Problem

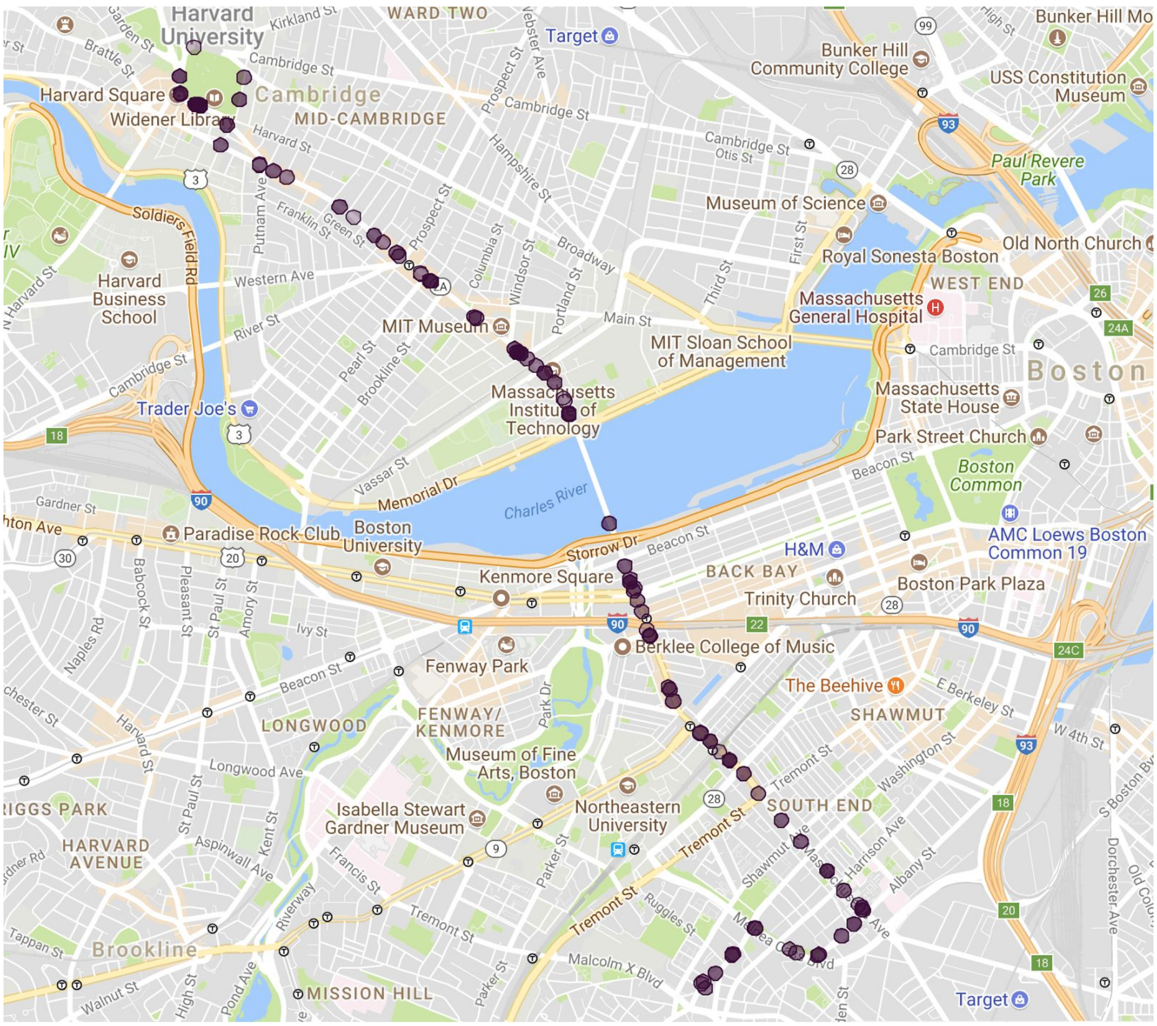
Bus networks have several unique characteristics which differentiate them from normal car networks. In particular, buses have to adhere to a schedule, and must arrive at stops at specified times. Additionally, several buses operate on the same route at the same time, and they influence one another. Specifically, buses exhibit a phenomenon called clumping. When a bus starts running behind schedule, more passengers arrive at future stops. The bus then has to wait additional time to pick up the extra passengers. This causes even more passengers to pile up at future stops, making the bus slow down even more. As this happens, buses at previous stops catch up to the delayed bus and clump up. The opposite effect happens when a bus starts running ahead of schedule. This complicated dependency between buses in a network makes predicting arrival times an interesting problem to study.

