COMP534- Applied Artificial Intelligence

Assignment 2 – Neural Network for Regression

Submitted by,

Jishnu Prakash Kunnanath Poduvattil	201581347	j.kunnanath-poduvattil@liverpool.ac.uk
Akhil Raj	201594703	a.raj@liverpool.ac.uk

Introduction

This report consists of discussion and analysis of solving a regression problem using neural networks using House sales data in King County, USA.

Libraries Used

We mainly used the following libraries for modelling and Exploratory data analysis (EDA): -

Library	Utilisation	Link
Scikit-learn	Data pre-processing, model performance analysis, Hyperparameter tuning etc	https://scikit-learn.org/stable/
Tensorflow	Building Sequential model using dense layers	https://www.tensorflow.org/
Keras	KerasRegressor for hyperparameter tuning	https://keras.io/
Pandas	Load data	https://pandas.pydata.org/
Numpy	Numerical operations	https://numpy.org/
Matplotlib	Plotting diagrams, charts for EDA	https://matplotlib.org/
Seaborn	Plotting pair plots, heatmap etc for EDA	https://seaborn.pydata.org/

Strategies for Hyperparameter tuning

I) Manual Hyperparameter tuning – (manualTuning.ipynp)

Each hyperparameter was tuned individually on a base model to study its influence in training. A base model was setup with minimal architecture and re-iterated with different values of one hyperparameter, where others remained constant. After training, a best value for hyperparameter is selected. This is used to tune the next hyperparameter.

II) RandomizedSearchCV (randomTuning.ipynp)

RandomizedSearchCV method from sci-kit learn is used for tuning the hyperparameters. A set of values for tuning a hyperparameter with a base model is passed in to RandomSearchCV algorithm. The algorithm finds the best model by training given values of one hyperparameter with 5-fold cross-validation and repeating the process for all other hyperparameter.

Training and Testing Process

Data Pre-processing

A detailed exploratory data analysis was done on the data to visualise linear relations between the features, find the outliers and clean the dataset. Please refer **EDA_kcHouse.ipynb** for detailed procedure on EDA, computing new features and removing outliers.

New Features: Added new features **age** (sold year - build year) and **reno_age** (Sold Year - Renovated year) and **reno_flag** (1 if renovated, else 0).

Standard Scaling: The input features are scaled using Standard Scaler from sci-kit learn. The target variable **Price** is an economic variable. To represent Vertical changes in prices related to feature vectors, Logarithmic scale is used. So **converted price to log scale.**

Hyperparameter Values

Hyperparameter	Manual Tuning	Random Tuning
Learning Rate	[0.0001, 0.001, 0.01, 0.1, 0.5, 1]	[0.001,0.01,0.1]
Batch Size	[16,32,48,64,80,96,112]	[16,32,64]
Number of epochs	[10, 20, 30, 40, 50]	[10,20,30,40]
Activation Functions	['relu', 'tanh', 'elu', 'sigmoid', 'softmax', 'selu', 'linear']	['relu','sigmoid','softmax','softplus','softsign','tanh','selu','elu']
Optimizers	['sgd','rmsprop','adam','adagrad', 'nadam']	['sgd','adadelta','adagrad','adam','adamax','ftrl','nadam','rmsprop']
Loss Function	['mse', 'mae', 'mape', 'msle', 'cos']	['mean_squared_error','mean_absolute_error', 'mean_absolute_percentage_error','mean_squared_logarithmic_error','log_cosh','huber']
No. of Hidden Layers	[1,2,3,4,5,6]	[1,2,3,4]
No. of nodes in hidden layers	[8, 16, 24, 32, 40, 48, 54]	[24,32,40,48]

The specified values of hyperparameters were chosen by following the conventional methods and from keras documentation.

Train-test split

Train-test split ratio was defined as 80:20. In manual tuning, a validation split of 0.2 is also used. In random tuning, 5-fold cross validation is done.

Tuning Hyperparameters

The impact of hyperparameter is evaluated by computing R squared value. With R squared value, we can say the goodness of fit of predicted and true values. Also, root mean square error is printed to console to monitor the performance of different models. In random tuning, R squared is used as score metrics in cross validation.

Evaluation

Impact of Hyperparameters

We are going to present our results after performing hyperparameter tuning in two methods. Manual tuning and Random tuning, which is explained in the strategies section above. Please note that all reference regarding the manual tuning is in manualTuning.ipynb file and random tuning in randomTuning.ipynb file.

1) **Number of Hidden Layers**

Manual Tuning (Refer Section 4.1): -

- In the training and validation plots, a slight instability can be observed even though the model is guarded by Early stopping to avoid overfitting.

- In R squared value v/s number of layers plot the more the layers are the score improves. But the score is going down after 4th layer.
- Here, 4 layers gives the best performance. But when moving forward there was no improvement in the R squared value. So reverted and tested for the next best values and found out that 2 layers gives the best performance in next tuning stages.

Random Tuning (Refer Section 4.1): -

- Best performance was observed on the model with 2 layers.

2) Number of Hidden Nodes

Manual Tuning (Refer Section 4.2): -

- In the training and validation plots, a slight instability can be observed even though the model is guarded by Early stopping to avoid overfitting.
- In R squared value v/s number of layers plot the more the nodes are the score decreases.
- Best performance was obtained for the simplest model architecture.

