NYC Taxi Fare & Demand Surge Prediction using Graph Neural Networks

Abstract

Following the successful application of embeddings such as word2vec or BERT to NLP downstream tasks such as sentiment classification or text understanding, various graph neural networks (GNNs) methods have been proposed to form good embeddings for data with inherent graph structure. Using NYC yellow taxi data from 2009-2020[1], we construct a graph between pickup/dropoff locations, and we explore applying some of these methods (e.g. Graph Convolutional Networks, GraphSAGE, Graph Attention Networks) to this graph. We formulate regression forecasting problems including fare price to demand surge prediction. We compare the performance of graph neural networks for these tasks to the other machine learning methods including linear regression, random forests, and feed-forward neural networks.

1. Motivation

With the advent of ridesharing (Uber, Lyft, etc.), it has become increasing important for the taxi industry to provide users and drivers with accurate predictions regarding trip fares and demand. These predictions can help identify locations of future demand and aid in redistributing drivers to maximize availability. The inherent graph structure formed from pickup and drop locations is not adequately captured using other Machine Learning (ML) methods such as linear regression, feed-forward neural networks, and random forests. However, Graph Neural Networks (GNNs) can leverage this structural information from the graph to predict node features. The GNN method takes advantage of the graph structure by propagating information from the neighborhood of the node in constructing the embedding for the node. We examine applying GNNs to NYC taxi trip data to construct models that predict taxi fares and demand surges. We hypothesize that by using GNNs for these prediction tasks, this method will yield better results than other machine learning methods that do not leverage the graph structure of the data in their predictions.

[1] Link for NYC Yellow Taxi Data

2. Related Work

2.1. Graph Neural Networks

Graph Neural Networks encode inductive biases relevant to graphical data into their architecture: they are invariant under node permutations and respect locality structure through message passing algorithms. *Notation:* A graph $\mathcal{G}$ is defined by two sets $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents its vertices and $\mathcal{E}$ its edges. We denote the neighborhood of an arbitrarily node $v \in \mathcal{V}$ by $\mathcal{N}_v$.

The *modeling strategy* is to build representations $h_v \in \mathbb{R}^d, \forall v \in \mathcal{V}$, from the nodes' input feature by applying a series of layers where each update is informed by a node v's current representation and those of its neighbour. We review three established architectures and note that in practice they perform comparably. At a high-level they all pool the neighbors' features and then combine this information with a node's self-embedding to update representations using a non-linear activation function σ.

Graph Convolution Networks (Kipf & Welling, 2016) treat self-embeddings and the neighbors' embeddings the same and pool them together by simply computing an average:

$$h_v \leftarrow \sigma\left(W_k \sum_{u \in \mathcal{N}(v) \cup v} \frac{h_u}{\sqrt{d_u}\sqrt{d_v}}\right)$$

, where $W_k \in \mathbb{R}^{d \times d}$. **Graph SAGE (Hamilton et al., 2017)** aggregates the neighbors' features using either the mean operation, an LSTM, or pooling and intro-

duce an additional parameter matrix $B_k \in \mathbb{R}^{d \times d}$:

$$h_v \leftarrow \sigma\left(\text{concat}[W_k \cdot \text{Aggregate}(h_j)_{j \in \mathcal{N}_i}, B_k h_i]\right)$$

Graph Attention Networks (Veličković et al., 2018) introduce an attention mechanism to update node features by computing similarity measures between $v \in \mathcal{V}$ and all its neighbors $u \in \mathcal{N}_v$.

$$h_v \leftarrow \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha(h_v, h_u) W_k h_u\right),$$

where the similarity $\alpha(h_v, h_u) = \text{softmax}(e_{uv})$ and the attention coefficients $e_{uv} = \langle W_k h_u, W_k h_v \rangle$.

3. NYC Taxi Data set

The New York City Taxi & Limousine Commission provides data for all yellow taxi trips that have occurred since 2009. There are 265 taxi zones, each of which has a unique ID. Yellow taxi trips go from one taxi zone (pickup location) to another (dropoff location). Some of the fields in the dataset that we'll be using as features include the following:

- Pickup Timestamp
- Dropoff Timestamp
- Fare Amount (in USD)

Similar to (Upadhyay & Lui, 2017), we will derive some other important features of the taxi trips from the features above. These include:

- Surge Value (See Section 4.3)
- Boolean indicating whether trip occurs on weekend / holiday
- Time of Day in bins consisting of 4-hour intervals:
  - {2am-6am, 6am-10am, . . . , 10pm-2am}
- Location Information

We differentiate between weekdays and weekends/holidays because traffic patterns change significantly between the two. Additionally, we construct a similar categorical feature for the time of day interval. In the future, we might explicitly incorporate further a-priori knowledge about the data's temporal aspect, such as differentiating between seasons. We incorporate additional location information extracted from the taxi zone ID.

