CSE 151B Project Final Report

Author: Liopold Chen
lic028@ucsd.edu

1 Task Description and Background

1.1 Problem A

Github link: https://github.com/Liopold35894/CSE151B-Final-Project

Deep learning is a subset of machine learning, and deep learning tasks is important because we can train machine to learn from the data and make predictions or classifications to the real world tasks. Some real world examples includes self-driving car, chatting bot, etc. With deep learning, we are able to let AI do some tasks for us already, and possibly more and better in the future.

1.2 Problem B

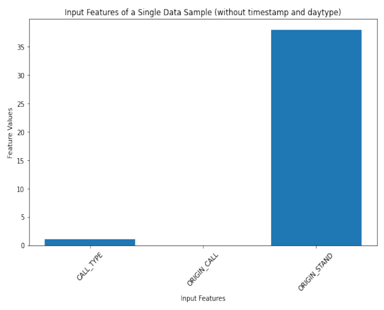
The input features (x) I used are call type, origin call, origin stand, timestamp, and day type. The output (y) represents the travel time. Formulating the predicting task as an optimization problem, the objective to to minimize the MSE loss between predicting time and the actual time. It can be use for other time series forecasting, like predicting for the stock price, whether conditions, and other travel time predictions. Since the based model is using LSTM, it can accomplish similar tasks easily.

2 Exploratory Data Analysis

2.1 Problem A

The size of training data is (1710670, 9). And the size of test data is (320, 8).

Input dimension is 9, and the output dimension is 1. 9 represents the number of input features in the data, and the output dimension of 1 represents the predicted travel time based on those input features.



Timestamp and day type have much larger value compare to these 3.

Table looks like this:
Input Features Feature Values
0 CALL TYPE 2.000000e+00
1 ORIGIN CALL 0.000000e+00
2 ORIGIN STAND 0.000000e+00

3 TIMESTAMP 1.372637e+09
4 DAY TYPE 0.000000e+00

2.2 Problem B

There is no specific validation dataset. The size of train dataset is just 1710660.

For feature engineering:
1. I remove the missing data from the train data according to missing data column.
2. I mapped the categorical features like call type and day type to numerical value for the training purposes.
3. For other missing values, I use fillna method to fill nan with 0 to ensures that model can process the data. Most of these is for model to effectively utilize the data and to prevent from any errors causing by it.

I did not normalize the data. Thinking about it, if I have used it, I will use mean normalization and standard deviation normalization for the initial process.

I did use call type as input feature for the model, but only by mapping each character to numerical representation. I have tried to use longitude and latitude provided in the metadata, but because of some unknown errors, I ended up going back using my original code.

3 Deep Learning Model

3.1 Problem A

I ended up using call type, origin call, origin stand, timestamp, day type as input features, and travel time as output feature.

The model I use is the LSTM model as it is well-suited for doing a predicting task to capture the long-term dependencies. The loss function I use is MSE, which is just a common loss function to use for the training.

3.2 Problem B

I have only used the LSTM model and improve it from there.

For the model architecture of the LSTM:
1. Input size: The input size is 5, with input features call type, origin call, origin stand, timestamp, and day type.
2. Hidden size: The hidden size is set to 64, which determine the size of hidden units.
3. The number of the layers is 3. This represent number of recurrence layers in the model.
4. Output size: This is set to 1 to represent the travel time.
5. LSTM layer takes in the parameters and runs them.

I have used the batch normalization to normalize the output of the LSTM layer along with the hidden size.

I have also applied the dropout regularization to prevent the model from over-fitting. The dropout rate is set to 0.95 (dropout 5% of the units).

4 Experiment Design and Results

4.1 Problem A

I'm using the CPU for the training and testing. The optimizer used is Stochastic Gradient Descent, With the initial learning rate set to 0.01 and momentum of 0.9. I also used a exponential decay scheduler with gamma set to 0.9 (10% of decay rate for each epoch). The way I tune the parameter is to test out different value and to the if the result is better than before.

