Neural Networks

Tommy Baird tbaird7@jhu.edu
Johns Hopkins University
Baltimore, MD 21205, USA

Editor: Tommy Baird

Abstract

In this paper we will discuss the application of a neural network machine learning algorithm to predict the outcomes of both classification and continuous targets. This application will be tested on 6 different data sets of varying lengths and complexity. We will look at simple, single layer regression models, multi-layer models incorporating hidden layers with activation functions, and an auto-encoder that will be transposed to a multi-layer model for testing.

Keywords: Machine Leaning, Neural Networks, Linear Regression, Logistical Regression, Back-propagation, Auto-encoder

1 Introduction

In this experiment, we will be creating three different neural networks for six different data sets for which we will test their ability to predict target values. Generally, we believe that the networks will perform better will more layers as the extra transformations should only help their predictive power. However, we will make the prediction that the Computer Hardware and Forest Fire data sets will show more improvements in their multiple layer adaptations relative to the simple models. We believe this due to their apparent exponential outputs that do not seem to grow in a linear fashion. Therefore, we hope the non-linear transformations will provide the most benefit to their predictions. The Breast Cancer, Congress Voting, and Car Evaluation test sets are all classification sets, so we still believe that they can be properly bucketed with a linear model. The Abalone test set on the other hand will show similar improvement to the categorical sets, but its target values seem to have relatively clean distributions and a large sample set that should allow it to be properly evaluated with a simple regression model.

Lastly, we believe that the model that is first tuned with the auto-encoder will have performance somewhere between the simple and other multiple layer models. We will discuss more of our experimental approach below, but for the sake of providing a hypothesis, our attack for this auto-encoder trained model should provide a hybrid approach to our other two. Once we have trained the auto-encoder and attached its other hidden layer and output layers, we will not be re-training those original weights from our input layer. Therefore, they will be trained to output close replicas of our input values, and we will only have one hidden layer that will be trained to produce the proper output. Thus, if we have already predicted some improvement across the board from single to multiple layer networks, then

we would expect some in-between performance improvement from the network that acts like one with only a single hidden layer.

2 Experimental Approach

We created our networks with the help of pandas and numpy libraries within python. Our networks were stored as data frames where the weights from one layer to the next are stored at the next layer's node. In other words, the pointer from the input layer to the first hidden layer is stored on the hidden layer. We chose this for ease in updating these weights during our back-propagations.

We trained our networks until we reached our version of convergence. Convergence was tested every three rounds of updates. For tuning, convergence was measured as either we did not reach a new best performance, our new best performance is less than a 1% improvement, or we already ran 15 total tests. For tests, convergence was updated to check for new best performance of less than 2% and we allowed for 30 tests. In tuning, we cared more for proving a hyper-parameters efficacy rather than letting it train completely, so 15 tests seemed like a reasonable amount of time to prove its worth. We expanded this to 30 for testing to ensure that the network had sufficient time to train fully, but we increased our band to 2% for required performance improvement to ensure that this increased slack in testing time did not lead us to over-fitting the model to our training data.

As discussed in the project description, our loss was measured with mean squared error for our regression data sets and cross entropy loss was used for our categorical data sets. For our activation function within our hidden nodes, we used the logistic function which is shown below for reference. Within the confines of our network, x would be the weighted input from the previous layer and $f(x)$ would be the output to be fed to the next layer. This activation function is performed at each hidden node

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

When we turned our attention to the auto-encoder, we chose to only train the re-attached layers of hidden nodes rather than adjusting the weights from the initial input layer. As mentioned in the introduction, this would provide us a potential new perspective from the other two tests as it would have the first hidden node activation function try to mimic the input and the second hidden node activation function try to mimic the output. This should behave as if we had a single hidden layer which we proposed would result in an in-between performance improvement from simple to multiple layers.

2.1 Data Assumptions Made

There were very few data assumptions made during the processing of these tests, but it is worth noting that we normalized our continuous variables, and we employed one hot encoding when faced with categorical variables. When splitting data sets, we ensured that our categorical test sets were properly stratified.

