Multi-Layer Perceptron

Logan Ladd logan.ladd@ecat1.montana.edu
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-3880, USA

Asher Worley asher.worley@ecat1.montana.edu
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-3880, USA

Editor: Logan Ladd

Abstract

The MLP neural network is a supervised training method used for classification or regression. Population based algorithms may be used to train these networks. Each mother requires tuning and no algorithm outperforms the other. Selection of training depends on the nature of the model. Population based algorithms take more time to train than backpropagation.

1. Introduction

The problem at hand is to compare the effectiveness of a feed forward MLP trained by pop based algorithms and backpropagation. The three algorithms are genetic, differential evolution, and particle swarm optimization. Each algorithm will be tested on 6 data sets. Backpropagation was implemented in project 3, so results will be reviewed and compared but there will not be any re-implementation or execution.

Population based algorithms are expected to outperform a backpropagation algorithm because they are less prone to being caught in local minima. It is likely that the genetic algorithm will perform better than the differential evolution algorithm and the particle swarm optimization algorithm because of its unique mutation operator. Mutation through something called a creep term introduces an entirely random component to the direction an offspring moves from its parent, which will introduce more diversity later in a run that the other two algorithms. While the mutation in genetic algorithms has the possibility of introducing diversity, it also makes the genetic algorithm similar to a random walk. We believe this will give it the slowest convergence rate. We believe the differential evolution algorithm will converge the quickest of three population based algorithms. This is because it only accepts a new member of the population if it is more fit than its direct parent.

Our hypothesis is that the population based algorithms perform better than simple backpropagation on the data sets.

We will test each data set with an MLP with zero, one, and two hidden layers. We will test each of these with the genetic, differential evolution, and particle swarm optimization algorithms.

1.1 Data Sets and Preprocessing

All examples in the data sets are scrambled at random and then assigned to sets for ten-fold cross validation. All categorical variables are converted to integers for convenience. Our preprocessor also generates a similarity matrix for each variable to determine distances between categorical variables. All numerical variables are normalized between 0 and 1.

Data Set	# of Features	Model Type Needed
abalone[5]	4177	Regression
glass[3]	214	Classification
soybean-small[4]	47	Classification
breastcancer[6]	2310	Classification
machine[2]	209	Regression
forestfires[1]	512	Regression

1.2 Models

Each of these models are population based models, meaning they each start with a population of candidate solutions that search through the solution space for the best solution. The models vary based on their search algorithms.

Genetic Algorithm

The genetic algorithm has three stages. The first stage is candidate selection, where the top candidates for next stage are chosen. This can be done in a multitude of ways, including tournament style, where each of the candidates is compared against another in a tournament, weighted random processes, where candidates are randomly chosen, with higher probability weightings given to better solutions, or a multitude of other selection processes. This step is important, since the genetic diversity of future generations relies upon the selection process.

The second stage is the reproduction stage, where the candidate solutions share genetic material. The genetic material in this case would be the weights of the nodes on the MLP. There are a variety of ways for the candidate solutions to share genetic material, most of which involve randomly swapping genes directly. This can be done by randomly choosing the genes for the new generation with an equal probability of choosing from the either parent. Another way is to randomly assign lengths of genetic material to and alternate taking from each parent that length of genetic material. This is known as n-point crossover. There are also other ways that do not preserve order or that only preserve patterns.

The third step is to possibly mutate the genetic material. There are also a variety of ways to do this. One way would be to have a chance to mutate each gene by a certain amount. This would create two different hyperparameters that would be necessary tune. The goal of having a mutation step is to introduce diversity into the candidate solutions, which helps to reduce the potential of the solutions from falling into local minima instead of a global minima.

After the mutation step, these steps are repeated for the new generation of candidate solutions. We repeat the process until a suitable solution is found.

Differential Evolution

The differential evolution algorithm works by randomly combining components of the candidate solutions. The first step is to select a vector, and there are many ways to do this. One way is to iteratively select each solution vector. Another way is to randomly select a solution vector. Once a vector is selected, a new candidate solution is created as follows: Randomly select one component of the candidate solution to be generated differently from the base solution vector, then for each of the other components, randomly decide whether or not to adopt the value of the base solution vector based on a crossover rate. So, we are guaranteed that at least one of the candidate solution's components will be different than the base solution vector. Then, for each component that will be different, we calculate the value of the component using the following formula:

$$y_i = a_i + \beta(b_i - c_i)$$

Where y is the new candidate solution, a, b, and c are three random solution vectors that are not the base solution vector, and β is a hyperparameter scaling factor.

Once this candidate solution vector is generated, we compare against the base solution vector using our cost function. If the candidate solution vector has a lower cost, then it replaces the base solution vector. Otherwise, we toss out the candidate solution.

This is repeated until the solutions converge. Note that the solutions are not guaranteed to converge to the global minima, but rather only to local minima. Also note that the number of candidate solutions is always steady and is a hyperparameter that must be tuned.

