Machine Learning Project 4: Population Based Algorithms

Nicholas Stone nicholas.stone2@student.montana.edu

Matteo Björnsson matteo.bjornsson@student.montana.edu

Abstract

In this paper we present an experiment analyzing the performance of three population-based algorithms on six data sets. 10 fold cross validation is used to evaluate each algorithm. We propose that, in general, the three population-based algorithms discussed in this paper will perform better on classification data sets than backpropagation. Additionally, we propose that, of the three algorithms, Particle Swarm Optimization will perform the best, followed by Differential Evolution and finally Genetic Algorithm. We did not find enough evidence to clearly support our hypotheses and discuss the possible factors affecting performance. No real conclusions can be drawn here without additional experiments.

Keywords: Genetic Algorithm, Differential Evolution, Particle Swarm Optimization, Neural Network, Backpropagation

1. Introduction

The population-based algorithms, Genetic Algorithm, Differential Evolution, and Particle Swarm Optimization, are supervised learning models that iteratively modify a population of possible candidate solutions to minimize or maximize a given fitness function. This paper applies these algorithms to learn the weights of 0, 1, and 2 hidden layer neural networks. Previously, we used backpropagation to learn these weights and will compare the performance of this alternative way of training the weights.

Each of these algorithms uses its own unique probabilistic tactic for generating a new series of weights. In this paper we implement each of these algorithms to learn the weights of each of the neural network architectures. We vary the number of hidden layers in the networks from 0-2 and compare results from three classification data sets and three regression data sets using ten fold cross validation. We hypothesize scenarios where these algorithms would perform well and where they would not perform well. Relevant background information is reviewed in Section 2, experimental hypotheses are presented in Section 3, and experimental design in section 4. We conclude with our results and discussion in Section 5.

2. Background Information

All of the population-based algorithms presented in this paper use a number of individuals that represent solutions to the problem the algorithms are trying to learn. The general idea is that the individuals in a population are modified iteratively to produce better solutions.

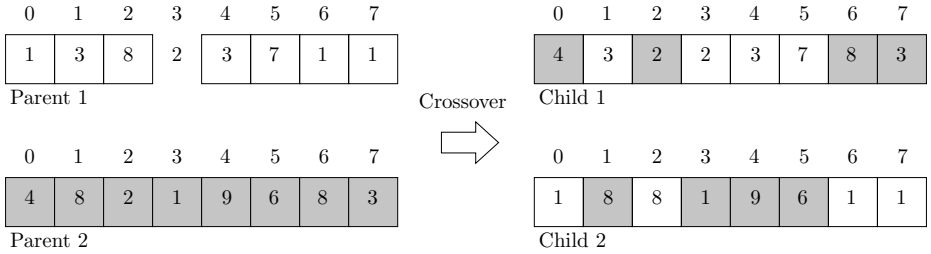

Figure 1: *An example of Uniform Crossover. Each position in the parent chromosomes has a chance of being exchanged in the children according to the crossover rate.*

"Better," in this case, is defined by a function called the fitness function that evaluates the quality of the solution. The fitness function is the key to tying these algorithms to a specific problem. For this experiment, we are implementing these algorithms to learn the appropriate weights of a shallow feedforward neural network for different classification or regression tasks. The fitness of an individual therefore measures the error in classification or regression of the given data set produced by the network using the weights supplied by the individual. We use the network architecture from our previous project and compare the performance of the population based methods of training the neural network with backpropagation.

It is important that any population based algorithm be able to explore the search space effectively and eventually converge on a the best possible solution without converging too soon on local optima. The way in which the individuals explore the space of possible solutions, or in other words are modified iteratively, as well as which individuals are modified, differs from algorithm to algorithm, which we detail here. All individuals are represented by an n dimensional vector, where n is the total number of weights required for the given network architecture.

2.1 Genetic Algorithm

Individuals in the Genetic Algorithm (GA) population are thought of as chromosomes. Each generation, individuals are chosen to breed a new generation by mixing their genetic material to create offspring. Individuals with a better fitness are selected with a higher probability to be parents of the next generation. Individuals can be chosen with a probability proportional to the strength of their fitness compared to the average fitness of the population, the rank of their fitness compared to the rest of the population, or via a tournament style selection. Our implementation of GA uses rank based selection. The probability of selection for a given individual in ranked selection is described in Equation (1).

$$p(\mathbf{x}_i) = \frac{2}{|P|}\left(\frac{|P| - \text{rank}(\mathbf{x}_i)}{|P| - 1}\right) \tag{1}$$

Here rank($\mathbf{x}$) is the rank of the fitness of chromosome i and $|P|$ is the size of the population. Note that individuals are selected with replacement, allowing individuals to be selected more than once.