4. Experiments

Since the NYC Yellow Taxi Dataset has an immense amount of data for each month (approximately 7 million rides per month) over 11 years, we will train and test our models, at least initially, using yellow taxi trip data from January-June 2019. We decided not to use recent data from 2020, as COVID-19 has impacted the use of taxis due to safety precautions such as limiting use of taxis and ridesharing for transportation.

4.1. General Graph Description

Given the nature of taxi rides having a pickup and dropoff location, NYC taxi zone locations can be represented as nodes in a graph. The edges between these nodes represent aggregate features of directed trips that go from the pickup node to the dropoff node. An example of this kind of graph from the surge prediction experiment described below is in figure 2.

Graph Neural Networks do not have an explicit method to contain edge feature information; however, there are methods to resolve this limitation. The most common method is by converting the original graph to a bipartite graph where each edge is replaced by a node containing the edge's features and two new unweighted edges (Zhou et al., 2019). Prediction can then be done on the original nodes or the edge feature nodes. We construct two versions of this bipartite graph with varying features for use in our two experiments: fair prediction and demand surge prediction.

A practical consideration is that of developing sparse initial graphs in order to learn robust representations and maintain computational efficiency. This is easily achievable through an initial pruning data preprocessing step where we remove edges between any two locations that are connected by very few rides.

We test this is feasible by plotting the histogram of in- and out- degrees for each node in the network (Fig. 1) and notice they are indeed concentrated around values $\ll$ the total number of nodes. This suggests a simple pruning operation is indeed all we need to construct our graph structure (Fig. 2).

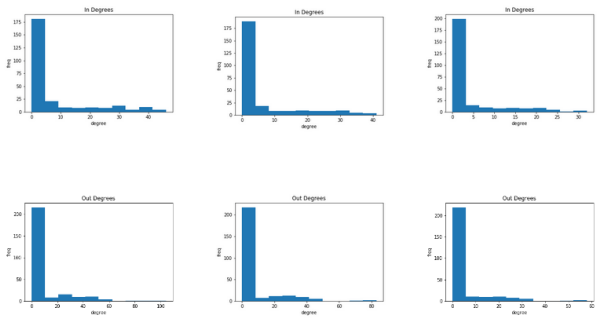

Figure 1. Distribution of nodes' degree from network graphs constructed from time-intervals of different lengths: 10, 20, or 30 minutes. We notice the distribution stays relatively constant regardless of the observation length and is peaked towards lower values.

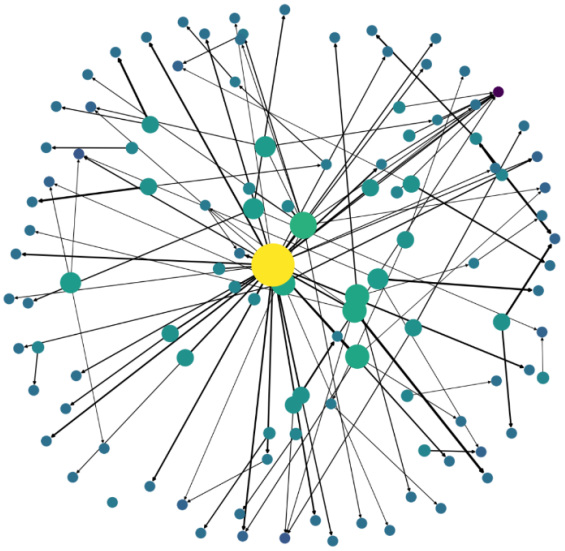

Figure 2. Example graph constructed from taxi ride data from a ten-minute time window on 1/01/2019 from 5:50-6:00pm. Each node is a location and the directed edges are taxi rides between these locations. This graph is a sample from the surge prediction experiment in section 4.3. Node color and size represent a larger "surge value" and edge weight represents a larger volume of rides between locations. The bipartite graph described below would be constructed from these nodes and edges.

4.2. Predicting Taxi Fares

4.2.1. Graph Data Specification

For the task of predicting taxi fares, the edge feature nodes in the graph represent the aggregate of all trips that occur at a certain time interval between pickup and dropoff nodes. For example, there exists a specific edge in this graph that represents an aggregation of all trips that occur between 6pm-10am from Grand Central Station to Time Square on 3/15/2019. As mentioned in Section 4.1, we can construct a bipartite graph, where these edges become nodes in the graph with features of the edge including the pickup and dropoff locations associated with the trips, the time of day these trips took place, and whether or not the trips take place on a weekend or holiday.