Particle Swarm Optimization

The particle swarm optimization algorithm works by simulating a swarm of solutions that move throughout the solution space with velocities. The velocities are controlled by two points, the best known position of that specific particle, called the local best, and the best known position of all particles within the topology, called the global best. To start with, the particles are initialized with random positions within the solution space and the local and global best are recorded. Then, for each particle, its velocity is calculated via the following formula:

$$v_i(t) = v_i(t-1) + \phi_1 r_1 (lb - x(t)) + \phi_2 r_2 (gb - x(t))$$

Where ω, ϕ_1, and ϕ_2 are hyperparameters, both r_1 and r_2 are random numbers, $x(t)$ is the current position of the particle, and lb and gb are the local and global best. This new velocity is added to the current position of the particle to get it's new position.

The particles then move around the solution space until both local and global best are equivalent over all particles, and they settle into that point. It is important to note that the algorithm isn't guaranteed to converge to the global minimum.

1.3 Hyperparameter Tuning

Separate hyperparameters needed to be tuned for each algorithm. For the genetic algorithm, the standard deviated of the mutation creep needed to be tuned and the chance of mutation needed to be tuned. For the differential evolution algorithm, the crossover rate and the mutation β needed to be tuned. For the particle swarm optimization algorithm, weights of global and local best needed to be tuned. For most of these, we tuned using a grid search algorithm, however for the particle swarm optimization algorithm, we assumed that $\phi_1 + \phi_2 = 2$, and did a search with that constraint. All of the optimal values found are recorded in the tables in the results section.

1.4 Evaluation Metrics

In order to evaluate the results of our tests, we once again used accuracy, error, and mean squared error. For the classification problems, accuracy and mean squared error were used. For the regression problems, we used mean error and mean squared error. Accuracy tells us what percentage of the testing set was classified correctly, mean error tells us how far, based on Euclidean distance with the regression sets, or based on the similarity matrix for the classification sets, and mean squared error is simply the square of the mean error.

1.5 Results

Upon taking the results of the best performing number of hidden nodes for each data set, and comparing against the results of the previous project, we found that the population based algorithms performed better on the regression data sets, however they performed worse on the continuous data sets than backpropagation did. Recognizing that the Breast Cancer data from the backpropagation experiment is an outlier and disregarding it, the average difference in accuracy for continuous data sets is 23.8% in favor of backpropagation, whereas the average difference in mean squared error for the regression data sets, again noting that the Abalone data set is an outlier and disregarding it, is 1.06 in favor of the population based algorithms.

While the margin between the success rates of the backpropagation and the population based algorithms with respect to the continuous data sets was significant, the same cannot be said with respect to the regression sets. A difference in mean squared error of 1.06 can very easily change significantly given a different set of initial solution vectors. The implica-

Genetic Algorithm	#of Hidden Nodes	MSD	Mutation Chance	Accuracy	MSE
Breast Cancer[6]	0	0.1	0.005	86%	104.90
Breast Cancer[6]	1	0.1	0.005	85%	206.00
Breast Cancer[6]	2	0.1	0.005	70%	1074.00
Glass[3]	0	0.1	0.05	72%	3.91
Glass[3]	1	0.1	0.05	63%	6.20
Glass[3]	2	0.1	0.05	51%	13.55
Soybean-Small[4]	0	0.1	0.02	67%	0.01
Soybean-Small[4]	1	0.1	0.02	63%	0.00
Soybean-Small[4]	2	0.1	0.02	64%	0.00

Genetic Algorithm	#of Hidden Nodes	MSD	Mutation Chance	Mean Error	MSE
Abalone[5]	0	0.1	0.005	0.22	862.74
Abalone[5]	1	0.1	0.005	0.26	967.06
Abalone[5]	2	0.1	0.005	0.21	2122.44
Machine[2]	0	0.1	0.02	2.30	0.04
Machine[2]	1	0.1	0.02	1.21	0.19
Machine[2]	2	0.1	0.02	0.54	0.00
ForestFires[1]	0	0.1	0.05	1.92	0.30
ForestFires[1]	1	0.1	0.05	2.00	0.28
ForestFires[1]	2	0.1	0.05	1.84	0.25

Differential Evolution	#of Hidden Nodes	Crossover Rate	β	Accuracy	MSE
Breast Cancer[6]	0	0.25	1.5	83%	118.50
Breast Cancer[6]	1	0.25	1.5	79%	159.66
Breast Cancer[6]	2	0.25	1.5	70%	903.76
Glass[3]	0	0.25	1.5	60%	8.11
Glass[3]	1	0.25	1.5	50%	9.09
Glass[3]	2	0.25	1.5	40%	14.91
Soybean-Small[4]	0	0.5	1.0	72%	0.01
Soybean-Small[4]	1	0.5	1.0	74%	0.01
Soybean-Small[4]	2	0.5	1.0	76%	0.02