2.2 Tuning

For tuning, we provided three options for the learning rate (.001, .01, .1) and two options for the percentage of hidden nodes that we employed relative to the size of the inputs (50% and 75%). There were a wide range of other possibilities to choose from, but these options seemed to provide sufficient life to the model training. However, it is worth noting that some tuning results were so large for the abalone and forest fire test sets that they were omitted. Thankfully, the other options worked well in their place.

For the simple models, we were only concerned with testing out our learning rate. Those results are shown in Table 1 and Table 2. There was a preference for .01 in most cases, but the congress voting set chose a larger rate of .1 and forest fires went smaller with .001.

Table 1: Categorical Simple Network Tuning

Learning Rate	Breast Cancer	Congress Voting	Car Eval
0.001	0.156	0.086	0.270
0.010	0.062	0.036	0.204
0.100	0.027	0.054	0.301

Table 2: Regression Simple Network Tuning

Learning Rate	Abalone	Computer Hardware	Forest Fires
0.001	7.124	9895.482	1.485
0.010	4.960	2302.460	40.057
0.100		1060.883	

For the multi-layer models, we also considered the percentage of hidden nodes relative to the input layer as shown in tables 3 and 4. There was more variation in the hyper-parameters that were chosen here, but it seems like the categorical models preferred higher learning rates and the regression models preferred lower learning rates. The main outlier here is the forest fires example that chose a higher rate of .1. For the hidden nodes, all of the sets preferred the lower rate of 50% with the exception of congress voting that narrowly picked 75%.

Table 3: Categorical Multi-Layer Network Tuning

Learning Rate	Proportion of Hidden Nodes	Breast Cancer	Congress Voting	Car Eval
0.001	0.500	0.282	0.291	0.367
0.001	0.750	0.282	0.291	0.365
0.010	0.500	0.281	0.291	0.363
0.010	0.750	0.281	0.291	0.363
0.100	0.500	0.042	0.063	0.174
0.100	0.750	0.015	0.064	0.161

Table 4: Regression Multi-Layer Network Tuning

Learning Rate	Proportion of Hidden Nodes	Abalone	Computer Hardware	Forest Fires
0.001	0.500	5.846	11461.734	1.899
0.001	0.750	7.318	23269.353	1.824
0.010	0.500	6.478	15246.113	2.562
0.010	0.750	7.493	11886.464	1.994
0.100	0.500		55850.509	1.636
0.100	0.750		44244.973	1.935

3 Experiment Performance

Our performance will be broken into a few parts. First, we will look at overall performance of our tests. This is what our hypothesis will be testing, so we will want to spend the most time analyzing there. Next, we will show the breakdown of each test in terms of the 5x2 validation that we were tasked with performing. Note that we sorted each data sets performance from best to worst as to show the range and volatility of each.

3.1 Model Comparison

For each data set, we wanted to provide a measuring point to see how well each performed relative to their expected classes. We chose to use the null model as it would be an example of an easy prediction technique that chooses the average of our results for regressions and the most common of our results for categorical. For regression, we took this average and found the mean squared error relative to the full data set when compared to the actual result. For categorical, we used cross entropy loss where we took the log of the probability of the most common class being the correct class in any given circumstance. This probability is calculated as number of occurrences divided by the total size of the data set. The null models, while not a perfect comparison, will hopefully provide a clearer picture into the effectiveness of our algorithms on each data set individually.

In the case of categorical results in table 5, we did not see the improvement in the multi-layer model as we had expected. Only car evaluation showed significant difference, and the congress voting set actually showed a decrease in performance. This may prove that the less defined categorical values can benefit from a few hidden layers of activation functions while the more defined class structures in breast cancer and congress voting did not need the help relative to the simple regression model. We will discuss the auto-encoder in a later section more thoroughly, but it did not seem to perform as well as we would hope in car evaluations.