Once selected, individuals are bred to create the new child chromosomes via crossover, an example of which is illustrated in Figure 1. In addition to crossover, every child chromosome is randomly mutated such that any individual part of the chromosome has a chance of being changed to a completely new value according to the mutation rate of the algorithm. Once the new generation is created the old generation is either entirely replaced or only the least fit individuals are replaced. This process continues until some desired measure of fitness is achieved or the algorithm is no longer finding better solutions.

2.2 Differential Evolution

Differential Evolution (DE) is very similar to GA in that it follows the same pattern of crossover, mutation, and replacement. Note that it differs from GA in that it does not select a subset of the population, but rather selects every member of the population each generation. When an individual is selected, instead of being combined with just one other individual, it is combined with a "trial" vector, $\mathbf{v}$, constructed from three randomly chosen other individuals, $\mathbf{x}^1$, $\mathbf{x}^2$, and $\mathbf{x}^3$ from the population (without replacement). The trial vector $\mathbf{v}$ is generated according to Equation (2).

$$\mathbf{v} = \mathbf{x}^1 + \beta\left(\mathbf{x}^2 - \mathbf{x}^3\right) \tag{2}$$

Here β is a tunable parameter that dictates the size of the differential component of the trial vector. Once the trial vector is created, the original selected vector is combined with the trial vector via crossover to produce a single child vector. One method that can be used for crossover is Binomial Crossover, where each value in the child vector is selected from the original vector according to some chance defined by a crossover rate parameter, and selected from the trial vector otherwise. This process is very similar to Uniform Crossover depicted in Figure 1. The fitness of the child vector is then evaluated, and the vector with the best fitness between the parent and child is placed back into the population.

2.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) treats the individual solutions as particles in the search space that have position and velocity. The position of any given particle is the solution it represents. The major difference between PSO and the previous two algorithms is each individual retains knowledge of its own personal best solution as well as some neighborhood best solution, depending on the chosen topology of the neighborhood. This topology can be represented by a graph of connections between particles. A "local best" topology connects all particles in a ring, where each particle knows the best solution between itself and its two neighbors. A "global best" topology connects all particles such that each particle is aware of the best solution found by any particle so far. Each time step, the position, velocity, and fitness of every particle is updated, as well as the personal best and local best for each particle. The velocity and position of each particle is updated each time step t according to Equations (3) and (4).

$$\mathbf{v}_i^{t+1} = \overbrace{\omega\mathbf{v}_i^t}^{\text{inertia}} + \overbrace{c_1 r_1\left(b_i - \mathbf{x}_i^t\right)}^{\text{cognitive}} + \overbrace{c_2 r_2\left(b_g - \mathbf{x}_i^t\right)}^{\text{social}} \tag{3}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{4}$$

The components of the velocity equation, inertia, cognitive, and social, are weighted with various tunable and random parameters. The inertia component is scaled via ω. Parameters c_1 and c_2 are tunable, whereas r_1 and r_2 are drawn from a uniform normal distribution between 0 and 1. b_i represents the personal best of particle i, therefore representing the "cognitive component" of the velocity, and b_g represents the neighborhood best, therefore representing the "social component." In this case the subscript g indicates the neighborhood topology is that of a global best. The idea is that, over time, the swarm continues to find better solutions and particles change their search trajectory somewhat randomly towards those personal bests and global bests. Due to the fact that the velocity can behave in sometimes unpredictable ways, the max value of any element of the velocity vector is clamped to some value v_{max}.

2.4 Data Sets

This experiment tests three classification problems and three regression problems. A quick summary of the data sets is presented in Table 1. These are the same data sets that were used to evaluate the Backpropagation Neural Network training algorithm from last paper. It is worth noting that the Fire and Glass data sets have proven to be difficult problems to learn.

Data set	Character	#Sample	#Attr.	Estimate	Notes
Fire	Mixed	517	12	Regress.	Difficult task, output trends toward 0.
Abalone	Mixed	4178	8	Regress.	Relevant features.
Machine	Mixed	209	7	Regress.	Some irrelevant features?
Cancer	Categorical	699	9	Class.	Class split 65/35. Some missing attr.
Soybean	Categorical	47	35	Class.	One class has twice the occurrence.
Glass	Real	214	9	Class.	Very uneven distribution of classes, difficult task.