Differential Evolution	#of Hidden Nodes	Crossover Rate	β	Mean Error	MSE
Abalone[5]	0	0.5	1.5	0.22	1618.64
Abalone[5]	1	0.5	1.5	0.22	2444.74
Abalone[5]	2	0.5	1.5	0.18	2748.45
Machine[2]	0	0.5	0.5	0.77	0.13
Machine[2]	1	0.5	0.5	0.59	0.06
Machine[2]	2	0.5	0.5	1.01	0.08
ForestFires[1]	0	0.25	1.5	1.90	0.30
ForestFires[1]	1	0.25	1.5	1.90	0.23
ForestFires[1]	2	0.25	1.5	2.14	0.21

Table 1: Genetic and Differential Evolution Algorithm Results

Particle Swarm	#of Hidden Nodes	ϕ_1	ϕ_2	Accuracy	MSE
Breast Cancer[6]	0	2.0	3.0	86%	111.60
Breast Cancer[6]	1	2.0	3.0	87%	93.40
Breast Cancer[6]	2	2.0	3.0	90%	92.95
Glass[3]	0	2.0	1.0	68%	6.02
Glass[3]	1	2.0	1.0	62%	6.03
Glass[3]	2	2.0	1.0	67%	6.80
Soybean-Small[4]	0	1.0	2.0	68%	0.05
Soybean-Small[4]	1	1.0	2.0	66%	0.07
Soybean-Small[4]	2	1.0	2.0	72%	0.13

Particle Swarm	#of Hidden Nodes	ϕ_1	ϕ_2	Mean Error	MSE
Abalone[5]	0	3.0	2.0	0.22	1778.29
Abalone[5]	1	3.0	2.0	0.21	1808.33
Abalone[5]	2	3.0	2.0	0.22	1974.31
Machine[2]	0	1.0	3.0	20.66	0.26
Machine[2]	1	1.0	3.0	0.92	0.23
Machine[2]	2	1.0	3.0	1.76	0.19
ForestFires[1]	0	2.0	2.0	1.15	0.33
ForestFires[1]	1	2.0	2.0	1.09	0.32
ForestFires[1]	2	2.0	2.0	1.12	0.34

Backpropagation	#of Hidden Nodes	Step Size	Accuracy	MSE
Breast Cancer[6]	0	0.2	25%	316.80
Breast Cancer[6]	1	0.2	26%	248.81
Breast Cancer[6]	2	0.2	17%	4835.91
Glass[3]	0	0.1	90%	29.45
Glass[3]	1	0.1	99%	0.75
Glass[3]	2	0.1	100%	0.30
Soybean-Small[4]	0	0.1	90%	0.60
Soybean-Small[4]	1	0.1	88%	1.05
Soybean-Small[4]	2	0.1	88%	1.11

Backpropagation	#of Hidden Nodes	Step Size	Mean Error	MSE
Abalone[5]	0	0.1	0.00	0.65
Abalone[5]	1	0.1	-0.03	0.67
Abalone[5]	2	0.1	-0.02	0.71
Machine[2]	0	0.1	0.04	0.30
Machine[2]	1	0.1	0.06	0.28
Machine[2]	2	0.1	0.01	0.23
ForestFires[1]	0	0.1	0.31	2.76
ForestFires[1]	1	0.1	0.48	26.40
ForestFires[1]	2	0.1	0.40	2.23

Table 2: Particle Swarm Optimization and Backpropagation Results

tion is that, in general, it is difficult to say whether or not the population based algorithms do better than simple backpropagation with respect to regression data sets. However, with respect to categorical data sets, the margin is much larger. Because of this, we are forced to reject our hypothesis.

The backpropagation model took the least amount of time to converge, followed by differential evolution, then the genetic algorithm, then the particle swarm optimization model. This is different than expected, specifically with regard to the genetic algorithm. We had surmised that the genetic algorithm would take the longest, given the mutation factor. However, that was not the case. It is possible that the specific solution spaces were not conducive to the particle swarm optimization, though this is only conjecture.

2. Summary

The only evidence that supports the hypothesis was very weak, and the evidence against is was fairly strong, so the hypothesis was rejected. The difference between how the regression data sets and the categorical data sets performed over all the models suggests that, at the least, backpropagation is superior to population based models when considering categorical data.

References

[1] Paulo Cortez and Anibal Morais. *UCI Forest Fires*. 2007.

[2] Phillip Ein-Dor and Jacob Feldmesser. *UCI Relative CPU Performance Data*. 1987.

[3] B. German. *UCI Glass Identification Data Set*. 1987. URL: `https://archive.ics.uci.edu/ml/datasets/Glass+Identification`.

[4] R.S Michalski. *UCI Small Soybean Database*. 1987.

[5] Dr. Sam Waugh. *UCI Abalone Data*. 1995.

[6] Dr. William H. Wolberg. *UCI Wisconsin Breast Cancer Database*. 1991.