I currently use 30 epochs for the training. Because the model takes really long time to train, so 30 epochs is most doable number. 1 epoch take approximately 20 minutes to train. The batch size is 64.

Many of these values is being adjusted base on the performance and the result of training.

4.2 Problem B

Since I have only use the LSTM model, I will be reporting for this model.

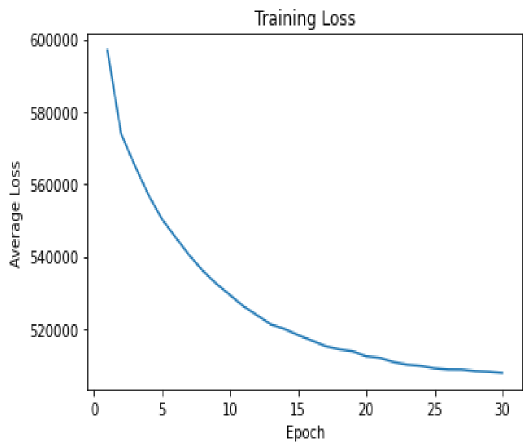
With this model, my current best score on kaggle with 30% of the data is 782.4.

The training time for the LSTM model is fairly long. That might also because I am using the whole data for the training, and I think it will be better for the result and the runtime if I only use part of the data for training. To improve for the training speed, I did apply the momentum of 0.9 in optimizer, it is used to accelerates the training process by accumulating the exponentially weighted moving average of past gradients.

Sum over the total parameters in the model, we get 83905.

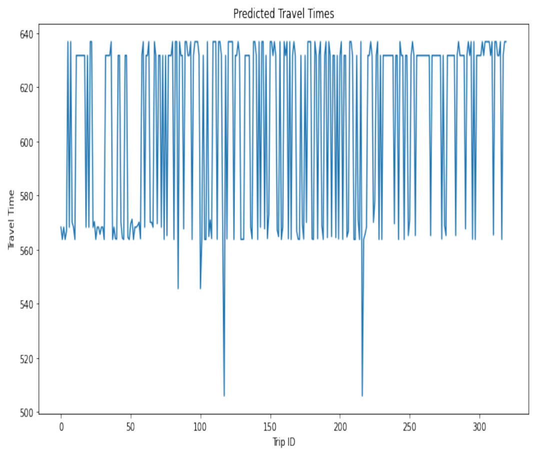
4.3 Problem c

Here is the plot for training loss vs. number of epochs.



After running for 30 epochs, we can see the exponential decay, which is what we expected.

Here's the plot of predicted travel time:



My current ranking on leader board is 76, with score of 781.70816.

5 Discussion and Future Work

5.1 Problem A

I believe that the feature engineering strategies I used is only a small part. After watching others presentations, there is still a lot of things I can apply and modify to get a better result. I should also try out different models in the future.

Gradient clipping and learning scheduling is pretty effective. Gradient clipping prevent the explore gradient problem that is causing errors in my code, and also solved some potential issue. Learning rate scheduler decrease the learning rate throughout each epoch help the model to have faster convergence.

Hyper parameter tuning really helps me increase my ranking on the leader board. Even though it only increases a bit, but it did helps me to produce a better result. The most effective tuning is probably the adjustment of the learning rate. When I change my learning rate to start at 0.01, I did get a better prediction.

The biggest bottleneck for me is the fact that I need to do everything on my own as a one person team. I end up spending more time debugging than trying out some new ideas. And I also believe that with more people in the team, each person can add some changes to the model and run their own version of model on their computer and compare the result with each other after the training. But I will have to do the training each time when I make a small adjustment to the code. This can be really time consuming.

The advise that I have to the beginner is: Before doing the implementation, it is really important to fully understand the data, as it can save the debug time on things like mismatch dimension, etc. Communicate with others when your having trouble fixing the code, to make any improvements.

Before exploring more other things, the first thing I have to do is learn from the other groups and see how I can modify my code to get a better result. After that, I will want to learn to implement a transformer, and apply it to the real world scenario.