Table 1: *Data set summary.*

3. Hypotheses

We expect that one factor that will affect the performance of these three algorithms is the number of attributes in the data set. This is because we expect these algorithms to perform better in a lower dimensional space and the number of attributes increases the number of weights the algorithm needs to learn. This corresponds to a higher dimensional space the algorithms need to search through. Therefore, we anticipate that Machine data set will perform well for this reason, and the Soybean data set will perform poorly because of the comparatively large number of attributes.

We believe that the three algorithms will perform worst on Glass and Forest Fires because they are clearly difficult problems. However, we think that the heuristic algorithms may do better with these two problems than backpropagation because a randomized approach may encounter better solutions to these more complex problems by chance.

We hypothesize that backpropagation will perform better in regression problems in general while the meta-heuristic algorithms will perform better than backpropagation in classification problems. We suspect this because the backpropagation algorithm uses the known gradient solution to the error surface to iteratively traverse this surface, which corresponds more readily to regression which is continuously valued.

When comparing the performance of the Genetic Algorithm, Differential Evolution algorithm and Particle Swarm Optimization algorithm, we believe that PSO will perform the best, followed by DE then GA. This hypothesis is due the fact that PSO always has knowledge of the global best, and updates its positioning with this given knowledge. This means that the algorithm is always working towards the global best. Comparatively, differential evolution makes the same progress probablistically without knowing what the best solution is. Finally, the genetic algorithm will likely move towards this global best, but it is not guaranteed since the crossover process pulls only from the weight values of its parents. Meaning that, in some cases, a new child is generated on the basis of a less fit parent.

In summary, we hypothesize that the 3 population-based algorithms will perform better on classification problems as compared to the backpropagation algorithm. Additionally we predict that the algorithms will perform better on data sets with a smaller feature space, meaning that they will perform better on the Machine data set and will perform worse on the Soybean data set. We also expect that the Cancer and Machine data sets will perform well because they have each performed well with other learning algorithms.

4. Experimental Design

4.1 Preprocessing

Before the six data sets were used to test the algorithms, some preprocessing was required. The first step was to remove features from the data sets that do not have any influence on the result. This is seen only in the Machine, Cancer and Glass data sets with the removal of a sample index and the name of the model and manufacturer of each machine product. This of course is because an index does not have any clear correlation to the estimation of a given data point. Before normalizing all of the data, additional processing was done to data sets that had categorical features. These features were mapped to a integer value if they were not already represented by one. This is because all three meta-heuristic algorithms use candidate solutions represented by a vector of real values and the network weights to be learned must be real valued as well. For example, with the Forest Fire data set, the days of the week and month names were converted to an equivalent integer value from 0-6 and 0-11, respectively.

Every data point that had missing attributes was randomly assigned a new value between the minimum and maximum values from the feature space.

It should be noted that the same preprocessing steps were taken with this data as when we trained the neural networks using backpropagation. This way, we can directly compare the meta-heuristic performance with backpropagation. Since the data is fed through a network, the values of each feature were normalized to a value between zero and one. This was done to all data sets. This is an important step to prevent one feature from dramatically "outweighing" another feature in the network. Finally, all of the regression target variables

are also normalized between zero and one so a sigmoid activation function can be used to estimate regression output values.

4.2 Hyperparameter Tuning and Experimental Setup

Hyperparameter Tuning Each of the three algorithms had several hyperparameters that needed to be tuned or set. Parameters were tuned by selecting the set of parameters that performed the best over three trials using a random 90% of the data set to test the remaining 10%. Hyperparameters were tuned for each data set and for each network architecture.

For GA, the mutation rate and crossover rate were tuned. Values of 0.2, 0.5, and 0.8 were tested for mutation rate, which determines the probability that any given value within the chromosome would be reassigned a random value between -10 and 10. The values tested for crossover rate were 0.2 and 0.5, corresponding to a probability of a somewhat low mixture of both parents vs an even split of genetic material from both parents.

For DE, we tested values in ranges suggested by Rainer Storn (Storn, 1996) for β and the crossover rate. For β, values 0.2, 0.5, and 0.8 were tested. For the crossover rate, values of 0.1, 0.3, and 0.8 were used, corresponding to a small, somewhat small, and large proportion of the child chromosome which is likely composed of the trial vector.

