# COURSEWORK 1

F20BC – BIOLOGICALLY INSPIRED COMPUTATIONS

*Raghu, Dhruv  Thomas, Rithin*

## IMPLEMENTATION

The ANN implementation is split into 4 classes: Activation, Neuron, Layer, and Network. The weights of each layer are initialized using He initialization which has proved to impact convergence in neural networks. The input and output layers are automatically generated based on the input data and the class labels. The output layer is created using the logistic activation function by default considering we are using the network for classification. The forward method in the neural network object passes a row of input data through the network and saves the raw output from the final layer. The prediction method passes input data row by row through the forward function and converts the raw neuron outputs into a class prediction. The train with PSO function of the network is used to initialize a PSO object with the specified parameters and attempt to tune the weights and bias of the network to classify the input data correctly.

The PSO with the informant's implementation is split into 2 classes, the Particle class and the PSO class itself. The position of each particle in this PSO implementation is a neural network object. While creating the particles, the weights and biases of the neural network object passed to it are randomized so that particles can explore the search space properly and not all start from the same point. The particle fitness is evaluated using cross-entropy loss using the loss function in the neural network class. The algorithm aims to return the network with the lowest fitness. With a few changes to the code, the particle fitness can be evaluated using accuracy calculated on the train set instead of cross-entropy loss. The particle class is also responsible for updating the velocity and the position throughout the algorithm. A reflective bound is used for the particles to avoid large changes in the position of the particle.

The main function of the PSO algorithm is the optimize function. This function iteratively causes the particles to update position and velocity and update global and informant bests if necessary. The PSO algorithm implements informants. The informants are decided by finding 'k' closest neighbors to a certain particle. The closest particles are found by comparing the fitness of two particles. The optimized function of this PSO implementation has a dynamic parameters parameter that can linearly decrease the inertia weight. There is also some code to implement changing multiple parameters based on the iteration. This can be used in more complex problems to balance exploration and exploitation for optimal results.

## EXPERIMENTAL INVESTIGATION

As explained in the Background Theory section, the Artificial Neural Network and Particle Swarm Optimization algorithm have various parameters affecting how data relations are formed and how the neural network learns. Instead of randomly adjusting the 3 ANN parameters and the 7 PSO parameters, which would be inefficient and time-consuming, a plan is devised to optimize a few parameters at a time until the whole network is fully optimized.

### PSO Parameters

Before analyzing the impact of ANN parameters on the network output, it is important to optimize the PSO algorithm.

## Cognitive, Social, and Global Weights

The first test focuses on testing the three coefficients: cognitive, social, and global parameters. According to **<Insert paper>**, a common practice is to configure these coefficients so that they sum up to 4. Considering this guideline, six parameter sets are created, each focusing on a distinct combination of these parameters.

*Table 1: Test sets focusing on different approaches for coefficients.*

| No | Focus | Cognitive | Social | Global |
|---|---|---|---|---|
| 1 | More Emphasis on Global Exploration | 0.8 | 0.8 | 2.4 |
| 2 | More Emphasis on Personal Knowledge | 2.4 | 0.8 | 0.8 |
| 3 | More Emphasis on Swarm Knowledge | 0.8 | 2.4 | 0.8 |
| 4 | Emphasis on Personal and Swarm Knowledge | 1.5 | 1.5 | 1.0 |
| 5 | Equal Importance to all Parameters | 1.3 | 1.3 | 1.4 |
| 6 | Balanced Exploration and Exploitation | 1.0 | 1.0 | 2.0 |

Given the limited size of the banknote authentication dataset (4 features, binary classification), a neural network with a single hidden layer, 5 neurons, and a ReLU activation function is created as a constant. Moreover, the swarm size is set to 25 particles [1] the number of informants is set to 4 [1], the number of iterations is kept at 25 due to the dataset's simplicity, and the inertia is set to 0.5 to strike a balance between exploration and exploitation.

The network is trained using the parameters from the first test set. The accuracy of the test set is computed and stored in a list. Given the stochastic nature of the PSO algorithm, each set is tested 15 times, and the average accuracy is calculated. This process is then repeated for each test set, allowing for a comparison of the accuracies from each run of every test set to identify the optimal parameter set.

