in the PSO algorithm for finding the optimal set of network weights.

Values for parameters to find optimal values

For the hyperparameters in an ANN optimized with PSO, the following numerical ranges are considered based on the literature and general practices:

- **Number of Neurons in Each Hidden Layer:** The range for the number of neurons is generally set between 10 to 100. This range is chosen because it allows for adjusting the network’s complexity according to the complexity of the problem. More complex problems might require a larger number of neurons.

- **Number of Epochs:** The range for epochs while training the neural network is between 20 to 100. Fewer epochs can result in underfitting, as the network might not learn enough from the data whereas if there are too many epochs, this can lead to overfitting where the model performs well on training data but poorly on unseen data.

- **Number of Layers:** The number of layers in the neural network is chosen between 1 to 3. The choice depends on the complexity of the problem, simpler problems may require fewer layers, while more complex ones might need more layers.

- **Number of Particles in Swarm:** The default number of particles in a swarm can be set upto 50 which enables a comprehensive evaluation of different sets of neural network weights.

These ranges can be adjusted based on specific dataset characteristics and the problem. It's important to conduct experiments to find the optimal values within these ranges for specific applications. We will be conducting experiments using these specified values to optimize and evaluate our neural network models.

Exp Num	Num Layers	Num Epochs	Num Nodes	Num Particles	Num Informants	Coefficients
Exp 1	1	50	20	15	4	0.8, 0.8, 2.4
Exp 2	2	70	50, 20	25	5	2.4, 0.8, 0.8
Exp 3	3	100	30, 50, 30	25	5	0.8, 2.4, 0.8
Exp 4	2	150	40, 30	35	6	1.4,1.4,1.2
Exp 5	3	80	20, 40, 10	50	7	1.5, 1.5, 1
Exp 6	2	10	34, 22	35	6	1, 1, 2

RESULTS

Configuration	Avg Accuracy	Avg Loss
EXP 1	99.45%	0.0493
EXP 2	98.54%	0.0424
EXP 3	96.91%	0.1964
EXP 4	99.82%	0.0187
EXP 5	98.00%	0.1212
EXP 6	90.72%	0.2752

DISCUSSION AND CONCLUSIONS

Upon trying out the above six experiments evaluating different hyperparameters of our artificial neural network and particle swarm optimization, the ANN was able to learn the underlying structures of the dataset and was able to generalize well to the new, unseen testing data. We observed the following from our test cases.

1. **Layer and Node complexity:** The networks with a larger number of layers and nodes showed better performance. This could be because of the ANNs ability to comprehend complex patterns within the data. However, it is important to balance the complexity to prevent overfitting, in which the model completely memorizes the training data, without actually learning its structure, which was not seen as much in our tests.

2. **Epochs:** The duration of training epochs had a positive effect on network performance up to a certain point. For example, EXP 4 with 150 epochs performed better than others, while Experiment 6, which was limited to only 10 epochs, didn't perform well, showcasing why a minimum number of epochs is necessary to prevent underfitting and allow the model to learn the associations in the data.

3. **Generalization:** EXP 1 showed minimal discrepancy in testing and training sets, indicating that the ANN generalized well. On the other hand, the low performance in EXP 6, with the lowest accuracies, shows inadequate learning, potentially due to poor training epochs.

4. **Population and Informants:** Larger population and informants yield better optimization, but can prove to be computationally expensive.

In conclusion, the experiments prove the essential role of hyperparameters in the performance of ANNs. Better network complexity and longer epochs contribute to better model performance but must be carefully configured to prevent overfitting. In optimization, PSO hyperparameters play an essential role in how the model improves over time, the coefficients need to be balanced in order to have a balance between exploration and exploitation. Finally, it is important to consider both loss and accuracy in performance evaluations to gain a proper understanding of a model's capabilities.