For PSO, ω, c_1, and c_2 were tuned. Values of 0.1 and 0.4 were tested for ω. Values of 0.1, 0.5, 0.9, and 3.0 were tested for both c_1 and c_2. We used value of 1 for v_{max}.

When presented with a need for a random value for a weight, such as initializing each individual in a population or for mutation, values were drawn randomly from -10 to 10. This is based off of the fact that most weights fell within this range for the previous backpropagation implementation. A population size of 500 was used for each algorithm. This parameter dramatically affected the computation time of each algorithm, and was therefore limited to 500. It would be beneficial to tune this parameter as well in future experiments.

Experimental Setup To keep the results of this experiment consistent with those of our previous paper, we will be using the same experimental design to evaluate the performance of these three algorithms as we used to evaluate the performance of backpropagation to train a neural network.

For the Genetic Algorithm, selection is determined using a weighted roulette rule, weighted by the fitness rank of the individual, described by Equation (1). A parent population of size $|P|/2$ is selected with replacement; parents are bred in order of selection to produce two children. The entire population is then replaced with the new generation. Uniform crossover is used when generating child chromosomes.

For Differential Evolution, population members are selected in order. Two vectors are used to generate the differential (for a total of three vectors randomly chosen to generate the trial vector each time). Binomial crossover is used between the selected vector and the trial vector to produce the new child vector.

For Particle Swarm Optimization, the max velocity is clamped to v_{max}. Additionally, a global best topology is used to determine neighborhood.

Each of the algorithms, GA, DE, and PSO are used to train a 0, 1, and 2 hidden layer feedforward neural network on each data set using the appropriately tuned parameters. For any given algorithm and network architecture, the model will be trained and tested using

ten fold cross validation where the data set is divided into ten parts, randomly if regression and with stratification of the classes if classification. Each one of the ten folds will be classified or estimated on a model trained with the remaining nine folds. The performance of the estimation will then be evaluated using several loss functions described below.

4.3 Evaluation

Different loss functions are used to evaluate classification and regression data sets. The performance of the algorithm on categorical data sets is evaluated using 0/1 Loss and F1 score. Regression is evaluated using Mean Squared Error and Mean Absolute Error.

0/1 Loss is a measure of classification accuracy and only measures the percent of examples classified correctly. 0/1 score does not weight the consequence of misclassification, but such a measure is not required for this experiment. Furthermore, for each class the number of True Positive, True Negative, False Positive, and False Negative assignments are counted. From these counts the precision and recall are calculated on a per-class basis. The F1 score of each class is the harmonic mean of these two values. The total F1 score for the model is a weighted average of all of the per-class F1 scores.

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The regression data sets were evaluated using Mean Absolute Error (MAE) and Mean Squared Error (MSE). MAE takes the absolute value of the difference between the real regression value and the estimated value. MSE takes the difference between the two values and squares the result.

5. Results and Discussion

Experimental results are summarised in Figures 2 and 3 for comparison. The data charted in these figures is also presented in Tables 3 and 2 for specific values.

The original hypotheses proposed the backpropagation algorithm would perform better on regression data sets than the meta-heuristic algorithms. There is no clear evidence of this one way or another from our results. Backpropagation sometimes performed better than some of the meta-heuristic on some network architectures and data sets, and sometimes worse, for both classification and regression. For classification, the backpropagation algorithm performed comparably to the population-based algorithms for the most part, though slightly worse on Soybean, and with notable discrepancies occurring on the Cancer data set, performing much worse with two hidden layers and better with 0 layers. For regression, backpropagation performed comparably on Fire, but had distinct discrepancies on Machine (doing much better on 0 layers but much worse on 1 and 2 layers) and underperformed on just 0 layers on Abalone. While we cannot draw distinct conclusions from these outcomes, it does seem to support the fact each data set has complexities that are difficult to predict and work well with some algorithms and not with others.

