

Project 4: Networks for Classification and Regression

Sarah Wilson

303-921-7225

Engineering Professionals Computer Science

Johns Hopkins University

Baltimore, MD 21218, USA

SWI1S117@JHU.EDU

1. Introduction

Neural Networks are collection of algorithms that are taught the underlying patterns in a data set through a process of supervised learning. Neural Networks operate using a process that is similar to the human brain, they contain simple processing nodes, called neurons that are connected in layers. These layers contain weights that determine an output based on a given input. The basis of an understanding of Neural Networks actually starts with Linear Regression.

Linear Regression is a supervised Machine Learning algorithm that can be applied to Regression problems. Linear Regression assumes there is some sort of linear relationship between the input data and the output value that can be represented by a line or plane of best fit between the inputs and the output. Linear Regression is meant for continuous outputs, when Classification tasks are required (tasks that have outputs with class labels), then the concepts of linear regression can be applied, but instead the Sigmoid function is used, hence the name Logistic Regression. The sigmoid function is able to take classifiers that are not numeric and map them to values between 0 and 1. Linear and Logistic regression are the basic concepts that build the foundation for Neural Networks. Neural Networks can be thought of as a series of Logistic regression tasks that happen across many layers and nodes. Neural Networks can also be augmented through the use of developing an Auto Encoder. The Auto Encoders are trained such that an output pattern to a network always matches the input pattern. The Auto Encoder process can be thought of a method to fine tune the weights feeding the prediction output from a Neural Network.

The problem statement presented in this paper is to understand and implement three major categories of networks; Linear, Feed-Forward Back Propagation and Feed-Forward Back Propagation with an Auto Encoder. For the Linear Network: a Linear Regression will be applied to Regression problems while a Logistic Regression will be applied to the Classification Problems.

The hypothesis of this report is that the Feed-Forward Network with an Auto Encoder will perform better than the Feed-Forward Back Propagation and Linear regression approaches on both the Classification and Regression problems. The reason for this hypothesis is that the Auto Encoder builds upon the Linear Regression and basic Neural Regression. Both Linear Regression and Neural Networks rely on updating weight based on the error between the predicted output and the actual output of a training set. The Auto Encoder improves upon this process by trying to learn and mimic the input pattern of the data and apply that to the output or predicted value. This extra encoding and decoding steps, serve to fine tune the weights feeding the neural network. The hypothesis is based

around the concept that fine tuned weights should outperform coarsely tuned weights in making predictions based on new input data to the network. The results from experiments ran on the provided data sets will be discussed and presented against the outlined hypothesis.

Section 1 has provided the introduction, problem statement and hypothesis in regards to three major learning algorithms that will be explored. Section 2 will provide an in-depth explanation of the algorithms. Section 3 will present the results obtained by the different algorithms. Section 4 will discuss the results that were obtained and compare them to the hypothesis that was outlined in the introduction. This report will conclude in Section 5 with a discussion of lessons learned and areas of possible future work.

2. Algorithms and Experimental Methods

Linear and Logistic Regression

Regression is an a problem solving approach in Machine Learning that models a target predictor based on independent variables or values. Linear Regression and Logistic Regression both are discriminant-based approaches that do not care about estimating the densities inside a particular class region, instead the discriminant-based approach aims to estimate the boundaries between the class regression. The output of Linear and Logistic regression models is a weighted sum of the input attributes or features. The magnitude of the weight applied against each feature, shows the importance of that feature towards the overall prediction. The weights are learned by the model during training. Linear Regression is used on Regression problems because the data that is being predicted is continuous. This Linear model will not work for Classification problems, as the output of the linear model would be a real value, when in Classification a class label is desired. Logistic Regression is used for Classification problems, as it maps the output of the weight sum presented in Linear Regression and maps it to a non-linear function that ensures the outputs are between 0 and 1. The function used to achieve this is the Sigmoid or Logistic function.