Accurate predictions can be used to make more reliable schedules and give passengers more ease of mind when using public transit. Additionally, predicting arrival times can significantly reduce waiting times for passengers. I will focus on modeling the Boston bus network with an LSTM network to predict bus arrival times with high accuracy.

All of the current models are relatively small, and are trained small datasets which span only a few months. This scale of data does not allow for training larger models or studying long-term effects. Furthermore, the current neural network techniques do not take utilize the fact that the data is sequential. Recurrent neural networks are more suited to sequential data than standard feed forward neural networks because they have sequential sensitivity.

Approach

The dataset I used is GPS data from the Massachusetts Bay Transportation Authority (MBTA). The MBTA runs a server which can be queried to get the latitude and longitude of all of the buses in Boston, along with other metadata about routes and stops. Jiahao collected several years worth of the GPS data. The dataset is several terabytes in raw XML, though it can be compressed to a more manageable size. The following image is a sample of the MBTA dataset.



The GPS data then has to be converted into features which can be used as inputs to the neural network. One important features is travel time or the difference in arrival times between two adjacent stops. Another common feature I used is the field is dwell time, or the amount of time the bus stays at each stops before leaving. I also used schedule adherence, which is the difference between the scheduled arrival time at a stop and the actual arrival time. I computed these values from the data by comparing the GPS location of the bus with published GPS locations for each of the stops, and doing interpolation. The feature vectors are composed by combining each of the metrics for each bus over its entire trip on a route.

Bus trajectories are explicitly time series data. This type of data is well suited to recurrent neural networks. Furthermore, there is important state about the current traffic network which needs to be learned in order to predict arrival times. This suggests incorporating some state into the model to reflect the current traffic conditions. Long short term memory (LSTM) networks are well suited for modeling this type of interaction. Furthermore, the size of the dataset and the complicated nature of modeling traffic networks suggest that larger networks may be well suited to the problem. Therefore I applied deep RNNs to model bus arrival times.

The model architecture is an LSTM with several hidden layers, and a dense output layer. LSTM models are based on how human memory works. The following figure is from the very useful blog post by Christopher Olah[1]. It shows an LSTM gate, the basic building block of LSTM networks. It consists of three gates, a forget gate, and input gate and an update gate. These gates work together to store and update the memory of the unit. Several of these gate combine together to make up an LSTM network.

