ECE 9063 Data Analytics Foundations

Assignment 2: Neural Networks

Student Name: Jianping Ye
Student Number: 250887769
Instructor: Katarina Grolinger

Problem Statement

The used car market is a perfect place for finding cars in decent conditions and with fair prices. The market demand has been growing in recent years. It will be beneficial for both buyers and sellers if there is a model to predict prices based on characteristics of the car. With the help of a suitable model, buyers will be able to make sure the car is worthy of its price, and sellers can get a more accurate price estimation in accordance with other cars having similar conditions. In this report, the forecasting problem is defined as follow: predict the price of a used car in the current year given a set of properties.

Dataset Description

Link to the data: https://www.kaggle.com/adityadesai13/used-car-dataset-ford-and-mercedes

These datasets list scraped data of used cars in the British market and are separated into files specific for each car manufacturer. In this report, the dataset selected is “Audi.csv”. It contains 9 attributes and 10668 samples. The dataset is suitable for this assignment as it has adequate attributes and samples. With over 10,000 samples, it is easier to trade-off between computational time and model reliability . The attributes are listed below:

- Model: car’s model code
- Year: registration year
- Price: price on market
- Transmission: type of gearbox
- Mileage: total miles travelled
- fuelType: type of fuel
- tax: road tax
- mpg: fuel consumption in miles per gallon
- engineSize: size of engine in litres

All the attributes in the dataset except “price” are considered features as they are important factors determining the price – the target variable – on the market.

Neural Network Architecture Overview

There are several potential network architectures to be considered, including Multi-Layer Perceptron (MLP), Convolutional Neutral Network (CNN), and Recurrent Neural Network (RNN) or its variants.

MLP is under the category of feedforward neural network. It is built by stacking multiple fully connected layers of perceptron. With this type of architecture, complex target functions can be effectively approximated by adding layers and neurons. By progressing to deeper layers, data is transformed, and complex features are learned. As in multi-features regression tasks, multiple independent variables contribute to the dependent variable at each calculation step simultaneously. This process can be simulated by fully connected layers.

CNN is primarily used with image data as it can capture spatial relationship of nearby pixel points by applying convolution filters across the image matrix. Convolution is achieved by calculating a dot product between the convolution kernel and sub-regions of image matrix which of the same size as the kernel. Convolution allows higher-level features to be extracted and effectively reduces the original data into a more compressed form without losing much critical information. A CNN architecture involves convolution layers, pooling layers which is placed between successive convolutional layers to serve as down sampling, and fully connected layers. Noticeably, neurons are partially connected in convolution layers.

RNN is commonly chosen to model time-series data as it can simulate time dimension by creating recurrent feedback loop to learn from sequential and temporal behaviors. In other words, it combines outputs from previous layers along with hidden states as the new input. RNN takes historical information into account and it is capable of processing varying length of input. However, computational time and training can be very difficult for RNN partly due to increased connection complexity.

Overall, MLP is selected to model our task. Firstly, each training sample in our dataset, namely each sale of used cars, is assumed to be independent and identically distributed. Each

sale is considered to has negligible relationship with previous sales. Thus, there is no time aspect to be modeled and RNN or its variants may not be a good choice at this scenario. Also, spatial relationship does not exist as with image data and make no use of advantages of CNN.

The MLP network comprises of one input layer, several hidden layers, and one output layer. The number of neurons in the input layer equals the number of features. The number of neurons in the hidden layers will be a hyper-parameter to be determined. The output layer is simply one neuron as the objective is to output a predicted value. All the layers except the output layer are fully connected (Dense), that is, every neuron in a layer is connected to every neuron in the following layer. Table 1 shows the complete list of hyper-parameters that are tuned.

Table 1: Full List of Hyper-Parameters

Hyper-parameters	**Parameters Distributions**
Number of hidden layers	5 ~ 8
Number of neurons	20 ~ 100
Learning rate	0.01, 0.001, 0.002
Weight initialization strategy	He normal, Random normal, Glorot normal
Activation function	RELU, ELU

Development Procedures

1. Exploratory Data Analysis

The data is examined to check whether there is invalid or missing entry. It turns out that the dataset is well-prepared and little cleansing is needed.

A heap map is generated from the correlation matrix to better visualize linear correlations between each attribute. By looking at the second column, we can see features all have a strong positive or negative correlation with the label.