Table 5: Categorical Model Comparison

Data Set	Null Model	Simple	Multi-Layer	Multi-Layer with Auto-encoder
Breast Cancer	0.187	0.040	0.038	0.039
Congress Voting	0.211	0.050	0.055	0.074
Car Eval	0.154	0.204	0.162	0.364

Within the regression models shown in table 6, the simple regression seemed to be the preferred approach with the exception of the forest fires data set that struggled to show any meaningful predictive power in the three models when comparing to the null model. We had hoped that our data sets that show exponential growth in their target outputs would benefit from this non-linear activation function, but if we look at computer hardware, it only seemed to magnify our error to a much higher degree. In terms of the auto-encoder trained model, it consistently under performed compared to the other two algorithms.

Table 6: Regression Model Comparison

Data Set	Null Model	Simple	Multi-Layer	Multi-Layer with Auto-encoder
Abalone	10.392	5.630	6.705	10.351
Computer Hardware	25742.761	6881.610	19698.299	22626.502
Forest Fires	1.951	2.014	1.970	2.021

3.2 Simple Regression

Referencing table 7, we see that our categorical data sets had a steady success rate with our simple regression values. It is interesting to see a wider range of outcomes for congress voting although, as we noted in table 5, it maintained a performance level well above the null model. Car Evaluation on the other hand never outperformed its null model in any test run.

Table 7: Categorical Simple Network Results

Test ID	Breast Cancer	Congress Voting	Car Eval
1	0.032	0.031	0.188
2	0.033	0.038	0.196
3	0.035	0.041	0.198
4	0.038	0.050	0.201
5	0.040	0.051	0.201
6	0.040	0.051	0.204
7	0.041	0.053	0.209
8	0.045	0.055	0.211
9	0.046	0.062	0.212
10	0.048	0.068	0.224

Moving on to table 8, we also had some decent success with the regression test sets. Abalone and Computer Hardware had a tight range that maintained levels below the null model error rate. Forest Fires had a few test runs that beat the null, but as shown in table 5, it under performed in the aggregate.

Table 8: Regression Simple Network Results

Test ID	Abalone	Computer Hardware	Forest Fires
1	4.947	3973.648	1.754
2	5.250	4537.550	1.813
3	5.333	5785.294	1.833
4	5.378	5864.149	2.041
5	5.485	6109.796	2.061
6	5.677	6359.265	2.063
7	5.809	6426.432	2.084
8	5.812	8380.737	2.111
9	6.065	9362.893	2.142
10	6.549	12081.058	2.234

3.3 Multi-Layer Back-propagation

The headline of our multi-layer back-propagation in table 9 is the strong improvement of the car evaluation data set relative to its simple regression model. Unfortunately, this did not bring its average below the null model, but its improvement is promising. Further tweaking to our hyper-parameters may take us even lower.

Table 9: Categorical Multi-Layer Network Results

Test ID	Breast Cancer	Congress Voting	Car Eval
1	0.026	0.044	0.114
2	0.029	0.045	0.131
3	0.031	0.045	0.141
4	0.034	0.048	0.142
5	0.039	0.053	0.145
6	0.039	0.056	0.169
7	0.040	0.057	0.171
8	0.043	0.062	0.179
9	0.047	0.069	0.181
10	0.048	0.074	0.249

The most interesting aspect of the regression test sets in table 10 when we introduced more layers to our network is the increased variance of our computer hardware results. We found a lower low and a much higher high in our range which left us with a much higher average in aggregate. This was the opposite of our expected outcome.