---

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

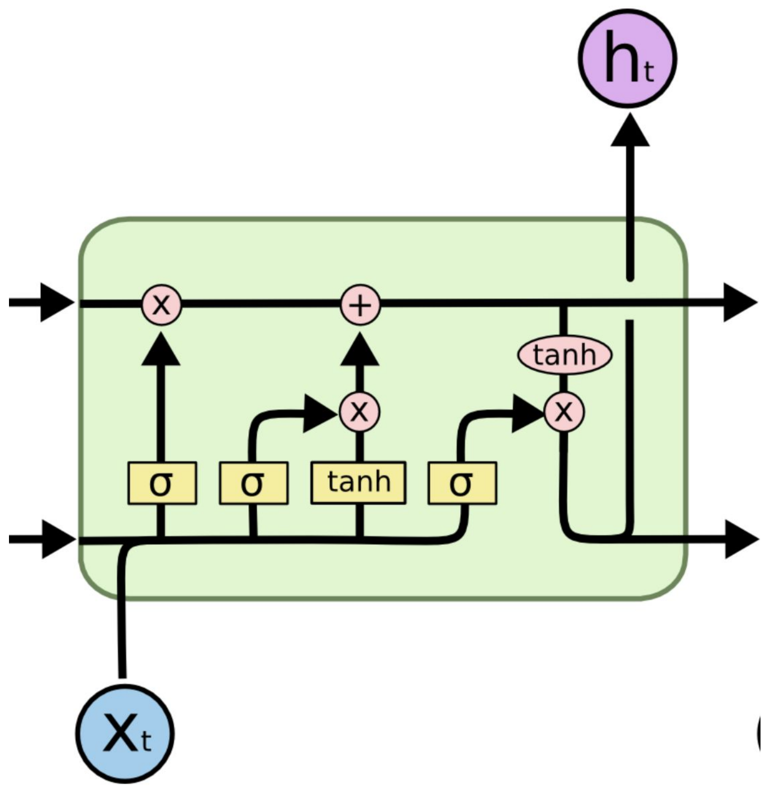


Figure 1: LSTM unit

Figure 2 shows an example of a simple LSTM. The blue boxes represent the network, green boxes represent training data, and the red boxes represent predicted probability distributions. LSTMs are just like normal feed forward models, except they have state which persists between consecutive passes through the network. The state is passed from one timestep to the next, and updated via a module called an LSTM gate. The state is combined with the input, passed through a series of linear and nonlinear transformations, and a probability distribution is the output. The probability distribution represents the networks belief about what the next input datapoint will be.