The graph described above is an example of the input to the GNN. The corresponding target label to this input data is a similar bipartite graph where the edge nodes have an associated value corresponding to the average fare price for the aggregated trips associated with that edge node. We assume the average fare price is a comprehensive estimate for any one trip from a pickup to dropoff location. Using this estimate for any particular trip, could be a potential fail point, if the specific path taken during a trip has a significant impact on the fare amongst other features that may not be accurately captured by using aggregate data from multiple trips.

For prediction, the GNN uses tuples of input graphs representing the aggregated trips and target label graphs that have the associated average fare for those trips and uses Mean Squared Error (MSE) for determining loss, thus heavily penalizing predictions that are very different from the actual average fare for those trips. The model learns the weights to approximate the function to determine the average fare for a set of trips between a pair of pickup and dropoff locations at a certain time of day, on a weekend/holiday or weekday.

4.3. Predicting Location Surges in Taxi Demand

4.3.1. Graph Data Specification

For the surge prediction task, we restrict our ride data to two, sequential, ten-minute time windows. We construct separate bipartite graphs from the two time windows where edge features are the number of rides from one location node to another. We then compute the "surge value" of each location node as the difference of it's out-degree and in-degree. Locations where the out degree is much higher than its in degree (a positive

"surge value") are deemed as surging locations as the demand for vehicles in the area is high.

The training data is formed from these sequential graphs. The earlier graph is used as the input features to the GNN model. The later graph forms the target prediction labels–particularly the surge values at each location node.

4.4. Graph Neural Network Methods

We leverage the DGL library (Wang et al., 2019) which efficiently implements standard graph neural network architectures viewed under the message passing paradigm using parallelized sparse-dense matrix multiplication and sampled dense-dense matrix multiplication operations. We will explore multiple GNN variants such as Graph Convolution Networks and Graph Attention Networks

4.5. Evaluation

Our goals with using GNN models are to accurately predict taxi fares and the amount of surge expected at different NYC locations. To evaluate the efficacy of our models to meet these goals, we present the following split in the data for training, validation, and testing along with an associated evaluation metric.

For the taxi fare prediction problem, we split the data into training, validation, and test randomly in proportions of 80%, 10%, 10% from January 2019-June 2019.

For prediction of future demand surge amount, we will likewise spilt our contrived data-set (described in section 4.3) into training, validation, and testing subsets of the same proportions. The surge data-set will also be constructed from data ranging from January to June 2019.

For evaluating the GNNs used in each of these predictions, we use the evaluation metric of Mean Squared Error (MSE). We will compare the MSE values against alternative machine learning methods and the results of the methods as discussed in 5.

5. Alternate Methodologies

We plan to evaluate the performance of alternate ML methodologies on the regression tasks of yellow taxi fare and demand surge prediction. Using the methods of Linear Regression, Neural Networks, and Random Forests, we devise a basis of comparison with the performance of these methods on these regressions tasks to compare with the performance of our GNNs on the same regression tasks. We elected to compare the performance of these models against the performance of our GNN method because these methods do not leverage the data's inherent graph structure. We will use the same features, train, validation, and test data that we used for the GNNs as discussed in Section 4.5 for these methods to have consistent evaluation.

6. Future Work

We have presented the NYC Yellow Taxi Dataset, devised methods using GNNs for the tasks of taxi fare and surge demand prediction, and introduced several interesting graph-learning benchmarks. In addition to yellow taxi data, the NYC Taxi and Limousine commission provides trip data for popular rideshare services such as Uber and Lyft. The data for ridesharing trips have similar fields to the yellow taxi dataset. Since the trip data for ridesharing services in NYC forms a graph structure and has similar features to the yellow taxi data, similar regressions tasks such as those discussed in this paper can be formulated. Therefore, GNNs should also be applied and evaluated for the ridesharing trip data. Other predictions tasks can also be formulated using the yellow taxi data such predicting locations where the largest fares can be earned and at what times these large fares can be made. GNNs can be trained and evaluated for these tasks and the performance can be compared against other machine learning methods. There are other features we could have derived that may have impacted our results such the season in which the trip took place, whether the trip was headed to a particular popular location such as an airport or tourist site, which (Upadhyay & Lui, 2017) utilized for their predictions of taxi fares in Thessaloniki, Greece.

References

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. URL http://arxiv.org/abs/1706.02216.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Upadhyay, R. and Lui, S. Taxi fare rate classification using deep networks. 09 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks, 2018.

Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., Li, M., Zhou, J., Huang, Q., Ma, C., Huang, Z., Guo, Q., Zhang, H., Lin, H., Zhao, J., Li, J., Smola, A. J., and Zhang, Z. Deep graph library: Towards efficient and scalable deep learning on graphs. *CoRR*, abs/1909.01315, 2019. URL http://arxiv.org/abs/1909.01315.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications, 2019.