Programming Project #04 – Neural Networks

Jacob M. Lundeen JLUNDEE1@JHU.EDU

Department of Data Science
Johns Hopkins University
Baltimore, MD 21218

Abstract

This paper describes twelve experiments utilizing six data sets (three classification and three regression). Each data set is run through three separate algorithms: Linear Discrimination, Feedforward Neural Network with backpropagation, and an Autoencoder. Both neural networks utilize two hidden layers with the number of nodes tuned for each data set. We found that the linear functions outperformed both neural networks across all six data sets.

1. Introduction

Artificial Neural Networks (ANNs), or more commonly referred to as Neural Networks (NNs) are a family of algorithms inspired by the biological neural networks that make up human brains. The idea was first introduced in 1943 by Warren McCulloch and Walter Pitts. The idea behind the NN is that the network is made up of artificial neurons which are connected to other neurons (like synapses in the brain). The neurons receive a signal, or in this case a real number, process it, and then send a new signal to the other neurons. The NN is trained by providing inputs to the first layer, which run through at least one "hidden layer", and then a result is produced. The training is terminated based upon a predetermined criteria (this is supervised learning).

For this experiment, four separate algorithms are going to be tested. A standard Feedforward Neural Network (FNN), an Autoencoder (AE), and the standard linear and logistic regression algorithms. Both the FNN and AE will utilize backpropagation and their stopping criteria will be the number of iterations run.

Problem Statement

This experiment will compare the effectiveness of the FNN and the AE. The linear and logistic regression algorithms will be utilized to provide a baseline. My hypothesis is that while the FNN and AE will both outperform the linear and logistic algorithms, the AE will only provide marginal improvement over the FNN and will not provide any improvements in computational cost (if anything the AE will take longer). My reasoning for this hypothesis is that NNs are very good at finding the underlying patterns in data and can use that pattern to predict more accurately. For the AE not performing better than the FNN, my reasoning is that the computational costs of training the encoder/decoder portion and then training the prediction portion only complicate the learning process and do not provide additional insight into the underlying pattern in the data.

2. Algorithms and Experimental Methods

Data Sets

The data sets used are the same six used previously: Abalone, Breast Cancer, Car, Forest Fires, House Votes, and Machine.

The abalone data set is a regression set used to predict the age of an abalone from its physical characteristics.

The breast cancer set is a classification set used to classify a tumor as benign or malignment based on characteristics of the tumor.

The car dataset is a classification problem to determine acceptability of a car based on certain specifications.

The forest fire data set is a regression problem looking to predict the burned area of forest fires utilizing meteorological and other data.

The house votes data set is a classification problem attempting to classify congressmen as republican or democrat.

The machine data set is a regression problem to estimate performance of computer hardware.

Pre-Processing

Pre-processing of the data is handled the same as in the previous experiments. The data is read in, and missing values are handled by inputting them with the mean of their respective column. ‘Class’ and ‘target’ variables are moved to the end of the data sets so the algorithm can automatically locate them. The three regression data sets are normalized. All categorical variables are either encoded with dummy variables (dropping the first dummy) or, if they are ordinal, one hot encoded. Lastly, the forest data set’s target variable ‘area’ is skewed, so the variable is log transformed to remove the skew.

Experimental Approach

Linear and Logistic Regression

To start, we are going to implement a standard logistic regression algorithm for the classification data sets and a standard linear regression algorithm for the regression data sets. Both algorithms will be implemented utilizing linear discrimination, where we assume a model directly for the discriminant, bypassing the estimation of likelihoods or posteriors. This approach assumes on the form of the discriminant between the classes and makes no assumption about, or requires no knowledge of the densities (Alpaydin, 2020). The linear discriminant is popular because it is simple, and both the time and space complexities are O(d) (Alpaydin, 2020).

Before continuing, we need to discuss Gradient Descent (GD). GD is an optimization algorithm that is used when training machine learning models (Donges, 2021). GD finds the values of a

function's parameters (or coefficients) that minimize a cost function as far as possible. A "gradient" is a measure of the change in all the weights regarding the change in error. The higher the gradient, the faster a model can learn (it can learn too fast). GD is hiking down to the bottom of a valley until you get to the very bottom.

GD will be used throughout this experiment in both the linear discriminant algorithms as well as the FNN and AE algorithms.

