Neural Networks for regression to predict housing prices

Implementing neural networks (3 and 5 layered) for regression to discuss accuracy over one another.

Kartheek Surampudi
Machine Learning Intern
AI Technology and Systems
kartheek2000mike@gmail.com
www.ai-techsystems.com

***Abstract*—Neural Networks are argued to be more effective for traditional machine learning operations like regression and classification in picking up long range patterns. This paper attempts to predict housing prices with different types of neural networks.**

Keywords—Regression, Deep Learning, Dropout, Data Cleaning

I. INTRODUCTION

Performance of a neural network model is governed by a lot of hyper parameters, one of which is number of hidden layers. This paper attempts to predict the sale price of housing with around 80 columns by regression using a three layered and five layered neural networks to evaluate the results. The 'score' [1] metric from Kaggle [2] for the corresponding dataset can be considered the accuracy metric to evaluate the models. The dataset was processed and trained on a three layered neural network until an optimal fit is achieved and then the process is repeated for the five layered neural network.

II. DATA

The dataset is the part of a Kaggle Contest [3] .The data was observed via visualization techniques and the required data cleaning techniques were implemented. The columns such as 'Id', which had almost no correlation with the target variable i.e. Sale Price, were dropped.

Columns with categorical data were encoded using label encoding. For example a columns contains values as dog, cat, mouse, dog and so on, the encoded result would look something like 0, 1, 2, 0. A label encoding technique was preferred to the traditional one-hot encoding to keep the dimensionality of the model low as a lot of the columns of the dataset are categorical.

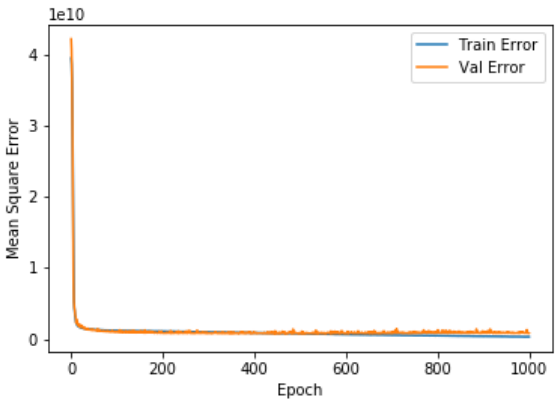
The dataset unlike any other contained a lot of null values. The null values in the categorical columns were imputed with zero as a result of label encoding. The null values in non-categorical data were replaced by the median values of the non-null part of the column. There was an exception for one column named *GrgYrBlt* which represented the year in which the garage was built. The null values in this column were replaced by the minimum value among the non-null values of this column as an intuition that the origination of null value in the column was due to the loss of records.

The encoded data was normalized using a min-max scaler.

$$X_{sc} = (X - X_{min}) / (X_{max} - X_{min})$$

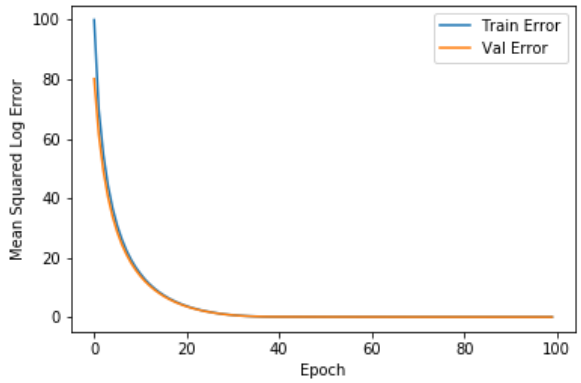
III. BUILDING MODEL AND TRAINING

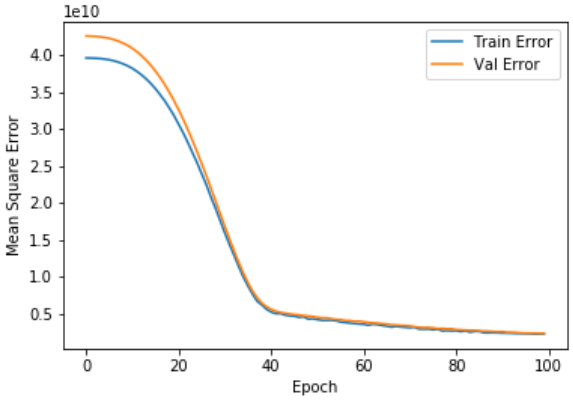
Model refers to the neural network itself. Keras [4], a high level library was used to construct and train the models. The total count of columns after cleaning the data was 79. The number of hidden units in each hidden layer, which is another hyper parameter, was fixed to 128. The ReLU [5] function was picked as the activation function for the weights. The gradient descent optimizer was chose to be RMSProp [6] with a learning rate of 0.01.The loss function was set to Mean Square Logarithmic Error. The output layer consists of one unit which corresponds to the predicted sale price of the house and has a linear activation function. This model was trained on 80% of the training data for a 1000 epochs. The model was tending to show signs of classic overfitting [7] .To avoid this Early Stopping [8] technique was implemented. The model was used to predict the target on the test dataset and results were not the best. This primitive model was given a score of 0.72858 on Kaggle.