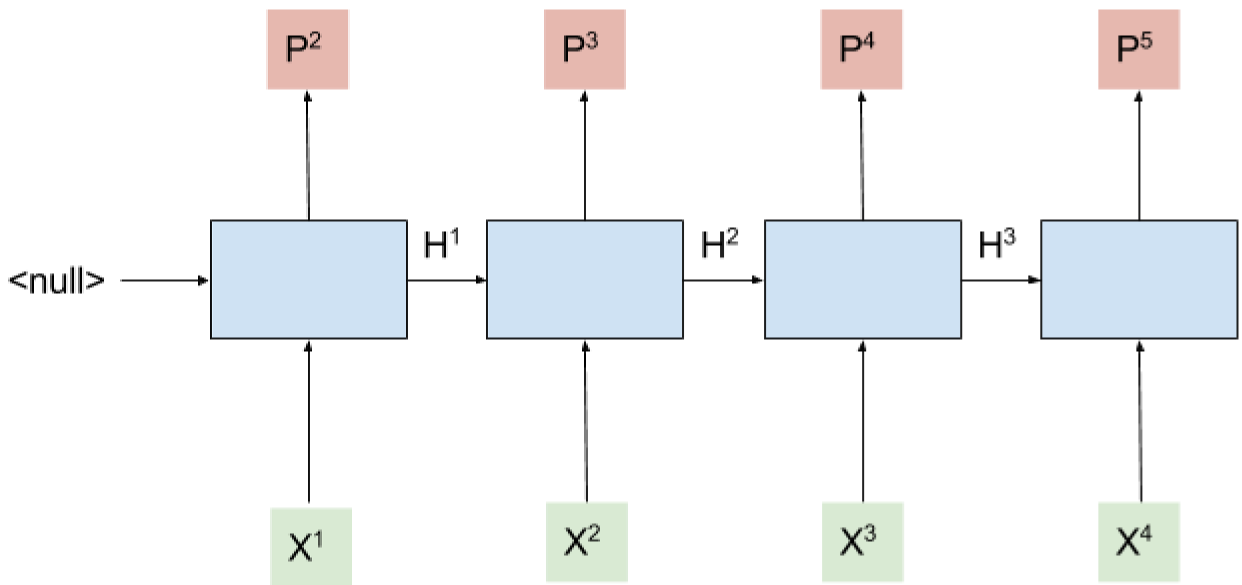


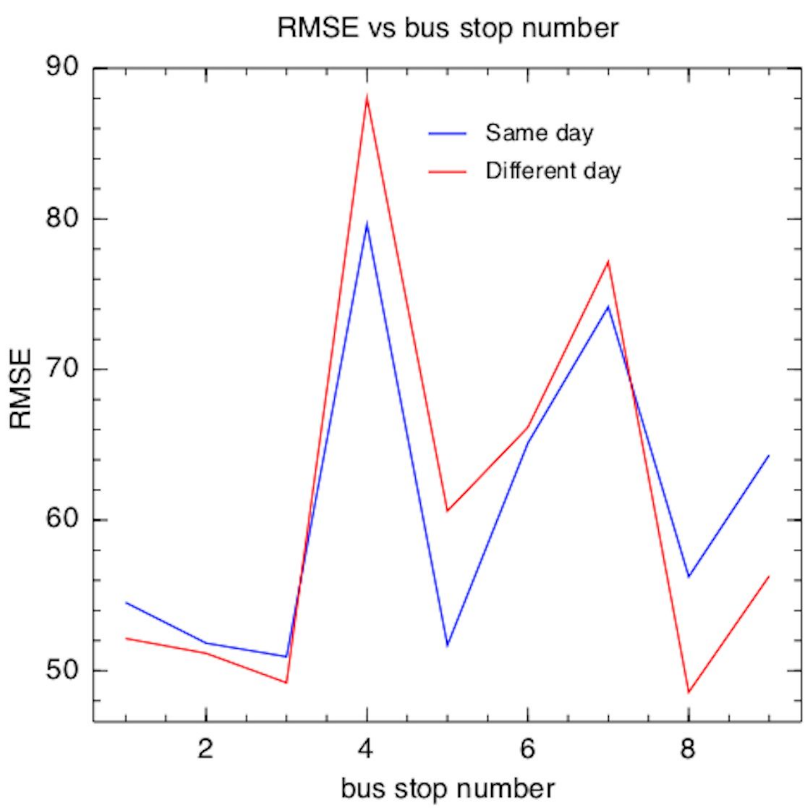
Figure 2: Simple LSTM

These types of networks can be trained in an unsupervised manner by feeding example training data through the network. The network can then be used to predict future sequences given a prefix. In this case the training data is a sequence of features vectors for each stop on a route. Each feature vector consists of the travel time, dwell time, and schedule adherence for each stop. The training data was used sequentially to train the model parameters via backpropagation through time. The dataset contains GPS data for all of the routes in Boston, so a different model was trained for each route. The accuracy for predictions along each route can then be compared.

Additional characteristics which can be studied with this type of model are long-term and periodic effects. Traffic networks exhibit periodicity on a daily, weekly, and yearly basis. The size of the dataset allows for studying long-term behavior. Furthermore, LSTMs are well suited to modeling short term as well as long-term behavior. In order to study this behavior, temporal information can be added to the feature vector such as day of week, time of day, and year. The accuracy on test data can then be compared to the model trained without temporal data to determine the significance of periodic effects.

Results

The following graph shows the error rate of prediction for test data on a simple feed forward neural network. With a few exceptions, most prediction fall between 50-70 seconds of the actual value.



In general this matches up with research. The large amount of variance and stochastic nature of traffic networks limit the accuracy of predictions. However these predictions are good enough for most passengers. The RNN model does not perform quite as well, though I hope to improve it.

Conclusion

Predicting bus arrival times can significantly decrease wait times for passengers. However current models are too simple to capture periodic and long-term effects. The MBTA dataset is unique in its time scale and size, and can be used to train larger models. Neural networks work fairly well on predicting trajectories. However, LSTMs may hold the answer to improving prediction accuracy by modeling both short term and long-term periodic behavior.