The logistic discrimination algorithm pseudocode is below (Alpaydin, 2020):

$$
\begin{aligned}
&For\ j = 0, \dots, d \\
&\quad w_j \leftarrow rand(-0.01, 0.01) \\
&Repeat \\
&\quad For\ j = 0, \dots, d \\
&\quad\quad \Delta w_j \leftarrow 0 \\
&\quad For\ t = 1, \dots, N \\
&\quad\quad o \leftarrow 0 \\
&\quad\quad For\ j = 0, \dots, d \\
&\quad\quad\quad o \leftarrow o + w_j x_j^t \\
&\quad\quad y \leftarrow sigmoid(o) \\
&\quad\quad For\ j = 0, \dots, d \\
&\quad\quad\quad \Delta w_j \leftarrow \Delta w_j + (r^t - y) x_j^t \\
&\quad For\ j = 0, \dots, d \\
&\quad\quad w_j \leftarrow w_j + \eta \Delta w_j \\
&Until\ convergence
\end{aligned}
$$

In the above pseudocode $\Delta w_j \leftarrow \Delta w_j + (r^t - y)x_j^t$ is the GD equation. It iterates over the dimensionality of the data set, taking the difference between the true value (r^t) and y, which is the estimator. 'y' is calculated using the activation function $o \leftarrow o + w_j x_j^t$ and *sigmoid()*. The sigmoid function, as seen below, is used to compress the output of the activation function to be between [0, 1]:

$$S(x) = \frac{1}{1 + e^{-x}}$$

For the linear discriminant, the activation function is:

$$\phi(v) = a + \boldsymbol{v}'b$$

The above algorithm is for a data set with only two classes. For cases where K > 2, we must use a different activation function called the *softmax*:

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

At K = 2, the softmax function operates as the sigmoid function.

Feedforward Neural Network

An FNN is a NN wherein the connections between the nodes do not form a cycle. The flow of information only flows forward, from the input nodes, through the hidden nodes/layers, and to the output nodes. For this experiment, backpropagation is used with both the FNN and AE, which will be explained in detail later.

For the FNN and AE, the *perceptron* is our basic processing unit. The inputs can be from the data set or the outputs of other perceptron's. Each input, $x_j \in R, j = 1, \ldots, d$ has an associated connection weight $w_j \in R$, and the output, y, is a weighted sum of the inputs (Alpaydin, 2020). Figure 1 shows a simple perceptron.

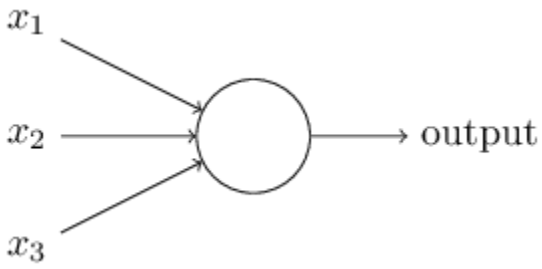

Figure 1 - Simple Perceptron

For this FNN and AE, we are creating two hidden layers as shown in Figure 2 (Dabbura, 2019):

