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.

Problem Statement -

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.

***This could be larger and include a bit of history if you need to beef things up***

Hypothesis -

POp based algorithms are expected to outperform backprop because they are less prone to being caught in local minima. It is likely that GA will perform better than DE and PSO 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 its mutation in GA has the possibility of introducing diversity, it also makes the GA similar to a random walk. This will give it the slowest convergence rate. DE will converge the quickest of three PBAs. This is because it only accepts a new member of the pop if it is more fit than its direct parent.

***This is where the algorithms should be explained, GA, DE, PSO***

Preprocessing -

All examples in the data set are scrambled at random and then assigned to sets for ten-fold CV same as the last project. All categorical variables are converted to integers for convenience. The preprocessor also generates a similarity matrix for each variable to determine distances between categorical variables. All numerical variables are normalized between 0 and 1. The preprocessor should also handle missing variables if need be.

Experimental Design Helper - Each population algorithm will contain a population consisting of complete feedforward networks parameterized by the weights of their layers. For ease of manipulation the weights of a feedforward network will be converted into single vectors of weight values. Each algorithm then performs successive manipulations on the vectors in each of the optimal feedforward networks. Whenever the fitness of a network is evaluated, the vector is converted back to a network structure. At the termination of each algorithm, the feedforward network with the best fitness is returned.

***This should be mostly copy/paste but if you could make it sound a little more professional through better grammar and remove abbreviations that would be appreciated.***

Evaluation metrics -

Classification datasets - accuracy and mean squared error (MSE).

The MSE metric implemented measures the squared error between the predicted and actual class distributions. Accuracy indicates how well the algorithm classifies examples individually.

Regression datasets - mean error (ME) and MSE

MSE takes the distance between real and predicted values and squares it. ME is obviously just not squared. MSE emphasizes the effect of outliers while ME captures whether the learner tends to over- or under-estimate the values in the test set. MSE and ME are computed using z-scores (the number of standard deviations from the mean) for comparisons.

***Pretty much the same stuff as we’ve been doing***

Tuning -

A grid search for each tuning parameter was implemented for this project.

Tuning the GA requires finding optimal standard deviation used for the mutation creep, Pe and Pm. Setting the standard deviation at 100\* the initial bound for the network weights turned out to be the best. The optimal values can be found in the second and third parameters for the raw output.

Tuning DE requires tuning the crossover rate, Pe, and mutation denoted B. The optimal values for each are found in the second and third parameters in the raw output.

Tuning PSO requires setting the values of Cc and Cs. PSO performs best when Cc = Cs = 2. A grid searched was performed in that fashion and the optimal parameters were selected based on fitness of the produced networks. The optimal values for each are found in the second and third parameters in the raw output.

Results -

With the final tuned parameters in place, backpropagation seemed to outperform the pop based algorithms on classification data sets while the opposite is true for regression. This slightly aligns with the hypothesis that PBA’s would perform better than backprop. This is because it is required that output layer weights are large, which is easier for a PBA to handle. Performance of every training method with 2 hidden layers had the worst performance, because of increased search space dimensionality. Backprop and PBA are neither free of this limitation. The greatest difference in performances between the two algorithms is shown on the forest fires. For a network trained with Backpropagation, the MSE was 26 while PBAs wass less than 3. There also seems to be no real preference between PBAs. The results vary by data set and layer. The results seem to suggest that the algorithms performance is situational. For convergence rate, that idea that backprop would be the fastest of all algorithms is true. DE is the next fastest, then GA, then PSO, as predicted.