		Abalone		Fire		Machine	
	Hidden Layers	MSE	MAE	MSE	MAE	MSE	MAE
Back Propagation	0	0.05722	0.00612	0.16673	0.04200	0.00522	0.00023
	1	0.05406	0.00552	0.16177	0.03936	0.02819	0.00264
	2	0.05870	0.00649	0.16163	0.03930	0.03026	0.00268
Differential Evolution	0	0.07327	0.01008	0.16534	0.04404	0.07090	0.02258
	1	0.05658	0.00615	0.16319	0.04076	0.01016	0.00195
	2	0.06184	0.00719	0.16640	0.04202	0.01819	0.00278
Genetic Algorithm	0	0.07586	0.01073	0.16718	0.04108	0.07694	0.02162
	1	0.05738	0.00629	0.16541	0.04121	0.01697	0.00277
	2	0.05590	0.00605	0.16657	0.04236	0.01730	0.00203
Particle Swarm Optimization	0	0.07367	0.01020	0.16600	0.04263	0.06969	0.02092
	1	0.05918	0.00663	0.16493	0.04004	0.01132	0.00060
	2	0.06248	0.00769	0.16581	0.04078	0.01363	0.00176

Table 2: *Experimental results for the regression data sets for zero, one, and two hidden layers for each algorithm. These loss scores (MSE and MAE) are a measure of the error of the regression estimate with respect to the known value.*

		Cancer		Glass		Soybean	
Algorithm	Hidden Layers	F1	0/1	F1	0/1	F1	0/1
Back Propagation	0	95.430	95.420	57.300	58.870	93.170	95.000
	1	96.890	96.850	60.270	62.990	94.400	96.000
	2	53.410	66.670	55.050	57.400	87.970	89.000
Differential Evolution	0	87.317	87.118	55.525	57.922	96.200	95.500
	1	92.655	92.565	54.100	59.870	100.000	100.000
	2	95.602	95.567	37.911	49.177	92.300	90.500
Genetic Algorithm	0	87.452	87.273	52.665	56.147	100.000	100.000
	1	95.553	95.561	61.643	64.545	95.667	95.500
	2	96.138	96.135	56.059	60.368	94.833	95.500
Particle Swarm Optimization	0	87.602	87.406	58.704	60.801	100.000	100.000
	1	96.024	95.992	60.821	64.026	92.533	94.000
	2	96.593	96.563	58.445	62.186	87.533	86.000

Table 3: *Experimental results for the classification data sets for zero, one, and two hidden layers for each algorithm. The performance values (F1 and 0/1 score) are a measure out of 100.*