The weight distribution of the model was observed and the activation function was changed to Leaky

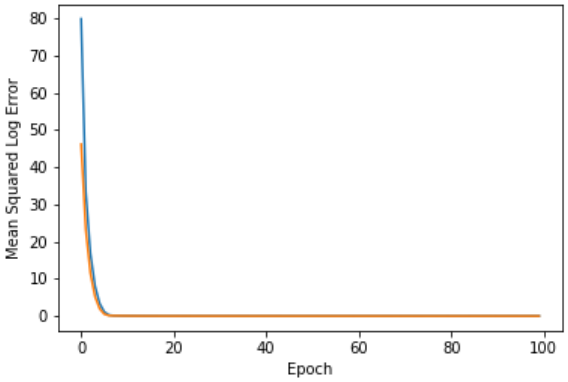
ReLU [9] as many weights had the tendency to towards zero. Regularization was also implemented via Dropout [10] with a rate of 0.2 after each hidden layer to avoid correlation between weights. The number of epochs was reduced to 100 and the learning rate was changed to 0.001.This time the model performed far better than the previous version with a Kaggle score of 0.39127.

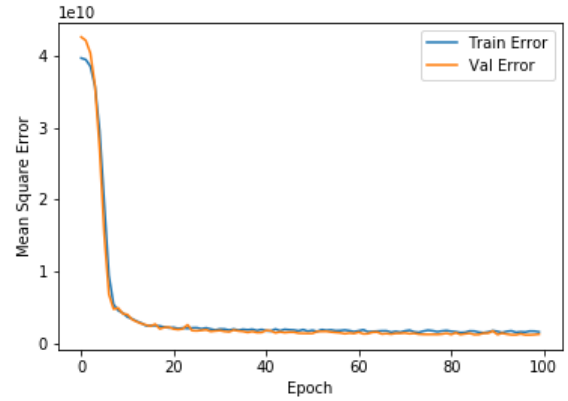




Different changes were made to the model such as increasing the hidden units, trying different optimizers such as AdaDelta [11] until the conclusion that the mentioned score was the best for a three layered neural network.

Now that the three layered network has reached its maximum potential the number of layers were upped to five with a Dropout layer of rate 0.2 after every hidden layer. As the general trend indicates the model has converged much quicker than the three layered network. Although the score has improved the validation error when visualized looked bumpy which indicated that the model was very certain of the predictions that were made even when they are wrong. The score achieved was 0.24255.





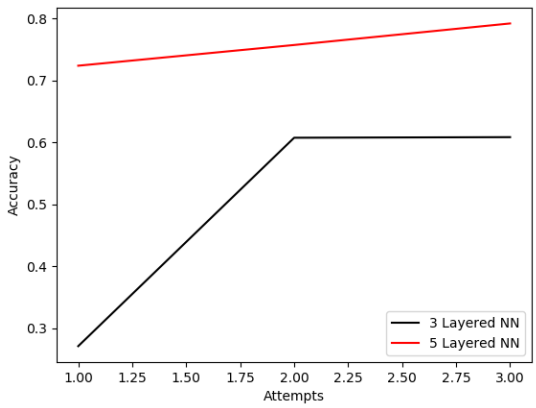
After minor changes and tweaking the hyper parameters the highest score this project could receive based on the methods used was 0.20779.

IV. COMAPRISIONS

Sheer numbers can show that the five layered neural network does a better job than the three layered neural network. The score has improved and so has the weight distribution over the network. Tuning the hyper parameters in both the networks can attributed to a lot of progress.

V. CONCLUSION

In conclusion, increasing the number of layers of a neural network can increase the accuracy of the model. The below model shows that the accuracy of the 3 layered neural network has stagnated after the second attempt while the accuracy of the five layered neural network increases gradually with the attempts.



This goes to explain that the raw data i.e. the encoded data without the selective features requires more number of neurons to pick up long range patterns.

REFERENCES

[1] Root Mean Squared Logarithmic Error.

[2] Kaggle is an online platform for data scientists (https://www.kaggle.com/)

[3] Hosing Prices: Advanced Regression Techniques (https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data)

[4] https://keras.io/

[5] Abien Fred M. Agarap, "Deep Learning using Rectified Linear Units (ReLU)" .

[6] https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[7] Shaeke Salman, Xiuwen Liu, "Overfitting Mechanism and Avoidance in Deep Neural Networks:.

[8] Rich Caruana et al. , "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping".

[9] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li, "Empirical Evaluation of Rectified Activations in Convolutional Network".

[10] Nitish Srivastava et al. , "Dropout: A Simple Way to Prevent Neural Networks from Overfitting".

[11] Matthew D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method".