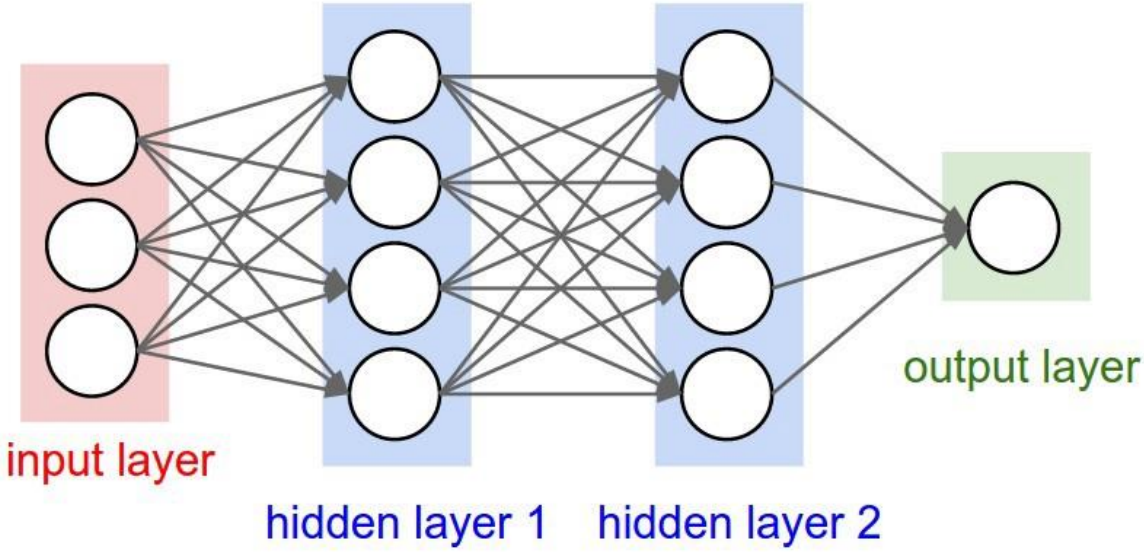

Figure 2 - Two Hidden Layer Neural Network

At each hidden layer and at the output, the information that it receives is run through an activation function. This is done to introduce non-linearity into the data. We do this because if the data was to remain linear, then each layer would simply be a linear combination of the previous layer(s). If this happened, no matter how many layers/nodes were used, the NN would simply act as a single layer perceptron. For this experiment, the activation function used varies. For both classification and regression, the sigmoid function is used for both hidden layers. For the output layer, the softmax function is used for classification to make sure that the output for each example is between [0, 1] and that they sum to 1. If the output is continuous, then the activation function on the output layer is linear.

Backpropagation

Finally, we incorporate Backpropagation (BP). BP is fine-tuning the weights of a NN based on the loss (Cross Entropy (CE) for classification and Mean Squared Error (MSE) for regression) obtained in the previous iteration. By doing this, we ensure lower error rates, which makes the model more reliable. Below is the pseudocode used (Alpaydin, 2020):

$$
\begin{aligned}
&Initalize\ all\ v_{ih}\ and\ w_{hj}\ to\ rand(-0.01, 0.01)\\
&Repeat\\
&\quad For\ all\ (x^t, r^t) \in X\ in\ random\ order\\
&\quad\quad For\ h = 1, \dots, H\\
&\quad\quad\quad z_h \leftarrow sigmoid\left(\boldsymbol{w}_h^T \boldsymbol{x}^t\right)\\
&\quad\quad For\ i = 1, \dots, K\\
&\quad\quad\quad y_i = \boldsymbol{v}_i^T \boldsymbol{z}\\
&\quad\quad For\ i = 1, \dots, K\\
&\quad\quad\quad \Delta \boldsymbol{v}_i = \eta\left(r_i^t - y_i^t\right)\boldsymbol{z}\\
&\quad\quad For\ h = 1, \dots, H\\
&\quad\quad\quad \Delta \boldsymbol{w}_h = \eta\left(\sum_i \left(r_i^t - y_i^t\right) v_{ih}\right) z_h(1 - z_h)\boldsymbol{x}^t\\
&\quad\quad For\ i = 1, \dots, K\\
&\quad\quad\quad \boldsymbol{v_i} \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i\\
&\quad\quad For\ h = 1, \dots, H\\
&\quad\quad\quad \boldsymbol{w}_h \leftarrow \boldsymbol{w}_h + \Delta \boldsymbol{w}_h\\
&Until\ Convergence
\end{aligned}
$$

Autoencoder

Autoencoders (AEs) are Multilayer Perceptron (MLPs) where there are as many outputs as there are inputs, and the required output is defined to be equal to the input. When the number of hidden units is less than the number of inputs, this implies dimensionality reduction (Alpaydin, 2020). An AE reduces data dimensionality by learning how to ignore the noise in the data.

The first layer acts as the encoder, with the hidden layer acting as the code. The MLP must then find the best representation of the input layer. The second layer is the decoder that reconstructs the code. Once the AE is trained, the output from the AE is then turned into the input for the

prediction layer (using one hidden layer), which is then trained. This prediction layer utilizes the same setup as the FNN previously discussed.

Tuning

For the logistic and linear regression algorithms, there are two hyperparameters that need to be tuned: number of iterations and η (learning rate). For the NNs, a third hyperparameter is H (number of nodes in the hidden layers). To tune these, nested for loops to iterate over a range of values for each hyperparameter using the validation set. The hyperparameters that minimized the loss for the FNN and AE were chosen.