2. Preprocessing

One-hot encoder is used to encode categorical attributes. Then, separate the dataset into features and target. The “train-validate-test” strategy is used to split data for different uses, that is, 60% for training, 20% for validation, and 20% for test. The purpose of validation set is to evaluate model loss at every epoch during training and hyper-parameters search.

Finally, we need to perform standardization on the data to speed up convergence of gradient descent algorithm. Noticeably, the standard scaler should only be fitted to the training set, then the fitted scaler is applied on validation set and test set.

3. Modeling

Grid search is a hyper-parameter estimation approach that exhaustively tries each set of hyper-parameters combination to build and evaluate the model, and ultimately find an optimal estimator which yields the lowest loss in the validation set. Due to computational power and time constraints, the grid search approach can only be applied on a limited range of hyper-parameters at this assignment. Therefore, a randomized search is also attempted on the full list of hyper-parameters shown in Table 1. The two best models obtained from the two approaches will be compared to select the most suitable one in the result comparison section.

In contrast to grid search, randomized search samples random combinations of hyper-parameters from the specified parameter distributions. The number of combinations sampled can be defined, allowing one to have explicit control of hyper-parameters attempts. This characteristic dramatically reduces tuning time. Its results might be slightly worse, but it is a better fit than grid search if one would like to search in a much larger parameter space. Early stopping is also used to halt training when the selected loss metric has stopped improving for a certain number of epochs or when the model starting to diverge.

4. Performance Metrics

To quantitatively measure the performance of the neural network, Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) will be used as error metrics. MAE measures the average magnitude of residuals between predicted and actual values. One advantage of MAE is that it is less sensitive to outliers. RMSE penalizes larger errors as the square value is much larger than the absolute value and it tends to neglect small errors. Generally, RMSE is preferred for regression tasks given that outliers are rarely present.

Both metrics will be applied on training data first. Cross validation is performed during hyper-parameters tuning. Finally, the best model is evaluated on test data to estimate its generalization errors.

The network's performance will also be compared with results obtained from previous assignment in which approaches besides neural network were taken. Also, the impact of hyper-parameters tuning will also be assessed.

Results Comparison

The following graphs depict two best models' loss measured in mean squared error versus epochs during randomized hyper-parameters search and grid search.