REFERENCES

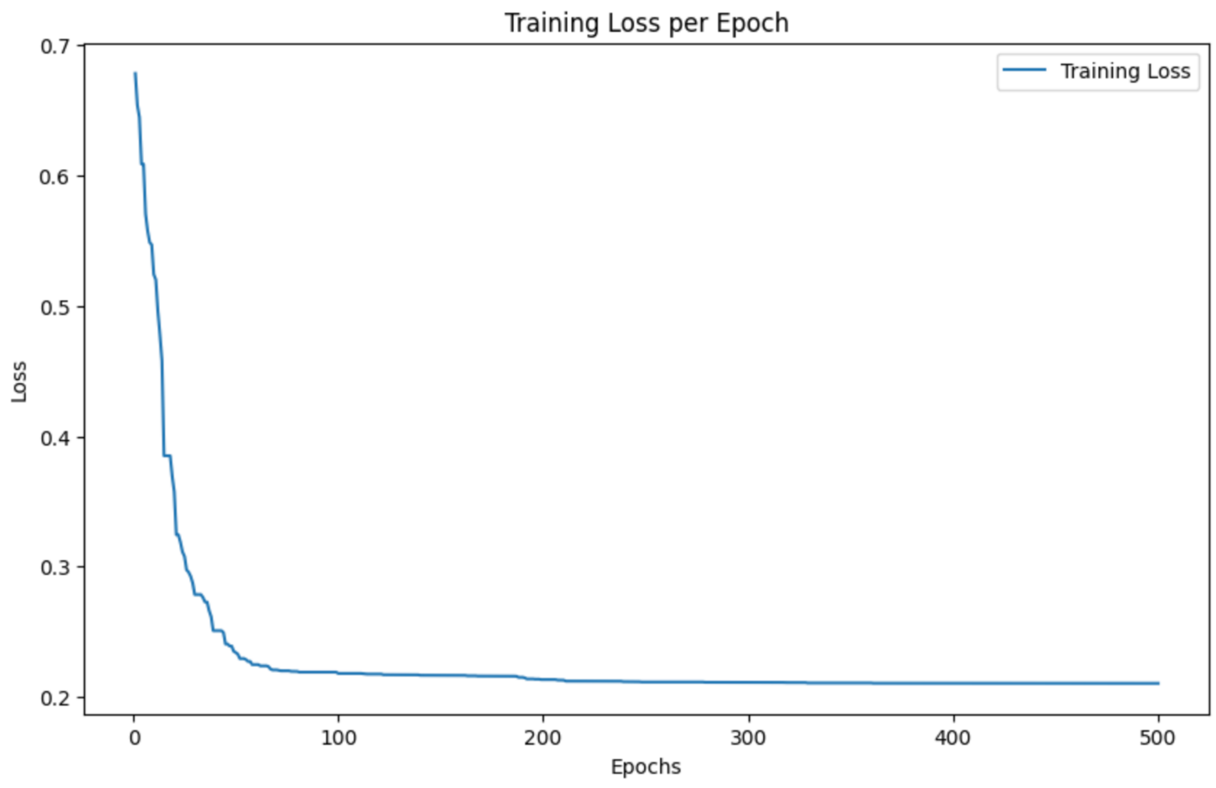
Zaghloul, L., Zaghloul, R. and Hamdan, M. (1970) *Optimizing artificial neural network for functions approximation using particle swarm optimization*, *SpringerLink*. Available at: https://link.springer.com/chapter/10.1007/978-3-030-78743-1_20#:~:text=PSO%20and%20ANN%20have%20several,particle%2C%20and%20the%20acceleration%20coefficients (Accessed: 24 November 2023).

Ye, F. *Particle swarm optimization-based automatic parameter selection for deep neural networks and its applications in large-scale and high-dimensional data*, *PLOS ONE*. Available at: https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0188746 (Accessed: 24 November 2023).

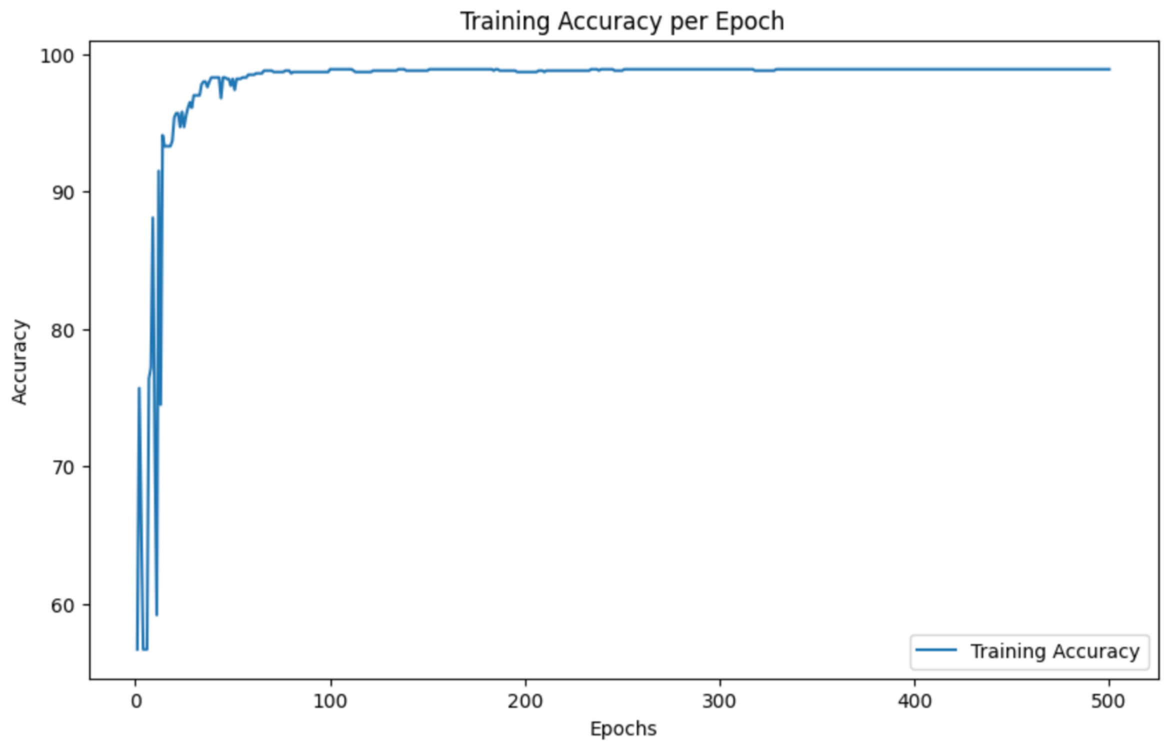
PSO parameters and hyperparameters (no date) *R*. Available at: https://search.r-project.org/CRAN/refmans/automl/html/pso.html#:~:text=,weights%20sets%20will%20be%20tested (Accessed: 24 November 2023).