3. Results

Tables 1, 2, and 3 show the results of the three algorithms on the regression data sets:

Table 1 - Results of Linear Regression

	Linear Regression				
	Iterations	η	R^2	Pearson	MSE
Abalone	200	0.112	0.509	0.715	0.492
Forest	100	0.001	0.005	0.079	0.993
Machine	400	0.001	0.863	0.941	0.145

Table 2 - Results of FNN on Regression Data Sets

	Feed Forward NN				
	Iterations	α	First Hidden Layer	Second Hidden Layer	MSE
Abalone	100	0.001	7	1	1.192
Forest	1000	0.01	1	1	1.311
Machine	100	0.00001	31	11	1.027

Table 3 - Results of Autoencoder on Regression Data Sets

	Autoencoder				
	Iterations	α	First Hidden Layer	Second Hidden Layer	MSE
Abalone	500	0.001	1	4	2.318
Forest	100	0.0000001	1	9	1.897
Machine	1000	0.0000001	1	26	1.644

Tables 4, 5, and 6 show the results on the classification data sets:

Table 4 - Results of Logistic Regression

	Logistic Regression			
	Iterations	η	μ	Accuracy
Breast Cancer	600	0.01	0.01	0.770
Cars	600	0.01	0.258	0.704
House	500	0.258	0.258	0.965

Table 5 - Results of FNN on Classification Data Sets

	Feed Forward NN				
	Iterations	α	First Hidden Layer	Second Hidden Layer	Accuracy
Breast Cancer	100	0.001	19	9	0.503
Cars	500	0.001	1	3	0.653
House	100	0.001	21	61	0.605

Table 6 - Results of Autoencoder on Classification Data Sets

	Autoencoder				
	Iterations	α	Encoder Hidden Nodes	Prediction Hidden Nodes	Accuracy
Breast Cancer	100	0.1	1	5	0.469
Cars	100	0.1	1	1	0.699
House	100	0.001	1	1	0.613

All algorithms were used with 5x2 Cross Validation and the MSE and Accuracy scores are the mean values of the cross validation.

4. Discussion

Regression Data Sets

For the three regression data sets, my original hypothesis was shown to be incorrect regarding how the FNN and AE performed compared to the linear discriminant algorithm. For all three data sets, the linear discriminant outperformed both the FNN and the AE. In fact, the AE performed the worst of all three by a considerable margin.

An interesting observation is the number of nodes for the hidden layers for both the FNN and AE. For the forest data set, the tuning settled at one hidden node for both layers. This would normally be considered a trivial solution, but even after changing the seed value numerous times, the tuning always came out to one node per layer. The machine data set was also unique in that for both the FNN and the AE, it settled on a high number of nodes for one of its layers. For the AE, it was interesting that the tuning settled on a single node for hidden layer of the AE.

Classification Data Sets

For the three classification data sets, my original hypothesis was shown to be incorrect again, albeit it being closer. The logistic regression did very well with the house voting data set and over 70% accuracy on the other two data sets. Neither the FNN nor the AE had an accuracy score over 70%.

Like the regression sets, there was some interesting tuning of the nodes for the hidden layers. Most surprisingly the FNN for the voting house data set settled on 61 hidden nodes for the second layer, which is more nodes than features. While there is nothing saying you cannot have more nodes than features, it is accepted that you do not have more nodes features. With the AE, both the cars and house voting data sets settled at a single node for both hidden layers.

5. Conclusion

The linear discriminant functions outperformed the FNN and AE across all six data sets. Both linear functions could by improved through feature and/or sample reduction. The FNN and the AE performed much worse than expected. Their performance could be improved like the linear functions, but I think what would improve their performance the most would be more extensive tuning of the hyperparameters.

Acknowledgements

I would like to acknowledge the love and support of my wife, Dr. Jordan S. Lundeen, and our wonderful family: Logan, Sydney, Mavis, Thea, Hermione, Ron, and Ginny. Without their unyielding support of my endeavors, I am not sure I would have made it this far. I would also like to thank Johns Hopkins for taking a chance on me.

References

Alpaydin, E. (2020). *Introduction to Machine Learning, fourth edition (Adaptive Computation and Machine Learning series)* (fourth edition). The MIT Press.

Donges, N. (2021, August 1). *Gradient Descent: An Introduction to 1 of Machine Learning's Most Popular Algorithms*. Built In. https://builtin.com/data-science/gradient-descent

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*(4), 115–133. https://doi.org/10.1007/bf02478259