## Inertia

To test the optimal value for the inertia parameter, constants from the previous test remain unchanged. However, this time, the algorithm's coefficients are set to the optimal values identified in the previous test, and only the inertia parameter is varied. Five inertia values (0.2, 0.5, 0.8, and 1.0) are tested, and the network undergoes 15 runs for each inertia weight. The average accuracy over 15 runs is calculated and compared for each inertia weight to choose the optimal weight.

## Swarm Size and Number of Iterations

Enhancing the swarm size and the number of iterations generally increases the performance of the neural network. Nevertheless, increasing the parameters too far might result in diminishing returns due to the added complexity, and as a result, increasing runtime. Four test sets are created to determine the most effective balance for the binary classification dataset and the impact of swarm size and iterations on accuracy and runtime is explored.

Table 2: Test set combinations for swarm size and iterations

| No. | Focus | Swarm Size | Iterations |
|---|---|---|---|
| 1 | Small Swarm, Large Iterations | 15 | 75 |
| 2 | Medium Swarm, Large Iterations | 25 | 50 |
| 3 | Medium Swarm, Medium Iterations | 25 | 25 |
| 4 | Medium Swarm, Small Iterations | 25 | 15 |
| 5 | Large Swarm, Small Iterations | 50 | 15 |

Note: The choice of swarm sizes and iterations is determined based on the simplicity of the banknote authentication dataset. What might be a large swarm size for this dataset could be perceived as a small swarm size for a more complicated dataset with a larger search space.

### Informants

As per findings in [1], the ideal number of informants typically falls within the range of 4 to 8. To test this, the network is trained using 2, 6, and 10 informants. If 6 proves to be the optimal number of informants, additional testing can be conducted to update the optimal number of informants within the 4 to 8 range.

### ANN Parameters

When building the Artificial Neural Network (ANN), it's often found that having three layers works well as it can[3] The network was tested with 1, 3, and 5 hidden layers and the selection of 1 hidden layer for the network was more than enough to get a high accuracy on the test set. Adding more layers to the network increases the complexity and the run time while not improving the accuracy substantially.[3] For the neurons in the hidden layers, a good rule of thumb is to use 70% to 90% of the nodes from the input layer.// This helps the network understand patterns without getting too complicated. Different activation functions like ReLU, Sigmoid, Tanh, and leaky ReLU were tested.

## RESULT DISCUSSION

### PSO Parameters

#### Coefficients (Cognitive, Social, Global)

***Figure 1*** shows the fluctuation in the accuracy of the network trained by the PSO algorithm across multiple runs. Given the stochastic nature of the PSO algorithm, an average of multiple accuracies must be. In this scenario, each test set comprises a distinct combination of cognitive, social, and global parameters, while all other parameters remain constant.

Table 3: Different Accuracies for different test sets

| Test Set No. | Avg. Accuracy | Max. Accuracy | Min. Accuracy |
|---|---|---|---|
| 1 | 0.938 | 0.992 | 0.793 |
| 2 | 0.952 | 0.985 | 0.810 |

| | | | |
|---|---|---|---|
| 3 | 0.916 | 0.995 | 0.786 |
| 4 | 0.936 | 0.980 | 0.876 |
| 5 | 0.966 | 1.0 | 0.924 |
| 6 | 0.970 | 0.997 | 0.924 |

