F21BC Stage 1 Report

Anthony Gabini H00356910

Jiancheng Zhang H00341619

Introduction

The purpose of this document is to record our efforts in the creation of a PSO-based ANN for the use of classifying the same dataset as in part 1. As before, our goal was to create a program with the maximal possible performance statistics, and to compare them to the original implementation.

Project Development Rationale

As part of our objectives is a comparison between PSO- and the GD-based ANNs, we had an additional design idea in mind; that is to create a program with fairness in mind when comparing with earlier results. Due to this, the foundational principles of our program were:

- Keeping implementation of the PSO functional, making hyperparameter experimentation easy;
- Maintaining an easy-to-understand code structure for debugging and viewer understanding;
- Additionally, keeping the code close design-wise to that of part 1.

We also wanted to improve on our previous design where we could to make it function efficiently; pseudocode provided by (Luke, 2013) formed the basis of the code we produced.

Methods

To make comparing with our old code possible, we decided to create an additional python file that housed the components of the PSO algorithm, which would be accessible by main.py file. This would ensure easy accessibility to both algorithms from main.py when performing the experiments.

The PSO is housed as a class of four functions, shown below:

```
class MLP_PSO():
                def __init__(self, number_of_layers, number_of_neurons, activation_functions,
loss_function):

    def fit_and_transform(self, X_train, X_test, y_train, y_test, epochs, swarm_size,
number_of_informants, alpha, beta, gamma, delta, epsilon):

    def update_fitness(self, X_data, y_data, position):

    def deserialization(self, position, weights, biases, activation_functions):
```

The **constructor** accepts the basic parameters of the neural network; we note that the activation function this time is represented by a number rather than a string. This allows for a function to be implemented which optimises the activation function used.

fit_and_transform() accepts parameters that are PSO specific (aside from epochs). They are:

- Alpha: Proportion of velocity kept for the update.

- Beta: Proportion of best velocity recorded by each particle kept for the update.
- Gamma: Proportion of best velocity recorded by each particle's informants kept for the update.
- Delta: Proportion of best velocity recorded globally kept for the update.
- Epsilon: Proportion of new velocity used for the update.

The function first initialises all velocities and positions in the matrix (rows corresponding to particles, and columns to the weights, biases, and activation functions; the informant of each particle is chosen randomly. The function then updates the particle, informant, and global best respectively. Afterwards, it updates each particle's velocity first and then updates via the three kinds of best records. Finally, global best fitness is returned with the test data used for accuracy.

update_fitness() deserializes the swarm into weights, biases, and activation functions, and then uses forward propagation to calculate the output array. Since accuracy is used rather than cross entropy, we simply use a sum of all correct predictions to define the accuracy; CE is not used simply due to PSOs not relying on gradients to update weights and biases, and instead judge the model fitness via accuracy on the dataset.

deserialization() accepts the position of each swarm and turns them into weights and biases matrices, and in turn gets each layer's activation function.

To generate results efficiently, we used an automated result generation and recording method in the main.py file. As we have more hyperparameters to review in this part, we chose to perform a grid search over 4 hyperparameters twice (to cover all eight) and chose to review 5 values over each hyperparameter range. This is due to preliminary experiments showing an accuracy generation time of 10 seconds; operation-time calculations showed us this would mean an estimated time of 5^4*10 seconds = 1.7 hours for each experiment (5 steps for each of the 4 parameters).

The output would allow us to perform a grid search, from which a second round of experiments could be made with the most interesting hyperparameters. These hyperparameters were the swarm size, alpha, and epsilon. Here we increased the granularity of the experiments, with 7 steps taken. Each setting would be tested three times to determine an average; this created an estimated experiment operation-time of 7^3*3*10 seconds, or 2.9 hours.

A benchmark PSO was established to be 30 epochs, swarm size of 100, informant size 20, alpha 0.5, beta 1.5, gamma 1.9, delta 1.8, and epsilon 0.5.

Results

The first round of experiments showed the following:

- The ratio between Beta-Gamma-Delta (BGD) is more important than their individual values; best accuracy values resulted when each of their values were kept within a certain proportion of each other.
- Larger Alpha values generated better accuracy and lesser time required for the experiment.
- Epsilon indirectly impacts accuracy and time; however, this is again dependent on the BGD ratio.

- Swarm size increase led to an increase in accuracy, with a linear increase in duration.
- Epochs linearly increase accuracy and duration.
- Increase of informants leads to a slight increase in accuracy but has no bearing on duration.

