Kaivalya Mishra CSCI561

Homework 3 Report

Overview of Parameter Groups

This report will be dividing the explanation of the parameters by the following groups:

- Preprocessing Parameters
- Hyperparameters
- Model Parameters

Preprocessing Parameters

For preprocessing data, various different combinations of inputs were tested and rearranged before finally deciding upon this list:

```
[
    # home_type_norm,
    house, townhouse, condo, multi_family, co_op, land,
    # contingent, mobile_house, for_sale, foreclosure,
    # coming_soon, pending,

    price_norm, bath_norm, sqft_norm,

    zip_code_norm, borough_norm,
    # manhattan, staten_island, bronx, brooklyn, queens,

    # lat_norm,
    long_norm
]
```

Firstly, some of the location data was omitted due to lack of consistency and misinputs (*e.g.*, `ADMINISTRATIVE_AREA_LEVEL_2`, `LOCALITY`). Other data was omitted due to not really impacting the neural network's accuracy (*e.g.*, `BROKERTITLE`, certain values of `TYPE`). The above data is all normalized to a range between 0 and 1. For all normalizing, the min-max method was used, as represented by the given formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

While initially trying out a label encoding, the `TYPE` values seemed to be more useful when one-hot encoded as some of the values did not contribute to greater accuracy. When label encoded, the home types were arranged by descending number of likely beds (determined by comparing their respective modes, medians, means, and number of occurences).

ZIP code and borough information data was extracted from the `FORMATTED_ADDRESS` column and then label encoded. The ZIP codes were ordered based on ascending median income and the boroughs were ranked by descending population density. A one-hot encoding of the boroughs was tested, but seemed about as good as the label encoding.

For the output values, one-hot encoding was used alongside a lookup list, with the columns corresponding to indices at which the number of beds was specified.

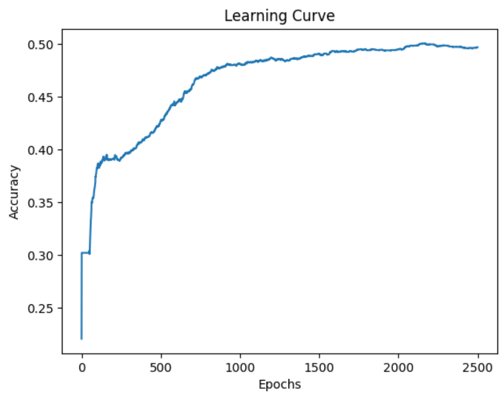
One interesting point to note, during testing, it was discovered that when using only the `LONGITUDE` values without the corresponding `LATITUDE` column seemed to yield more consistent accuracy. The reason for this behavior was unable to be determined.

Hyperparameters

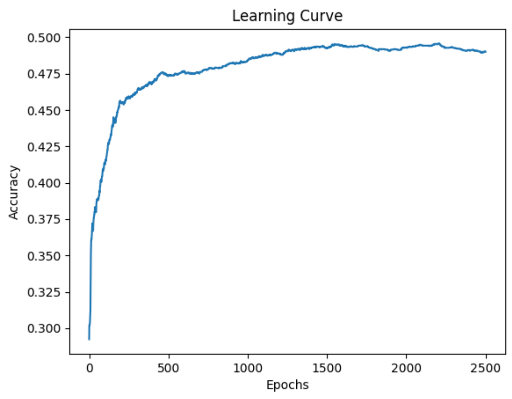
In regards to hyperparameters, the values of 2500 epochs, a 0.001 learning rate, and 64 batch size were determined to be suitable for this data and model. (For model parameters, the chosen values are explaned in the next section.)

Below are the accuracy values over the epochs on the first train test split for various different learning rates and batch sizes.

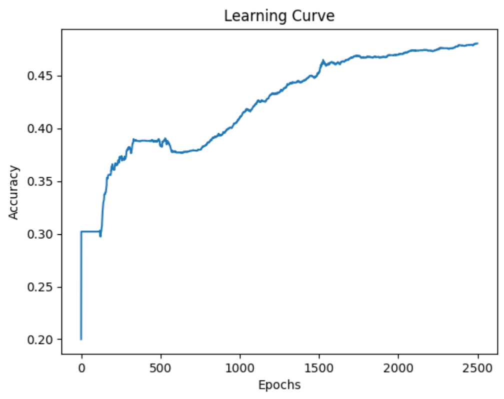
Learning Rate = 0.001, Batch Size = 64 (Base Values)



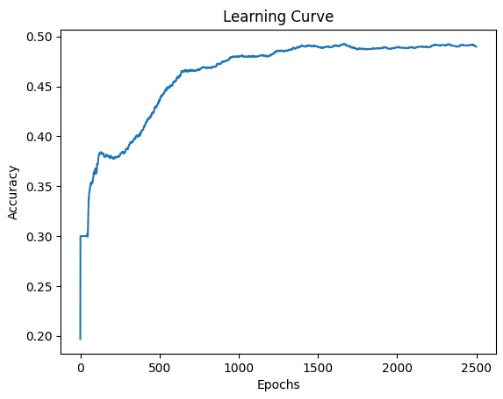
Learning Rate = 0.005, Batch Size = 64



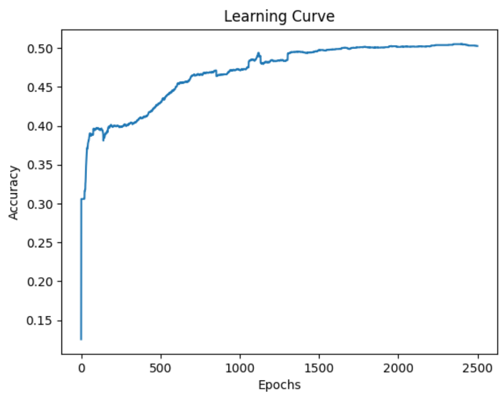
Learning Rate = 0.0005, Batch Size = 64



Learning Rate = 0.001, Batch Size = 32



Learning Rate = 0.001, Batch Size = 128



Model Parameters

For model parameters, the sigmoid activation function alongside 2 hidden layers (with 6 neurons in the first layer and 3 neurons in the second) were chosen as they returned compartively higher accuracy values consistently. The output layer utilized the softmax function, with the cross-entropy gradient being used when accounting for error.

When deciding which activation function to use, the sigmoid, ReLU, and Leaky ReLU functions were considered. ReLU on its own did not seem to be viable as it would encounter the vanishing gradient problem and output 3 beds for each data entry. Leaky ReLU seemed to work well, but did not seem to be as accurate as the sigmoid function.

Lastly, 2 hidden layers with 6 neurons in the first layer and 3 neurons in the second seemed to be most optimal. Adding additional layers did not seem to increase accuracy (in some cases accuracy decreased) and having only 1 layer was insufficient. Similar scenarios occured when changing the number of neurons in layers 1 and 2.