***Let me know if there are any issues***

Raw Output: (Same format as Project 3)

DiFFEvolution

breastcancer.csv, 0, 0.25, 1.5, 0.83, 118.50

breastcancer.csv, 1, 0.25, 1.5, 0.79, 159.66

breastcancer.csv, 2, 0.25, 1.5, 0.70, 903.76

glass.csv, 0, 0.25, 1.5, 0.60, 8.11

glass.csv, 1, 0.25, 1.5, 0.50, 9.09

glass.csv, 2, 0.25, 1.5, 0.40, 14.91

soybean.csv, 0, 0.5, 1.0, 0.72, 0.01

soybean.csv, 1, 0.5, 1.0, 0.74, 0.01

soybean.csv, 2, 0.5, 1.0, 0.76, -0.02

abalone.csv, 0, 0.5, 1.5, 0.22, 1618.64

abalone.csv, 1, 0.5, 1.5, 0.20, 2444.74

abalone.csv, 2, 0.5, 1.5, 0.18, 2748.45

machine.csv, 0, 0.5, 0.5, 0.77, 0.13

machine.csv, 1, 0.5, 0.5, 0.59, 0.06

machine.csv, 2, 0.5, 0.5, 1.01, 0.08

forestfires.csv, 0, 0.25, 1.5, 1.90, 0.30

forestfires.csv, 1, 0.25, 1.5, 1.90, 0.23

forestfires.csv, 2, 0.25, 1.5, 2.14, 0.21

PSO

breastcancer.csv, 0, 2.0, 3.0, 0.86, 111.60

breastcancer.csv, 1, 2.0, 3.0, 0.87, 93.40

breastcancer.csv, 2, 2.0, 3.0, 0.90, 92.95

glass.csv, 0, 2.0, 1.0, 0.68, 6.02

glass.csv, 1, 2.0, 1.0, 0.62, 6.03

glass.csv, 2, 2.0, 1.0, 0.67, 6.80

soybean.csv, 0, 1.0, 2.0, 0.66, -0.05

soybean.csv, 1, 1.0, 2.0, 0.66, -0.07

soybean.csv, 2, 1.0, 2.0, 0.72, -0.13

abalone.csv, 0, 3.0, 2.0, 0.22, 1778.29

abalone.csv, 1, 3.0, 2.0, 0.21, 1808.33

abalone.csv, 2, 3.0, 2.0, 0.22, 1974.31

machine.csv, 0, 1.0, 3.0, 20.66, -0.26

machine.csv, 1, 1.0, 3.0, 0.92, -0.23

machine.csv, 2, 1.0, 3.0, 1.76, -0.19

forestfires.csv, 0, 2.0, 2.0, 1.15, -0.33

forestfires.csv, 1, 2.0, 2.0, 1.09, -0.32

forestfires.csv, 2, 2.0, 2.0, 1.12, -0.34

Genetic

breastcancer.csv, 0, 0.1, 0.005, 0.86, 104.90

breastcancer.csv, 1, 0.1, 0.005, 0.85, 206.00

breastcancer.csv, 2, 0.1, 0.005, 0.70, 1074.00

glass.csv, 0, 0.1, 0.05, 0.72, 3.91

glass.csv, 1, 0.1, 0.05, 0.63, 6.20

glass.csv, 2, 0.1, 0.05, 0.51, 13.55

soybean.csv, 0, 0.1, 0.02, 0.67, -0.01

soybean.csv, 1, 0.1, 0.02, 0.63, 0.00

soybean.csv, 2, 0.1, 0.02, 0.0.00

abalone.csv, 0, 0.1, 0.005, 0.22, 862.74

abalone.csv, 1, 0.1, 0.005, 0.26, 967.06

abalone.csv, 2, 0.1, 0.005, 0.21, 2122.44

machine.csv, 0, 0.1, 0.02, 2.30, -0.04

machine.csv, 1, 0.1, 0.02, 1.21, 0.19

machine.csv, 2, 0.1, 0.02, 0.54, 0.00

forestfires.csv, 0, 0.1, 0.05, 1.92, 0.30

forestfires.csv, 1, 0.1, 0.05, 2.00, 0.28

forestfires.csv, 2, 0.1, 0.05, 1.84, 0.25