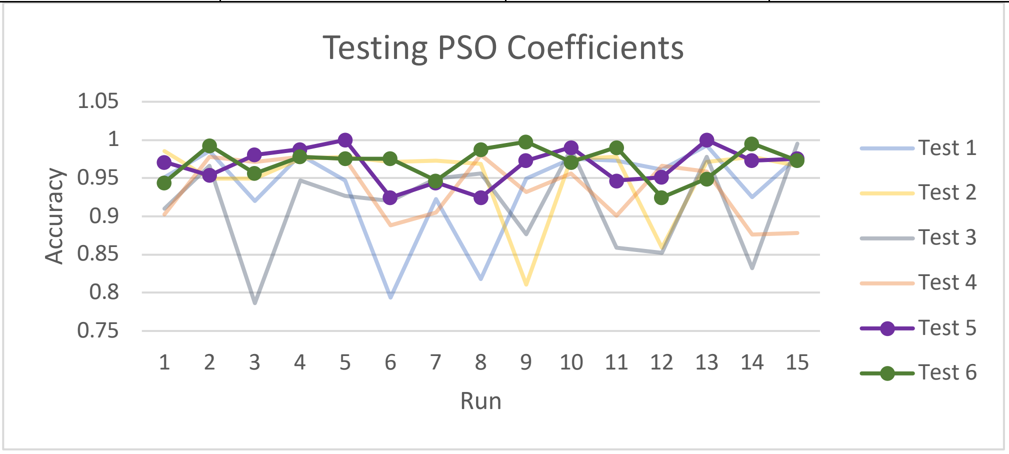

*Figure 1 : The accuracies for different test sets for 15 runs*

Test Set 1 focuses on exploring the global best. Although the maximum accuracy of the test set is 0.99 (in the 13th run), the average accuracy is 0.93. Accuracies can be seen as low as 0.79 in this test set. This proves that exploitation around the global maximum can increase the chance of finding an optimal solution, it also increases the risk of getting stuck at a local minimum since the entire search space will not be covered evenly.

Test sets 2 and 3 focus on personal best exploration and swarm best exploration. Once again, we can see that the algorithm does have the ability to find an optimal solution, however, it may get stuck on a local minimum. Test sets 1, 2, and 3 have the lowest minimum accuracy out of all the other test sets since they focus on searching around known best fitness, i.e. exploitation. Too much exploitation can lead to the PSO algorithm converging to a local minimum instead of a global minimum.

Test set 4 focuses on giving importance to both, the personal best, and the swarm best over the global best. Although the maximum accuracy of all the runs is the lowest compared to the other test sets at 0.98, the minimum accuracy is higher than the first 3 test sets. This is because the algorithm explores more of the search space individually giving it the ability to get out of local minimums.

The test set 5 tries to keep all 3 parameters the same while still adding up to 4. By doing this, the algorithm gives equal importance to all the known bests. An improvement can be seen in average accuracy and even a perfect accuracy of 1.0 on the test set. The minimum accuracy is also the highest at 0.92. This shows that a balance of all 3 parameters helps particles in the PSO algorithm explore the search space but also converge to the global best over time.

Finally, test set 6 tries to keep the same balance but increases the weight of the global parameter so that the particles start searching and converging to the global best a bit faster. As a result, test set 6 sees the highest average accuracy and the least variance out of all the test sets and is chosen as the optimal set of coefficients for this dataset.

## Inertia

***Figure 2*** shows the testing of the inertia parameters. 4 inertia values were tested: 0.2, 0.5, 0.8 and 1.0. The best inertia parameter was 0.5. It had the highest average, maximum, and minimum accuracies. Ideally, particles in a PSO algorithm should start off by exploring and over time slow down and converge to the global minimum. With high inertia values, particles don't slow down quickly enough and can't converge quickly enough to a local or global best, and with the maximum number of iterations in the PSO implementation, the algorithm ends with a sub-optimal solution. On the flip side, keeping the inertia too low means that particles converge too quickly to the personal, informant or local best (based on the coefficients). This could lead to the algorithm getting stuck on a local minimum. With the banknote authentication dataset, the dataset is simple enough for the algorithm to forgive converging faster to the position of best fitness, but having very low fitness values is generally not a good idea with more complex datasets.

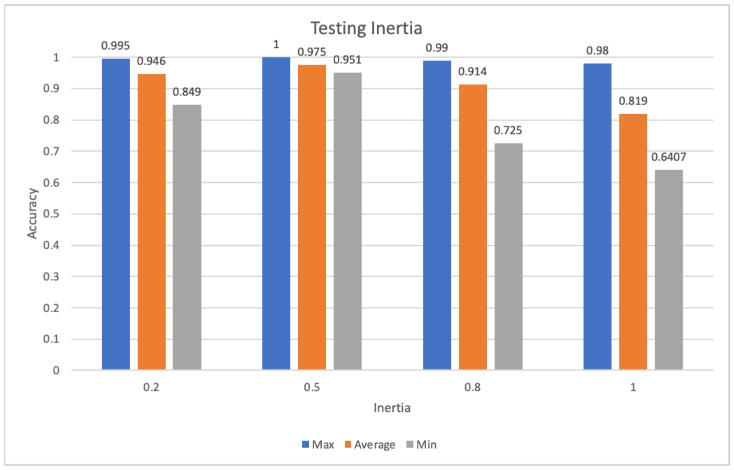

*Figure 2: Testing different values for inertia parameter.*

## Swarm Size and Number of Iterations

The swarm size and number of iterations test tries to find the best balance between the two parameters. It is known that increasing both the particles and a number of iterations together can increase the performance of the PSO but at the same time increase complexity and runtime.