The objective of training in both Linear and Logistic Regression is to determine the weights that will be used in the final model. The weight coefficients can be determined using the process of Gradient Descent. The general procedure for applying Gradient Descent is to first calculate the prediction based on the current weight coefficients and then calculate new coefficients based off the error in the prediction. This process is repeated until the coefficients are deemed accurate enough. The weight convergence process used in this paper to determine if the prediction is accurate enough will be to a set number of iterations. After the set number of iterations through the weight update process, the weights will be deemed good enough. At this point the model can be considered trained and will be evaluated for performance on the test data set.

Linear and Logistic Regression Algorithm

For each Feature (j) in the Domain(d) of the Data Set:

Set W_j to a random value between -0.01 and 0.01

Repeat the Following Steps Until the Number of Iterations for Convergence has been Reached

For each Feature (j) in the Domain(d) of the Data Set:

$$\delta W_j = 0$$

For each observation (t) in the Data Set:

Set the Temp Prediction (o) to 0.

For each Feature (j) in the Domain(d) of the Data Set:

$$(o) = (o) + W_j * X_j$$

If the problem is Regression:

Set the Model Prediction (y) to the Temp Prediction Value (o)

If the problem is Classification:

Set the Model Prediction (y) to the Sigmoid(o)

For each Feature (j) in the Domain(d) of the Data Set:

Linear Neural Networks

With a basic understand of the Linear and Logistic regression algorithms, these processing techniques can be expanded to build a Neural Network. For the purposes of this paper the Linear Neural Network will be built with a Input Layer, Two Hidden Layers and the Final Output Layer. The structure of a Neural Network is something that can be tuned and tweaked to provide different results, that fell outside the scope of work for this paper, however provides an area of future work. The structure used for the Neural Network in this paper, has the number of nodes in each layer being equal to the number of features in the data set. In other words, the number of nodes in the hidden layers is equal to the domain of the data set. The neurons in each of the layers are connected by edges and those edges are tied to weight values. The algorithm explored in this paper is Back Propagation. The general idea behind Back Propagation is to model a pattern or model in the data set by modifying the weight of the internal network structure. This network is trained using supervised learning, where the error between the predicted value and the actual output value of an observation, is used to back propagate and update the weights of the edges inside the Network.

An observation is feed forward through the network, using an initial set of weights. As each feature in the input data set passes through a Neuron the Neuron is activated, as in the input feature times the weight connecting the feature to that neuron is calculated and then passed through the Sigmoid function. The Sigmoid function is used for the activation function in the neurons for both Classification and Regression problems. The difference between the Classification and regression problems comes about in the output layer of the network.

Linear Neural Networks Algorithm

INSERT Algorithm

AutoEncoded Linear Neural Networks

INSERT Description

AutoEncoded Linear Neural Networks Algorithm

INSERT Algorithm

Data Sets

The following data sets were used during the classification and regression tasks for this project.

Breast Cancer

Description:

Task: Classification

Predictor: Diagnosis (Malignant or Benign)

Link:

<https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29>

Car Evaluation

Description:

Task: Classification

Predictor: Car Evaluation (Unacceptable, Acceptable, Good, Very Good)

Link:

<https://archive.ics.uci.edu/ml/datasets/Car+Evaluation>

Congressional Vote

Description: 1984 United States Congressional Voting Records

Task: Classification

Predictor: Party (Republican / Democrat)

Link:

<https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records>

Albalone

Description: Physical measurements of Albalone

Task: Regression

Predictor: Rings (int)

Link:

<https://archive.ics.uci.edu/ml/datasets/Abalone>

Computer Hardware

Description: Relative CPU performance data.

