Abstract - The MLP neural network is a supervised training method used for classification or regression. While applicable in various problems, it requires tuning of parameters to perform optimally. When tuned correctly, the network is capable of high performance.

Problem Statement -

Neural Networks are historical in the Machine Learning context. The backpropagation rule was published to move out of the realm of linear problems. They are composed of nodes that take in inputs with weights and sent to an output to another node in the network. The MLP network is used for universal function approximation. The network will be tuned to handle the dataset and performance will be evaluated. The task at hand is to find a model that predicts a real value or class based on a feature set.

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

Hypothesis -

The original task was to compare the MLP and RBF networks, however as RBF is no longer a requirement, we can only examine MLP on classification and regression data sets. MLP is expected to perform better on classification sets.

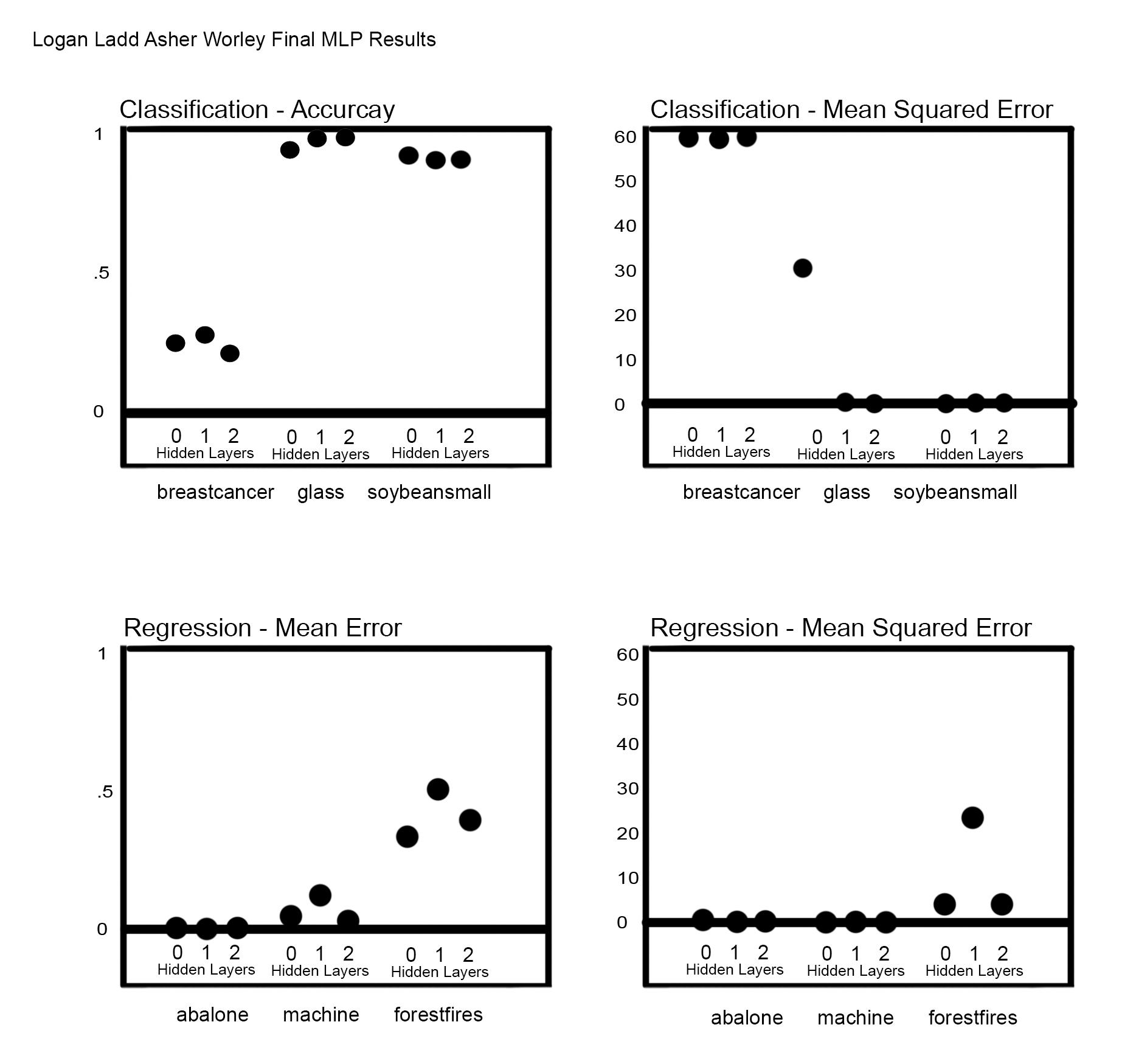
***This should not have to be changed (yay), see results.***

***This is where the algorithms should be explained, MLP, Gradient Descent,***

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.