Rendyk (2023) *Tuning the hyperparameters and layers of neural network deep learning*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/05/tuning-the-hyperparameters-and-layers-of-neural-network-deep-learning/ (Accessed: 24 November 2023).

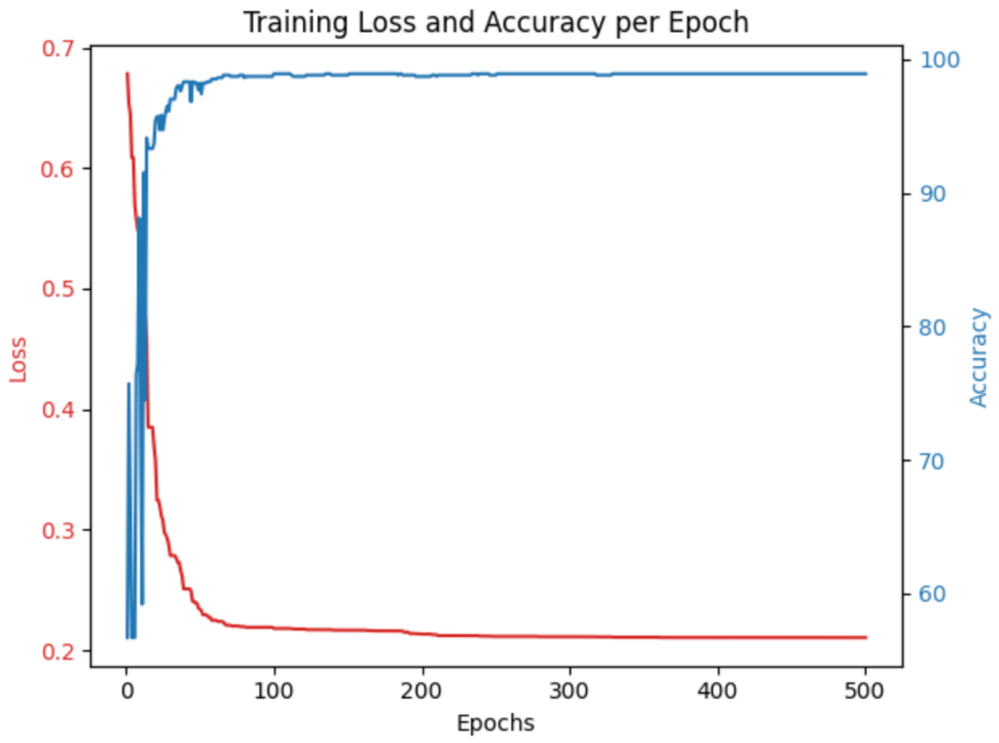
APPENDIX



Training Loss per Epoch
Training Loss
Loss
0.7
0.6
0.5
0.4
0.3
0.2
0
100
200
300
400
500
Epochs



Training Accuracy per Epoch
Training Accuracy
Accuracy
100
90
80
70
60
0
100
200
300
400
500
Epochs



Training Loss and Accuracy per Epoch
Loss
0.7
0.6
0.5
0.4
0.3
0.2
Accuracy
100
90
80
70
60
0
100
200
300
400
500
Epochs