Task: Regression

Predictor: PRP

Link:

<https://archive.ics.uci.edu/ml/datasets/Computer+Hardware>

Forest Fires

Description: Forest Fire burn area data

Task: Regression

Predictor: Area (float)

Link:

<https://archive.ics.uci.edu/ml/datasets/Forest+Fires>

3. Results

Tables 1-3 display the results from Classification Tasks: the Breast Cancer, Car Evaluation and Congressional Vote data sets while running all three algorithms. Tables 4-6 display the results from the Regression Tasks: the Albalone, Computer Hardware and Forest Fire data sets while running all three algorithms. Each algorithm was tuned prior to being run on the full experiment set and optimal values were chosen out of that tuning process for the Learning Rate and the Convergence Factor. These tables show the results obtained during each fold of the k -fold validation process. The results from each iteration of the tuning process was omitted from this report for brevity. The error used is Classification error and can be described as the Number of Times the Prediction was Wrong / Total Number of Comparisons. This error was averaged across the 5 folds to provide the Average Classification Error against each of the Classification data sets. The error used for Regression Tasks was the Mean Root Squared Error. This error was averaged across the 5 folds to provide the Average Mean Root Squared Error against each of the Regression data sets.

Table 1: Car Evaluation:Linear Networks - Experimental Results

Table 2: Breast Cancer: Linear Networks - Experimental Results

Breast Cancer: Logistic Regression		Breast Cancer: Linear NN		Breast Cancer: Autoencoded NN	
	Learning = 0.001 Convergnace = 10		Learning = 1 Convergnace = 10		Learning = 0.1 Convergnace = 100
k-fold[1]	0.0440	k-fold[1]	0.3240	k-fold[1]	0.6750
k-fold[2]	0.0600	k-fold[2]	0.6868	k-fold[2]	0.6860
k-fold[3]	0.0500	k-fold[3]	0.3400	k-fold[3]	0.3400
k-fold[4]	0.2900	k-fold[4]	0.6219	k-fold[4]	0.3780
k-fold[5]	0.1300	k-fold[5]	0.3700	k-fold[5]	0.6290
Average Classification Error Across Folds	0.1100	Average Classification Error Across Folds	0.4687	Average Classification Error Across Folds	0.5420

Table 3: Congressional Vote: Linear Networks - Experimental Results

Congressional Vote: Logistic Regression			Congressional Vote: Linear NN			Congressional Vote: Autoencoded NN	
	Learning = 0.001 Convergnace = 100			Learning = 0.001 Convergnace = 100			Learning = 1 Convergnace = 10
k-fold[1]	0.4000		k-fold[1]	0.6079		k-fold[1]	0.6079
k-fold[2]	0.4140		k-fold[2]	0.3880		k-fold[2]	0.6115
k-fold[3]	0.3285		k-fold[3]	0.4100		k-fold[3]	0.4101
k-fold[4]	0.3768		k-fold[4]	0.6021		k-fold[4]	0.3978
k-fold[5]	0.4492		k-fold[5]	0.3799		k-fold[5]	0.3799
Average Classification Error Across Folds	0.3930		Average Classification Error Across Folds	0.4777		Average Classification Error Across Folds	0.4810

Table 4: Albalone: Linear Networks - Experimental Results

Table 5: Computer Hardware: Linear Networks - Experimental Results

Table 6: Forest Fires: Linear Networks - Experimental Results

4. Discussion

The hypothesis presented in this report was that the Feed-Forward Network with an Auto Encoder will perform better than the Feed-Forward Back Propagation and Linear regression approaches on both the Classification and Regression problems.

Looking at the classification data sets,

Looking at the Regression data sets,

5. Conclusion

Overall there seemed to be some trends that supported the general hypothesis.

The .

A suggested area of future work, after fixing basic implementation details, would be to explore as part of the tuning process the shape of the network. In the experiment the Neural Network that was built for the algorithm that included an input layer, 2 hidden layers and an output layer. This network could have been built with a different number of hidden layers and a variable number of nodes within those hidden layers, that could have led to different results. That structure of the network is an area that could be looked into for future work.

6. References

1. Alpaydin, E. (2004). Introduction to machine learning (Oip). Mit Press.