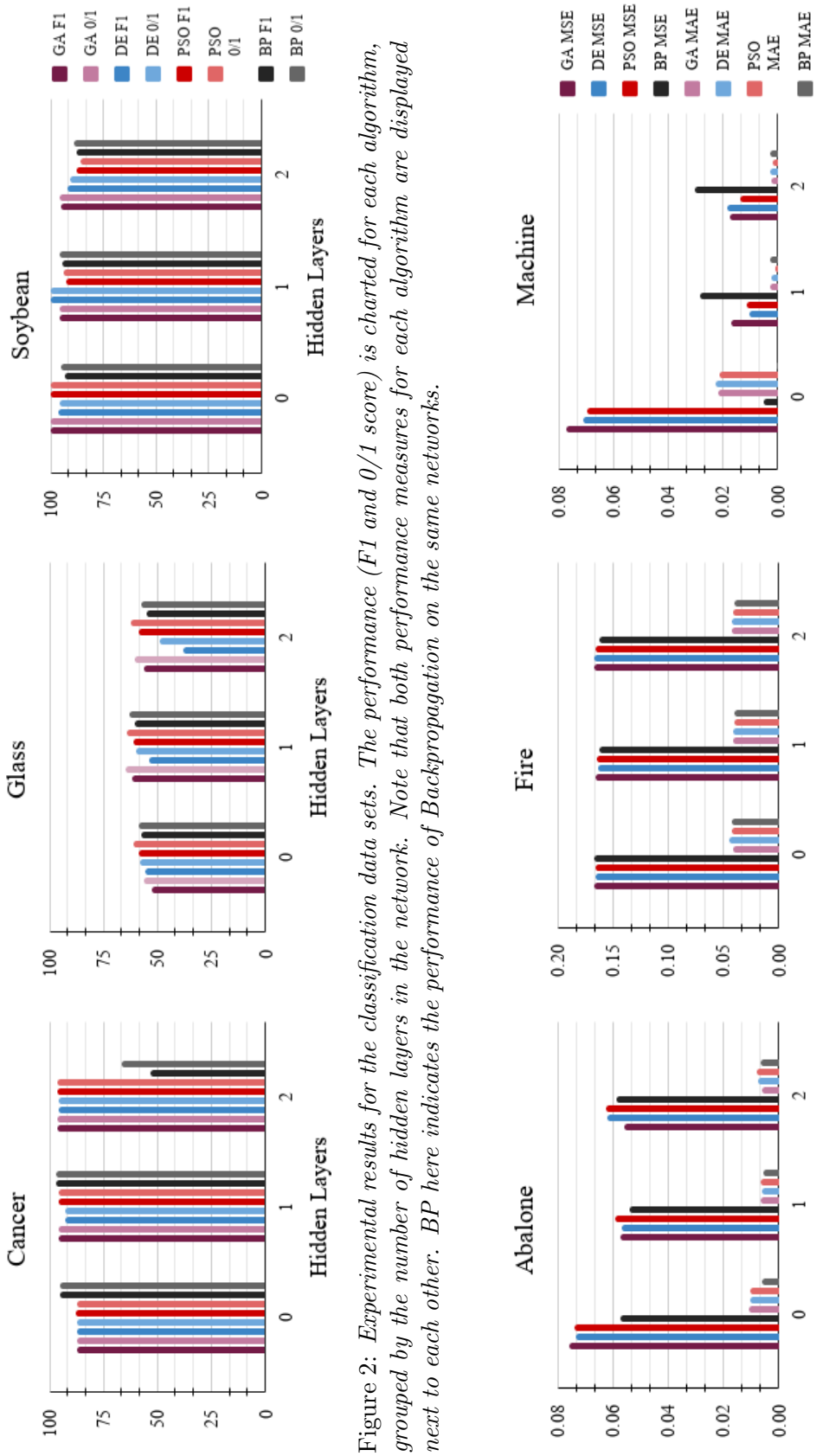

Figure 2: *Experimental results for the classification data sets. The performance (F1 and 0/1 score) is charted for each algorithm, grouped by the number of hidden layers in the network. Note that both performance measures for each algorithm are displayed next to each other. BP here indicates the performance of Backpropagation on the same networks.*

Figure 3: *Experimental results for the regression data sets. The performance (MSE and MAE) is charted for each algorithm. Note here that the MSE values are grouped together and the MSE values are grouped together. The performance values in total are grouped by the number of hidden layers. BP here indicates the performance of Backpropagation on the same networks*

We initially hypothesized that the the meta-heuristic algorithms would perform worst on the soybean data set due to its large feature space and that the Cancer and Machine data sets would perform the best. After looking at the results produced we recognize that, in general, the soybean data set actually performed very well, if not better than that of the cancer data set. It's possible that with such a small data set the meta heuristic algorithms were able to probabilistically determine a solution quite well despite the larger feature space. It seems that the number of features does not dramatically affect the performance in this specific task and at this scale (as opposed to thousands or millions of features). In the meantime, the Cancer data set still performed fairly well for all of the population-based algorithms. Interestingly enough, the meta-heuristic algorithms did not clearly do well or poorly on Machine, as its performance varied dramatically with respect to the number of hidden layers.

Finally, all of the evolutionary algorithms seemed to perform very similarly to one another, with the results not clearly favoring any of the given algorithms in any of the data sets. Our hypothesis did assume that PSO would do the best since it is working towards a known global best. Based on the experimental results, we can say that this hypothesis did not hold. While PSO did not clearly outperform the other algorithms, it is interesting to note that it did perform comparably despite the fact that it is known that a local best (ring) topology performs better than a global best topology. It is possible that PSO may have performed the best had we used a local best topology. We believe that the selection of performant chromosomes in GA, or only choosing the most fit individuals in DE, is in its own way maintaining a sort of "global best," either probabilistically in GE or by only selecting better individuals in DE.

6. Conclusion

This paper presented a study of three population-based algorithms as applied to learning the weights of neural networks with differing numbers of hidden layers. The performance of these algorithms was also compared to backpropagation, a common method for training feedforward neural networks. All three algorithms and back propagation were tested on six unique data sets with varying feature spaces, and the performance of each algorithm was evaluated using 10-fold cross validation. In reviewing the experimental results, we cannot reach any definitive conclusions regarding our original hypotheses. While the cancer data set did perform well, we had predicted that the soybean data set would have done the worse and for each of the evolutionary algorithms. This was not the case. However, one can conclude that backpropagation is very comparable to population-based algorithms for training shallow feedforward neural networks, depending on the data set. More extensive hyper parameter tuning and further experimental trials should be conducted in order better understand the relationship between these algorithms.

References

R. Storn. On the usage of differential evolution for function optimization. In *Proceedings of North American Fuzzy Information Processing*, pages 519–523, 1996. doi: 10.1109/NAFIPS.1996.534789.