*Table 4: Comparing the runtimes and average accuracies for different test sets.*

| **Test Set** | **Swarm Size** | **Iterations** | **Average Runtime** | **Average Accuracy** |
|---|---|---|---|---|
| 1 | 15 | 75 | 43.78076 | 0.96983356 |
| 2 | 25 | 50 | 49.1243745 | 0.99202497 |
| 3 | 25 | 25 | 25.0329608 | 0.98404993 |
| 4 | 25 | 15 | 15.0429938 | 0.96359223 |
| 5 | 50 | 15 | 28.576604 | 0.95284314 |

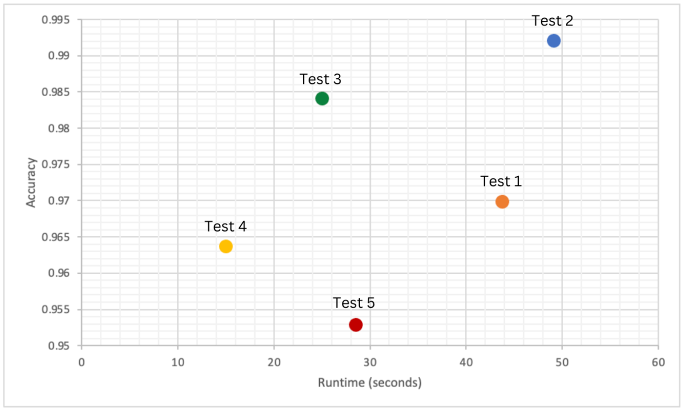

*Figure 3: Graph plotting the runtime accuracies for different test sets.*

In ***Figure 3***, test 3 has the best balance between accuracy and runtime. It has the second highest average accuracy by 0.01 but reduces runtime compared to the highest average accuracy by almost 50%. Striking a good balance between performance and adding complexity can help the PSO algorithm perform a lot more efficiently.

## Number of Informants

The final PSO parameter is the number of informants. This was selected based on the convention mentioned in **<Source>.** However, to analyze the effect of informants on the dataset, 2, 6, and 10 informants were tested.

| Num. Informants | Average Accuracy | Max. Accuracy | Min. Accuracy |
|---:|---:|---:|---:|
| 2 | 0.97597087 | 0.99757282 | 0.93203883 |
| 6 | 0.95970874 | 1 | 0.89563107 |
| 10 | 0.92063107 | 0.98300971 | 0.63106796 |

Although 6 informants give the best maximum accuracy, the average accuracy is lower with 6 informants. This might be due to the banknote authentication dataset having only a few features and being a binary classification problem. For this dataset, a smaller number of informants proves to be a better choice.

## Conclusion

The Particle Swarm Optimization (PSO) effectively improved the performance of the Artificial Neural Network (ANN), achieving consistent accuracy above 90% on the test set with tuned parameters. However, it was noted that PSO operated at a slower pace compared to the backpropagation algorithm. The simplicity of the banknote authentication dataset (CW dataset) limited its suitability for comprehensive algorithm testing, preferring global best exploitation and reducing the impact of linearly decreasing inertia weights. We also noticed that fewer informants, particles, and iterations proved effective for this straightforward dataset. Future exploration in this topic involves testing the algorithm on more complex datasets and experimenting with adaptive parameters that dynamically adjust based on the algorithm's evolving state, incorporating factors such as particle positions and velocities.

## References

[1] Effects of hidden layers on the efficiency of neural networks | IEEE ..., https://ieeexplore.ieee.org/document/9318195 (accessed Nov. 24, 2023).

[2] International Journal of Engineering Trends and Technology ..., http://www.ijettjournal.org/volume-3/issue-6/IJETT-V3I6P206.pdf (accessed Nov. 24, 2023).

[3] J. Garcia-Nieto and E. Alba, “Why Six informants is optimal in PSO,” *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, 2012. doi:10.1145/2330163.2330168