Melbourne housing market: Price modelling

ETC3555: Report

Jack Cameron, Eileen Dzhumasheva, Huize Zhang

14/10/2019

Introduction

In this project, we have been given a data set of all sold houses from March 2016 to December 2018 in Melbourne, detailing the sale price and a number of characteristics of the house. Our goal is to use this information, to predict the sale price of a house.

Section 1: Model

The Learning Problem

Y - Regression: Price (dollar amount) X - Input:

This section presents the learning problem and the models you have considered. Some of the questions you could answer include "what are the parameters/hyperparameters?", "how do you optimize these parameters?", "explain how these parameters control the complexity/flexibility of the algorithm", "does your method scale well with large data sets?", etc.

Section 2: Experiment

The dataset contains information on 63,013 house sales scraped from publicly available results posted every week from Domain.com.au. The dataset includes variables that detail the suburb, address, rooms, type of real estate, price, method of selling, seller, date, postcode, region, propertycount, distance from the CBD and council area. The dependent continuous variable, y, defines the sale price of the house in AUD. Further details about the data set as well as some preliminary analysis are included in Appendix.

We first divide the house price into 10 categories: (0, 100k], (100k, 300k], (300k, 500k] ... (1500k, 1700k], (1700k, ∞) and this allows us to perform neural network on a multi-class classification problem, which is much easier than prediction.

Kauko[reference] has provided an overview on current neural network application on predicting house price and introduced some advanced methods like self-organising map(SOM) and learning vector quantization (LVQ). However, there is no pre-trained architecture for house price forecasting and thus, we build our own two layer deep neural network from scratch. Based on the size of our data (48k observations with 277 features) and the project goal (classifying house price into 10 categories), it is not sensible to construct a neural network with more than 4 layers. Although Neural network with 3 layers may also work, adding an additional means more parameter tuning and computing, which appears to be too complex for my computer lo.

The data is first scaled and centred before sending into the neural network for training and normalisation is performed before each dense layer. This normalisation is necessary because it helps to speed up the converging process by normalising the extreme large value outputted from the previous layer.

The activation function for the middle layers are relu function. This choice is made based on two considerations.

1) relu function produces zero output for negative input, this would be helpful to avoid overfitting because a proportion of the features will be set to zeros.

2) unlike tanh function that will saturate at asymptotic, which means the weight will not be updated much when the input neural has a large value, on the other hand, Relu function will will increase to infinity. This property of relu allows the relevant weight to be updated when the input gets larger. And the potential drawback of getting extreme large output is overcame by batch normalisation discussed before.

The activation function for the final layer is softmax because this is the appropriate function for multi-level classification.

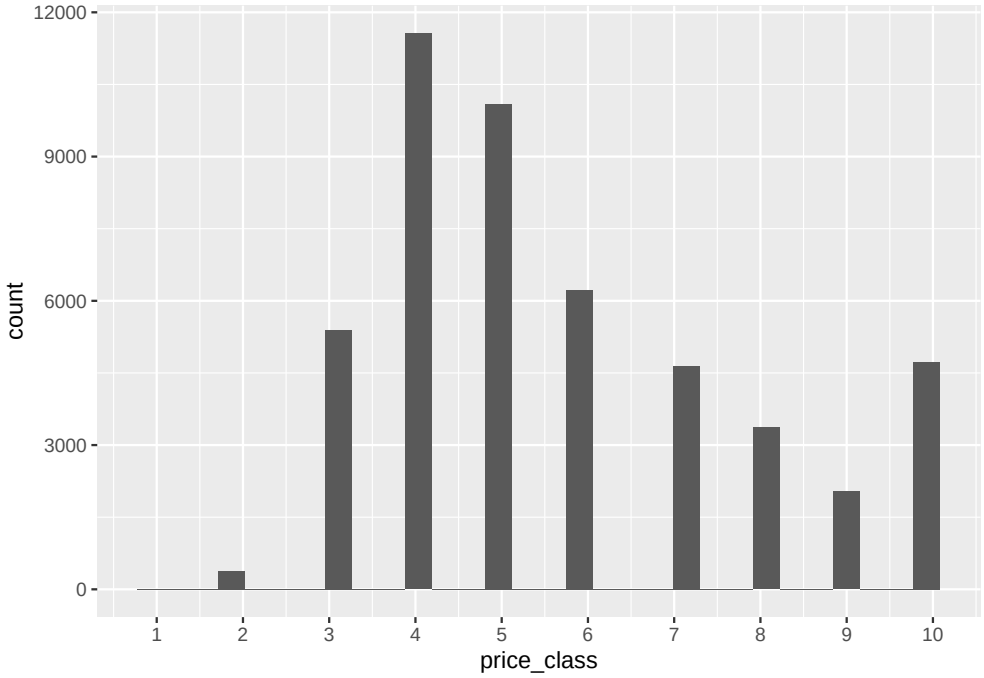
To avoid overfitting of the data, several regularisation methods are used. Dropout rate is used before each dense layer. One thing to notice is that the dropout rate is added before the batch normalisation because otherwise the normalisation will be based on the full data while part of them will be dropped out before supplied to the next layer. Regularisation is also applied in each layer with learning rate being a hyper-parameter to tune.

Early stop is also used to allow for more efficient updating on the parameter. We set epoch at a large fixed value of 500 and use `patient = 20` in the early stop in our model structure. This design allows the neural network to keep updating if needed while not wasting omputational power and time when the updates become slow.

The optimisation is done with an ADAM optimiser. Different optimiser such as RMSPROP or RGD could be used for tuning while theoratically ADAM is believed to have better performance since it incorporates the momentum from RMSPROP to RGD The learning rate for ADAM is treated as a hyperparameter to tune.

The loss function used is the categorical cross entropy, which is the only option for multi-class classification.

The metrics for evaluation is the loss function rather than the accuracy result. This design is chosen because from the data, the house price is not evenly distributed among all the categories. We can see from Figure **??**, price class 4 (ranging from (500k, 700k]) has the most number of observation falls into while very few observations belong to category 1 or 2. If accuracy measure is used for evaluating the algorithm, it will prone to predict category 4 or 5 to achieve a higher accuracy result while not learn the data well. Therefore, loss function, evaluating the performance of the algorithm based on its score on categorical cross entropy is a more reliable measure.



Section 3: Results and Discussion

This includes for example graphs and tables, as well as a discussion of the results. You should also present your model fitting, diagnostics, etc.

Section 4: Conclusion

This includes a summary of the findings.

Appendix