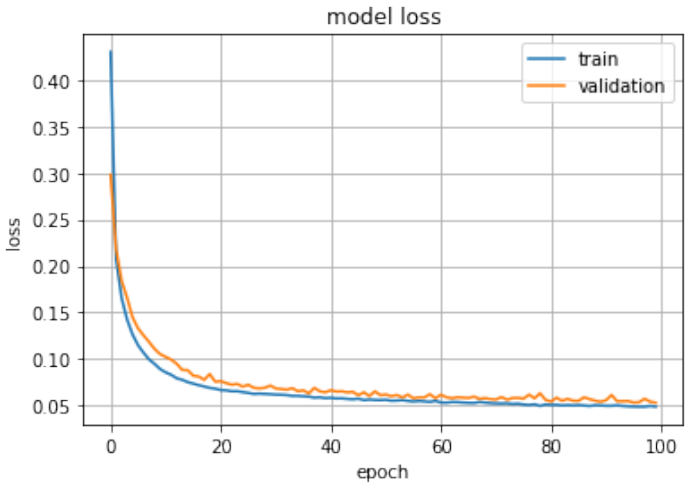


Figure 1: Best Estimator in Randomized Search

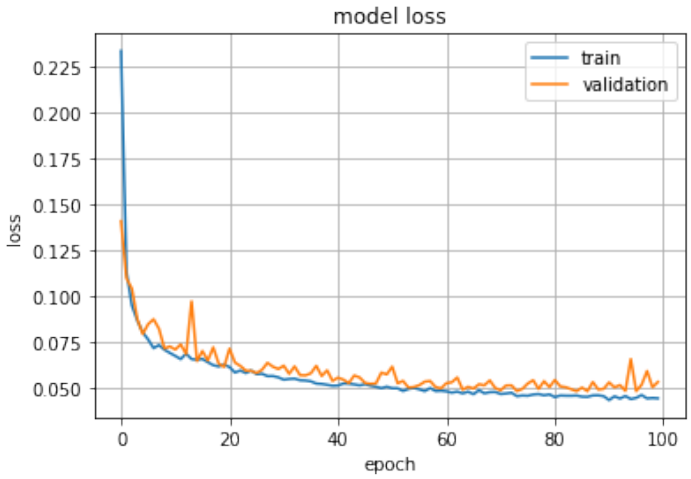
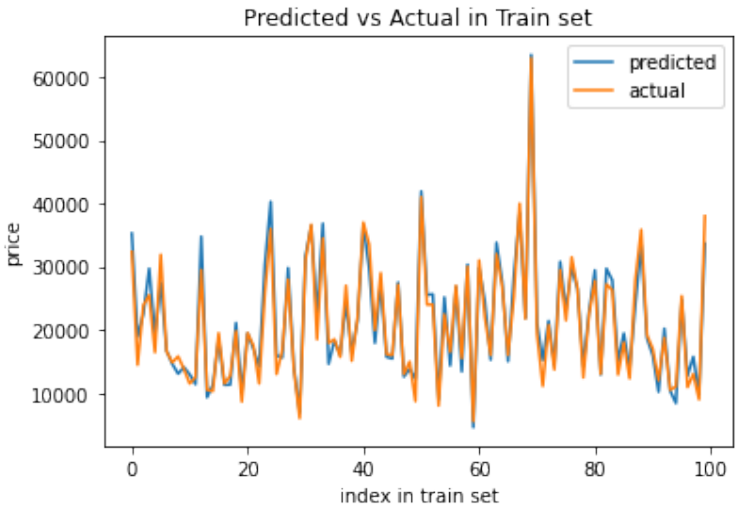


Figure 2: Best Estimator in Grid Search

In Figure 1, it is observed that the training loss and validation loss steadily decrease as epochs progress. Better still, the validation curve sticks close to and remain higher than the training curve, which indicates that there is very little overfitting occurs. The validation performance is smooth and stable.

In Figure 2, the validation loss oscillates more frequently than the previous model. This model might not generalize well as the first one. Its test RMSE is slightly better but only better off for a negligible margin. Hence, the model found in randomized search is selected.

The following graphs plot predicted values versus actual values in train and test set, respectively.



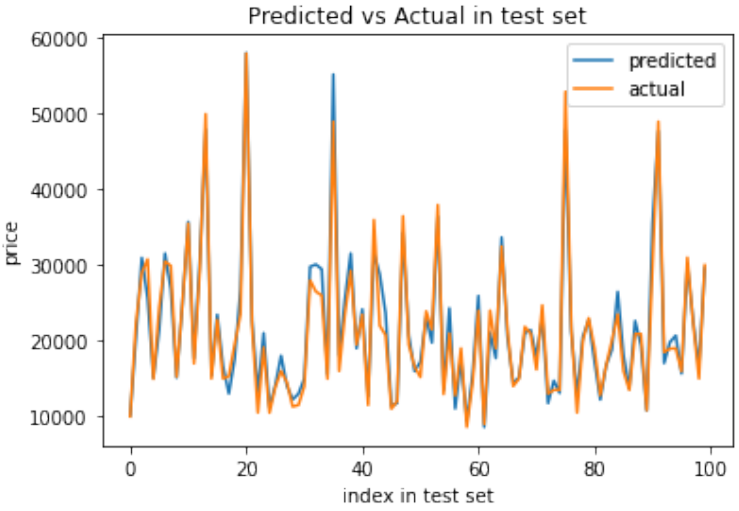


Table 2: Performance Measure

Model	Train MAE	Train RMSE	Test MAE	Test RMSE
Decision Tree	48	320	1940	3056
SVR	3167	6085	3420	6690
Random Forest	582	918	1527	2305
Tuned MLP	1715	2494	1843	2701
Untuned MLP	1836	2716	1987	2987

As shown in Table 2, even thought the tuned neural network does not beat the performance of random forest regression, it still outperforms decision tree regression and support vector regression. This result is considered acceptable given the fact that random forest regression utilizes ensemble learning, which could be the reason behind its outstanding performance. After tuning, performances of the network slightly improved.

To sum up, neural network is more flexible than regular regression algorithms in terms of it can adapt dynamically to the data instead of being bounded to a specific regression formula. By adding more hidden layers and neurons up to a certain optimal point, its complexity and prediction power increases. Noticeably, training of neural networks can be difficult and unstable as it suffers from several problems, such as vanishing and exploding gradients.