Accuracy										
	Beta	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
	Gamma	0.7	0.7	0.7	0.7	0.7	1.1	1.1	1.1	1.1
	Delta	0.2	0.6	1	1.4	1.8	0.2	0.6	1	1.4
Alpha	0.5	33.59375	28.125	30.46875	32.8125	37.69531	28.90625	34.17969	37.10938	39.84375
	0.7	30.66406	32.8125	42.1875	41.99219	42.96875	39.84375	39.84375	43.16406	52.73438
	0.9	43.16406	61.71875	90.23438	64.64844	86.32813	83.20313	76.95313	77.73438	86.91406
	1.1	92.1875	81.25	97.26563	97.46094	99.02344	92.77344	83.59375	95.50781	99.21875
	1.3	94.14063	97.85156	92.1875	73.63281	66.99219	95.89844	90.23438	96.875	96.875
Time										
		11.34219	11.20756	11.27321	11.39938	11.29332	11.22525	11.3492	11.24221	11.32288
		10.232	10.63931	10.24705	10.27903	10.25169	10.20561	10.23128	10.25782	10.17334
		10.27521	10.18734	10.26533	10.20782	10.21125	10.18605	10.22933	10.24763	10.23667
		9.949005	9.870224	10.06994	9.892838	10.14346	9.840178	10.15554	9.924147	9.87476
		9.780621	9.821445	9.755668	9.781844	9.762251	9.956223	9.821842	9.921834	9.93319

	Accuracy									
	Swarm Si	40	40	40	40	40	40	40	40	40
	Number c	5	5	5	5	5	10	10	10	10
	Epsilon	0.05	0.2	0.35	0.5	0.65	0.05	0.2	0.35	0.5
Epochs	10	37.69531	43.16406	43.16406	43.16406	96.875	34.57031	54.6875	43.16406	95.11719
	30	54.10156	61.71875	99.41406	97.46094	95.11719	43.16406	91.99219	96.67969	98.63281
	50	92.77344	97.65625	99.21875	99.41406	99.21875	83.98438	98.4375	93.94531	100
	70	96.28906	97.65625	94.53125	95.89844	97.46094	93.94531	98.04688	97.85156	98.24219
	90	43.16406	90.625	97.46094	99.21875	95.50781	93.16406	73.4375	100	95.70313
	Time									
		1.423334	1.407502	1.362633	1.372032	1.378814	1.442934	1.412135	1.356134	1.371068
		4.170135	4.10947	3.948937	4.04535	3.993736	4.163208	3.989038	4.02857	4.013366
		6.820801	6.501542	6.57569	6.49627	6.783472	6.851607	6.925213	6.791519	6.744923
		9.669671	9.069221	9.465556	9.230917	9.416411	9.645261	9.462678	9.176285	9.056139
		12.26901	11.58717	12.06891	11.75122	11.87875	12.01984	12.222	11.61582	11.56582

Figure 1: Excerpt of results obtained from the first round of experimentation, with accuracy shown on top and corresponding duration below

The range of results in this first stage of experiments was accuracy-wise from 28.18%-100%, and time-wise 1.36s-60s, with the range extremes mostly found in the experiments concerning swarm-size and number of epochs.

From here, we determined the hyperparameters we wanted to examine in greater detail using the best values found from the first round. Interestingly, alpha did not have an impact on the accuracy or time, and the same can be said for the value for epsilon. The swarm size had the largest impact, each step increasing the result for accuracy and duration linearly.

Accuracy																		
	Epsilon	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.35	0.4	0.45
	Swarm Si	80	80	80	80	80	80	80	100	100	100	100	100	100	100	120	120	120
Alpha	0.8	91.862	96.549	97.721	98.633	99.609	96.549	98.698	98.633	93.815	96.224	99.284	98.242	99.414	98.828	98.047	98.633	95.703
	1	99.544	97.982	98.307	98.112	98.893	91.081	98.438	99.674	99.544	99.414	96.159	98.763	98.633	99.089	95.247	98.763	98.633
	1.1	97.201	97.526	97.591	98.568	94.336	97.005	99.154	92.578	94.792	98.893	98.568	96.289	98.177	99.023	94.922	98.438	96.484
	1.2	96.745	97.917	96.875	91.667	98.763	99.609	96.159	99.023	98.763	95.313	99.219	99.544	98.763	98.828	97.917	97.786	98.438
	1.3	96.81	99.479	99.414	99.479	98.372	98.112	98.633	98.828	99.349	98.958	98.698	87.565	99.023	95.117	98.242	97.852	98.177
	1.4	97.917	99.674	98.633	97.526	98.828	96.549	95.638	96.224	98.958	98.568	97.721	99.023	98.893	98.047	97.786	98.372	98.633
	1.5	98.177	95.378	98.568	98.372	97.526	98.307	98.372	99.023	96.354	99.154	98.242	97.656	98.177	98.698	96.81	98.177	99.284
Time																		
	Epsilon	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.35	0.4	0.45
	Swarm Si	80	80	80	80	80	80	80	100	100	100	100	100	100	100	120	120	120
Alpha	0.8	8.092	8.0864	8.0508	8.1802	8.2926	8.2729	8.2591	10.24	10.132	10.876	10.902	10.338	10.719	11.132	13.511	12.31	11.997
	1	8.0865	8.204	8.0154	8.0469	8.0511	7.9691	8.1946	9.9729	9.9616	10.076	9.8838	10.013	10.131	10.111	12.039	11.924	12.068
	1.1	7.9058	8.0218	7.9089	8.0834	8.0072	8.228	8.0551	9.9207	9.856	10.017	10.062	10.485	10.218	10.141	11.796	11.632	11.888
	1.2	8.0572	7.9973	8.1046	8.2297	8.3704	8.1522	8.4998	10.082	9.9743	10.09	10.282	10.478	10.065	10.373	11.96	12.092	12.049
	1.3	8.0274	8.1335	8.2364	8.5112	8.3743	8.436	8.4913	10.045	10.04	10.127	10.442	10.007	10.369	10.731	12.366	12.526	12.119
	1.4	8.1966	8.1944	8.2245	8.274	8.1939	8.2836	8.5745	10.003	10.277	10.5	10.306	10.116	10.817	10.449	12.202	12.265	12.625
	1.5	8.4597	8.3456	8.2371	8.2763	8.3229	8.1969	8.0555	10.219	10.501	10.353	10.345	10.048	10.741	10.768	11.844	12.287	12.494