Table 10: Regression Multi-Layer Network Results

Test ID	Abalone	Computer Hardware	Forest Fires
1	6.208	3351.482	1.630
2	6.249	4968.871	1.749
3	6.440	5129.655	1.881
4	6.562	5245.236	1.948
5	6.636	11806.397	1.958
6	6.665	22591.508	1.975
7	6.736	29053.971	1.999
8	7.021	32615.180	2.159
9	7.139	38700.245	2.198
10	7.390	43393.052	2.202

3.4 Pre-Trained with Auto-encoder

For our auto-encoder, we had varied success with replicating our inputs with our first hidden layer activation function. It also was interesting that a successful replication did not guarantee a successful prediction at the output layer when we appended our other hidden node and class layers back on. For categorical values, we can compare table 11 to table 12 to see that breast cancer and congress voting both had mostly solid success rates in replicating inputs and predicting outputs. However, the car evaluation data set struggled with both outcomes. Given that it had increased success with multiple layers relative to the simple model, we may need to spend more time tuning the inputs to match this specific scenario.

Table 11: Categorical Auto-encoding Results

Test ID	Breast Cancer	Congress Voting	Car Eval
1	0.010	0.022	0.974
2	0.012	0.023	0.980
3	0.018	0.025	0.981
4	0.030	0.025	0.987
5	0.030	0.026	0.995
6	0.032	0.028	0.996
7	0.033	0.029	1.010
8	0.034	0.032	1.015
9	0.034	0.039	1.017
10	0.041	0.053	1.028

Table 12: Categorical Auto-encoder Pre-Trained Network Results

Test ID	Breast Cancer	Congress Voting	Car Eval
1	0.026	0.047	0.362
2	0.028	0.054	0.363
3	0.030	0.055	0.363
4	0.036	0.056	0.364
5	0.039	0.058	0.364
6	0.040	0.061	0.365
7	0.044	0.063	0.365
8	0.044	0.070	0.365
9	0.054	0.073	0.365
10	0.055	0.294	0.366

When we shift our attention over to the regression sets in table 13 and 14, we seemed to have good success in replicating our inputs with the auto-encoder in both the abalone and computer hardware test sets. However, both test sets ended up with their worst performance of the three models as we saw in table 5. Forest fires did not have as much success in replicating its inputs, but its performance stayed relatively flat relative to what we have seen throughout all of our tests on this data set.

Table 13: Regression Auto-encoding Results

Test ID	Abalone	ComputerHardware	ForestFires
1	0.081	0.032	0.295
2	0.081	0.032	0.296
3	0.082	0.037	0.305
4	0.082	0.041	0.305
5	0.083	0.042	0.311
6	0.084	0.043	0.315
7	0.084	0.045	0.316
8	0.084	0.062	0.322
9	0.084	0.097	0.327
10	0.086	0.120	0.330

Table 14: Regression Auto-encoder Pre-Trained Network Results

Test ID	Abalone	Computer Hardware	Forest Fires
1	9.600	11878.844	1.945
2	9.823	15183.322	1.948
3	10.156	17944.710	1.960
4	10.250	19580.520	1.986
5	10.267	20445.523	2.001
6	10.369	21362.767	2.049
7	10.524	24429.387	2.058
8	10.623	25022.741	2.066
9	10.915	29979.329	2.091
10	11.000	42979.781	2.103

4 Conclusion

As we return our attention to our hypotheses, we will see that our predictive power was no better than some of the models that we generated. We were correct in thinking that we would not see much improvement in the multi-layer model relative to the simple when looking at our breast cancer and congress voting data sets. Their simple regression performed about as well as any of the layered networks did for both, but that improvement with multiple layers was closer to none than it was to relevant. Car evaluation ended up being the one model that showed significant improvement when we added the two hidden layers, but its improvement ended there as the auto-encoded multi-layer model stalled out at a cross entropy loss well above the null model. We had also hoped that the multi-layer model would improve the prediction power of computer hardware and forest fires to help adjust the linear model for the exponential growth of their outputs, but computer hardware's error increased dramatically and forest fire's error did not show much movement in any network adjustment. Lastly, the Abalone test set had its best performance with the simple model. The other multi-layer models did not live up to this success with the auto-encoder trained model pushing closer to the null model success rate. Thus, our hypotheses of multiple layers improving our outputs was mostly wrong with the original two hidden layers and the auto-encoded model.