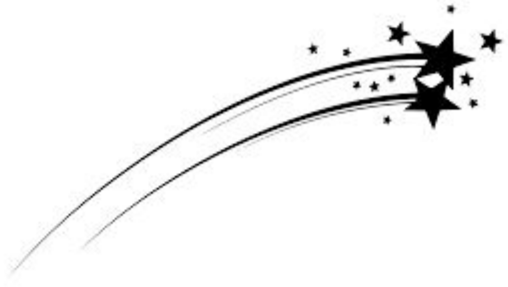
Final Project Presentation

Team: Solo Fighter
Member: Liopold Chen

Summary

- This is a one person team. I solve the problem by testing out different techniques and different values to my model and keep the best result. I acknowledge that there is a lot of way to approach this project, and many ways to modify the model for it to produce a better result.
- I use the LSTM model for the training. The best score I get for the competition on 30% of the data is 781.7.
- I will like to keep improve my model in the future and test out different ideas.



Key Words

Learnin
g Rate
Schedul
er

Dropout

LSTM

Batch
Normaliza
tion

gradien
t
clippin
g

Introduction

Team Introduction

The team name is call “Solo Fighter” because I am working on my own on this project, and I also see this as a opportunity to get better in deep learning.

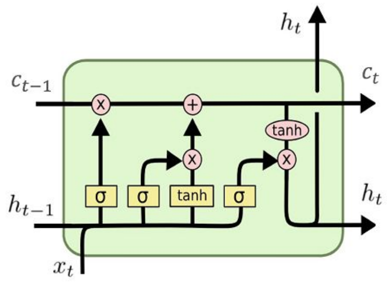
Methodology

Data Processing

- I first filter out the missing data in train_data, where the missing data = True, so I don’t need to deal with missing data.
- I use a function to calculate the travel time using the polyline column in train_data and create a new travel time column in train_data.
- I mapped the categorial features call_type and day_type from A, B, C to 1, 2, 3 for model to learn.
- The custom dataset and test dataset class takes the preprocessed data as input and stores the input features.
- I also replace any missing values with zeros in the train and test datasets to prevent issues happening in training.

Deep Learning Model

- The model I am using is recurrent neural network with long short-term memory (LSTM) module. It works well with time predicting task.
- LSTM layer takes in input_size, hidden_size, num_layers, and output_size as parameters.
- The Batch normalization layer performs batch normalization along the hidden_size dimension. It help with model stability and speeding up the training process.
- Dropout layer applies dropout regularization to the input tensor. It is use to prevent overfitting problem.
- The fully connected layer performs a linear transformation of the input features produce the final output.
- Forward method defines the forward path of the model.



LSTM
(Long-Short Term Memory)

Engineering Tricks

- I have use the exponential learning rate scheduler, that gradually decrease the learning rate over epochs. With this, it can potentially leading to faster convergence and better generalization.
- I have also applies the gradient clipping, that limit the maximum gradient norm to 1. It can prevents exploding gradients during backpropagation.
- The optimizer I used is stochastic gradient descent (SGD), with momentum to accelerates convergence.
- The anomaly detection is used to identify any computational issues or errors. It helps a lot with debugging.

Experiments

Experiment 1

- After the first successful training, the result got a score of 790.26529. The model is using some basic stuff, and the model is only be trained for 10 epochs.
- Even though it generate some kind of result, but the loss during the training is not decreasing constantly for each epochs. It seems like the loss is randomly decreasing and increasing. The features include timestamp, missing data, call type, and trip id.

Experiment 2

- I start applying things like dropout regularization and batch normalization to the model. The model seems to be learning properly and the loss is decreasing exponentially throughout the epochs. The score did not change much.

Experiment 3

- I changed the input features to call type, origin call, origin stand, timestamp, and day type to see if model can learn more from the data. But during the training, there is nan value being serve through the back propagation, resulting the nan value when generating the loss value. So I add the gradient clipping to solve the problem. I also add the learning rate scheduler to see if this can improve the training. I get a score around 788 which is much better than last result.

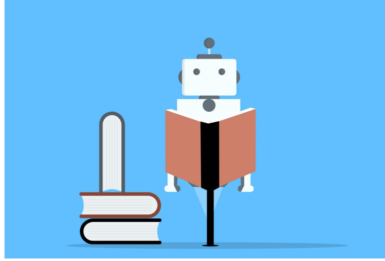
Experiment n

- During many experiments, I have also try to use the metadata and apply the longitude and latitude as an input feature for the training. But because the amount of error producing in the process, I decided to go back to the original version of the code and work from there.
- Now I've been messing around with different value of hyperparameters and dropout probability. With dropout probability of 0.7, I get a score of 787. With 0.8, I get a score of 785.7. With 0.9, I get a score of 784.3.
- Even though there is still a lot of improvement for my model, but at least there is some improvement in the score.

Discussion

What have you learned

- While trying to make my own model for this project, I gained more understanding towards the RNN and LSTM module.
- I learned the importance of data processing and how it can really affect the result of training.
- I have also learned that in order to improve the model, one need to trying out different techniques and values for hyperparameters and other parameters to find the best result, the process of trial and errors really helps a lot for improving the result.
- Next time, if I undertake a project similar to this in the future, I will actively seek out for a group. It's really hard to handle all the tasks on your own with limited ideas and simultaneously troubleshoot the code. Working in a group will be a better choice.



Future Work

- I will dive deeper into deep learning to learn more from it, and implements what I learned to get a better predicted result of this project. As what I have said, all the changes I make to my code is still some basic understanding to it.
- I will like to come up with a more unique and creative solution to the project, and trying out different models in the future.