Figure 2: Excerpt of results for the second round of experiments. Note the linear, step-like increase in duration with respect to swarm-size.

This round of experiments yielded a tighter range of results, with an accuracy range of 87.57%-99.94%, and duration range of 7.91s-20.96s. This suggests the benchmarks set for this round of experiments let to a better efficiency on average.

We were able to achieve a relatively very high accuracy with low hyperparameter settings, with an accuracy of 99.5% and duration 8.09s, with a swarm size of 80, alpha 1, and epsilon 0.35, and rest of hyperparameters set as the benchmark.

Discussion and Conclusions

Fine-tuning settings to high degree of specificity for hyperparameters is nothing new when having to experiment with MLPs, however this is something that became a hurdle far more significantly than the earlier MLP worked on in stage 1. Having to implement a PSO on the dataset provided required a greater level of preparation and made us consider factors that weren't even considered in the earlier stage. Thus, the main subject of consideration from this stage is the level of appropriateness of using one kind of MLP implementation over another one.

How we define appropriateness is not explicit and relies on a multitude of factors. PSO-based MLPs work best on a local scale; communication between particles is based on the concept of locality, and as a result there is no idea about the global situation (Rath et al., 2021). This would suggest PSO being a better solution for the provided dataset than the previous GD-based ANN, given its relative very small size.

The dimensionality of the search space impacts the ability of an ANN algorithm to give good results effectively. In our case, the search space is high-dimensional, meaning it is unlikely to be a convex function. As a result, Gradient Descent may not be the best solution, as there is an increased chance of the algorithm stopping due to small gradients. This is unlike the PSO, which has the capability of searching despite small gradients (as it effectively trains multiple neural networks simultaneously).

The convergence of the algorithm to a point is guaranteed in the case of the PSO (Rini, Shamsuddin and Yuhaniz, 2011), however does not necessarily mean that a PSO provides a guaranteed global best solution; indeed, a limitation of the algorithm is that it can get trapped in local extreme points, with the possibility of a better solution being unrecognised (Aote, Raghuwanshi and Malik, 2013).

There is yet room to explore more of PSO characteristics. There exist plenty of variations of the PSO, from different biological inspirations to variations on the mathematical principles on which the basic form is inspired from. Experimenting with these variations could lead to better results, for example, "stretching" has been used to alleviate the effects of local minima preventing from obtaining the global minimum efficiently (Parsopoulos, 2010).

References

Luke, S., 2013. *Essentials of Metaheuristics*. 2nd ed. Fairfax: Lulu, p.57.

Rath, M., Darwish, A., Pati, B., Pattanayak, B. and Panigrahi, C., 2021. *Swarm Intelligence forResource Management in Internet of Things*. London: Elsevier, p.23.

Rini, D., Shamsuddin, S. and Yuhaniz, S., 2011. Particle Swarm Optimization: Technique, System and Challenges. *International Journal of Applied Information Systems*, 1(1), p.21.

Aote, S., Raghuwanshi, D. and Malik, D., 2013. A Brief Review on Particle Swarm Optimization: Limitations & Future Directions. */ International Journal of Computer Science Engineering (IJCSE)*, 2(5), p.199.

Parsopoulos, K., 2010. *Particle swarm optimization and intelligence*. Hershey, PA: Information Science Reference, p.153.