Random Tuning (Refer Section 4.2): -

- 24 nodes per hidden layer gives the best performance.

3) Optimizers

Manual Tuning (Refer Section 4.3): -

- In the training and validation plots, a slight instability can be observed.
- An interesting observation is that ADAM, SGD, NADAM and RMSPROP gives similar performance.
- So, a combination of all these optimizers were tried for tuning the next one, and selected ADAM based on intuition.

Random Tuning (Refer Section 4.3): -

- 'ftrl' is selected by the algorithm which have the best performance.

4) Learning Rate

Manual Tuning (Refer Section 4.4): -

- In the training and validation plots, a slight instability can be observed for learning rates greater than 0.001 and worse performance for learning rates 0.5 and 1.
- The model convergence was too quick and highly overfitting was observed for 0.5 and 1.
- Stable and optimal convergence was observed in 0.001 learning rate.

Random Tuning (Refer Section 4.4): -

- 0.01 is the best performing in [0.001,0.01,0.1]

5) Activation functions

Manual Tuning (Refer Section 4.5): -

- We can observe that tanh, sigmoid and SoftMax activations are not suitable for this task and training is stopped by Early stopping.
- Stable and optimal convergence was observed for RELU activation.
- We get a similar performance for SELU, ELU and LINEAR which are also good fit for regression tasks.

Random Tuning (Refer Section 4.5): -

- Best performing model has 'relu' activation function

6) Loss

Manual Tuning (Refer Section 4.6): -

- Here, all the loss functions are performing well. We can observe a similar training and validation losses and convergence in all the loss functions.
- Mean squared Error was the best performing with the model which yielded in highes R squared value.

Random Tuning (Refer Section 4.6): -

- 'huber' function gives the best performance.

7) Batch Size

Manual Tuning (Refer Section 4.7): -

- By convention, the smaller the batch size is, the higher the performance could achieve. But it has a high chance of overfitting.
- Here, batch size of 16 gave the best performance and fast convergence.

Random Tuning (Refer Section 4.7): -

- R squared value is high when batch size is 16

8) Epochs

Manual Tuning (Refer Section 4.8): -

- By convention, the higher the number of epochs is, the higher the performance could achieved. But getting an overfitted model is highly possible.
- Here, all the epochs yielded good R squared scores, but 30 was giving the best score.

Random Tuning (Refer Section 4.8): -

- The model is best performing at epochs 40

Final Models

After hyperparameter tuning, best models were trained with the following hyperparameters.

Hyperparameters	Model 1	Model 2
Learning Rate	0.001	0.1
Batch Size	8	16
Number of epochs	50	40
Activation Functions	RELU	RELU
Optimizers	ADAM	FTRL
Loss Function	MSE	HUBER
Number of Hidden Layers	2	2
Number of nodes in hidden layers	[8,8]	[24,24]

The model performance is given below,

Model	MAE	MSE	RMSE	R squared value
Model 1	0.23	0.085	0.29	0.59
Model 2	0.22	0.077	0.28	0.49

Conclusion on Final Models

The final models are of simple architectures giving their best results. The dataset is of house prices and the task is to predict house prices. Linear regression models are the perfect fit for these types of datasets. We are trying to build a neural network for regression to learn the process of

hyperparameter tuning. Finding the best network is a rigorous task of trial and errors using different combinations of model parameters.

The final model is performing better than all the other trained models is because of its ability to learn and converge faster. That is achieved by using the right hyperparameters. For example, using relu activation function rather than sigmoid, is the right choice because both are designed in different ways and for different purposes. So, finding the right set of hyperparameters is the key factor to improve the performance of the model and that is what we did by tuning the hyperparameters and building the final model with the best set of values.

Final Conclusions

Challenges

The challenges of the project are: -

- Batch normalisation was implemented in the base model as optional variable but using batch normalisation was not yielding better results.
- Similarly, dropout was also tried out in the final model, but did not yield better results.
- More complex architectures could result in better results, but that makes the model too complex which is not economical.
- Overfitting and instable training, even though overfitting is prevented by Early stopping method.
- Outliers in the data discovered in exploratory data analysis.
- Computational demand – For tuning the hyperparameters, it is a time consuming and computational power demanding task.
- We cannot explore all possible combinations of hyperparameter values
- One drawback of random search is that the suggested set of values should not be necessarily the best values.

Task Allocation

Jishnu Prakash Kunnanath Poduvattil

- Project Repository in GitHub: For keeping track and maintaining code
- manualTuning.ipynb
    - Read the data
    - Pre-processing the data for training
        - Cleaning and Standardising
    - Hyperparameter tuning and find best model
- Report.pdf
    - Prepared report and compared the results of each method
- EDA_kcHouse.ipynb
  A detailed exploratory data analysis was done for better understanding of the dataset. Relationships between the feature vectors were visualised and explained. Outliers were identified and removed.
- Readme.txt
  Consists of file information and how to run the script to reproduce the results.

Akhil Raj

- randomTuning.ipynb
    - Read the data
    - Pre-processing the data for training
        - Cleaning and Standardising
    - Hyperparameter tuning using cross validation, RandomizedSearchCV and find best model
    -
- Report.pdf
    - Documentation