***Copy/Pasted from project 2, change if need be***

Experimental Design Helper - The MLP has the basic structure of an input layer, hidden layers, and an output layer. Functionality of layers, nodes weights, and backpropagation are generic. Each layer may use linear or sigmoidal activation functions. The sig activation function in our case will be the logistic function. Again, MLP networks may have an arbitrary number of hidden layers with an arbitrary number of hidden nodes. Each MLP will require the number of both of these hidden entities as well as an activation function to store each layer. WEights of the layers will be initialized to a value between -.01 and .01. Tests will mostly use sigmoidal activation functions in the hidden layers to ensure the network functions as a universal approximator. Training with an arbitrary amount of examples is done in batches using backprop from the output through hidden layers. Momentum may be applied from here. The previous gradience vector that comes back from backprop is stored for usage for the next return from backprop. The networks detect the type of dataset we are using and the output layer is chosen upon training. A classification net will have an output node tied to each class in the dataset, each with a sigmoidal activation function. Predicted class will be chosen based on the node with the maximum activation value. A regression network will have a single output node with a linear activation function. Predicted values will be the activation value of this node given an input.

The training of each network is dependent on gradient descent , which is implemented in the backprop class. This class takes weights as well as a batch. For each example the algorithm will compute a prediction using the current weights. After the gradient for a weight is computed, A recursive structure is set up for backpropagation that can be applied to any number of hidden layers. Finally the gradient for each example in the batch is averaged and is returned to the learner for an update to the network.

***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 -

Tuning is done with the learning rate *n* which is the step size at each iteration in gradient descent. For classification sets, the initial value was set to 0.1, the other values tested include

.0001, .001, .01, .2, .3, .4, .5, 1, 2 and 4. Learning rates below 0.1 generally performed the worst and rates above .1 did not see better performance except on larger classification data sets because they contain significantly more points, so using the same number of iterations the learning rate can be larger to find the local minimum. Regression performed best with a learning rate of .1 generally the same as classification sets.

Another value tuned was the momentum multiplier. This value was tested at .5, .25, and 0. There was negligible change in the MSE when momentum was increased from .25 to .5, and accuracy practically experienced no change. Regression data sets as a whole performed the best with a momentum multiplier of 25.

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

Results -

With the tuned parameters, the data was run through the MLP algorithms for performance analysis. In general the MLP network was better on classification data sets compared to regression data sets. Our performance of these networks shows their effectiveness as universal approximators. As dimensionality increases, MLP requires more nodes in hidden layers and more data to successfully perform gradient descent during training.

Raw Output:

No Momentum

FILE: breastcancer.csv, 0, 0.2,0.25,0.25,316.80

FILE: breastcancer.csv, 1, 0.2,0.25,0.26,248.81

FILE: breastcancer.csv, 2, 0.2,0.25,0.17,4835.91

FILE: glass.csv, 0, 0.1,0.25,0.9,29.45

FILE: glass.csv, 1, 0.1,0.25,0.99,0.75

FILE: glass.csv, 2, 0.1,0.25,1.00,0.30

FILE: soybeansmall.csv, 0, 0.1,0.25,0.90.60

FILE: soybeansmall.csv, 1, 0.1,0.25,0.88,1.05

FILE: soybeansmall.csv, 2, 0.1,0.25,0.88,1.11

FILE: abalone.csv, 0, 0.1,0.5,0.65,-0.00

FILE: abalone.csv, 1, 0.1,0.5,0.67,-0.03

FILE: abalone.csv, 2, 0.1,0.5,0.71,-0.02

FILE: machine.csv, 0, 0.1,0.5,0.30,0.04

FILE: machine.csv, 1, 0.1,0.5,0.280.06

FILE: machine.csv, 2, 0.1,0.5,0.23,0.01

FILE: forestfires.csv, 0, 0.1,0.5,2.76,0.31

FILE: forestfires.csv, 1, 0.1,0.5,26.40,0.48

FILE: forestfires.csv, 2, 0.1,0.5,2.23,0.40

Momentum

FILE: breastcancer.csv, 0, 0.2,0.25,0.25,316.80

FILE: breastcancer.csv, 1, 0.2,0.25,0.26,248.81

FILE: breastcancer.csv, 2, 0.2,0.25,0.17,4835.91

FILE: glass.csv, 0, 0.1,0.25,0.9,29.45

FILE: glass.csv, 1, 0.1,0.25,0.99,0.75

FILE: glass.csv, 2, 0.1,0.25,1.00,0.30

FILE: soybeansmall.csv, 0, 0.1,0.25,0.90.60

FILE: soybeansmall.csv, 1, 0.1,0.25,0.88,1.05

FILE: soybeansmall.csv, 2, 0.1,0.25,0.88,1.11

FILE: abalone.csv, 0, 0.1,0.5,0.65,-0.00

FILE: abalone.csv, 1, 0.1,0.5,0.67,-0.03

FILE: abalone.csv, 2, 0.1,0.5,0.71,-0.02

FILE: machine.csv, 0, 0.1,0.5,0.30,0.04

FILE: machine.csv, 1, 0.1,0.5,0.280.06

FILE: machine.csv, 2, 0.1,0.5,0.23,0.01

FILE: forestfires.csv, 0, 0.1,0.5,2.76,0.31

FILE: forestfires.csv, 1, 0.1,0.5,26.40,0.48

FILE: forestfires.csv, 2, 0.1,0.5,2.23,0.40