Probabilistic Inference of NYC Congestion from Taxi Data

Jiafeng Chen Yufeng Ling Francisco Rivera

Abstract

1. Introduction

2. Background

The NYC Taxi and Limousine Commission (TLC) makes available[1] a dataset of taxi rides taken in the city. For our purposes, each ride is characterized by the time and location (latitude and longitude) of the pickup as well as the corresponding quantities for the drop-off[2]. Moreover, there is an abundance of data. In January of 2009 alone, there were over 14 million taxi rides recorded.

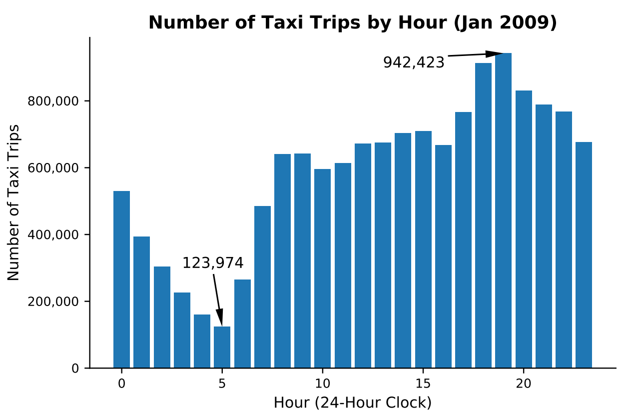


Figure 1. Taxi trips by time of day

When we aggregate by hour of day (Figure 1), we see some variance with anywhere between a hundred thousand and a million trips per interval. This is reassuring for future bucketing of the data.

While we are not lacking in data volume, two important challenges do appear when working with this dataset. The first is that the data set is not completely clean. Case in point, while most trip durations (Figure 2) are reasonable

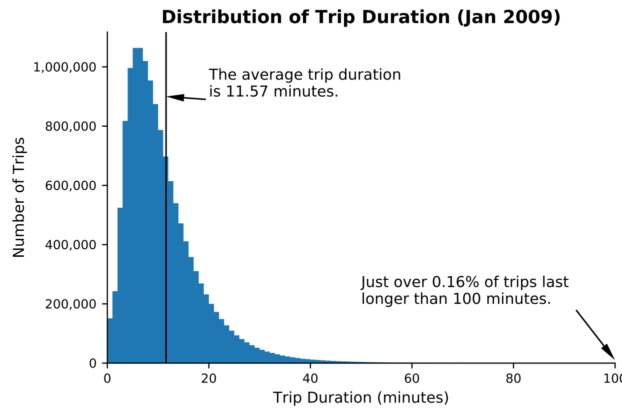


Figure 2. Taxi trips by duration

for a taxi trip, just over 1% of trips have a drop-off time that is before the pick-up time, making for a trip with negative duration. Furthermore, the longest trip has duration of just over 42 days. It is difficult to accurately assess the veracity of the pickup and dropoff coordinates, but we observe that some trips have geographic coordinates that place the start or end of the trip in the ocean.

Additionally, whereas we would like to infer road conditions along the route of a taxi trip, we can only observe the start and end-points. The fact that the route taken is unobservable is primarily responsible for our inference challenges.

3. Related Work

Travel time prediction has historically been a topic of research interest. On the topic of freeway travel time prediction, (Van Lint et al., 2002) proposed a Recurrent Neural Network (RNN) approach in order to address the temporal aspect of traffic prediction; (Wu et al., 2004) adopted the Supportive Vector Regression (SVR) method applied to time-seires analysis and reduced prediction errors in cases where previous methods generate especially large errors. We notice that earlier papers share the common feature of using freeway data for training and prediction, which probably arised from the limited availability of GPS data sources. Nevertheless, we realize that even for freeways,

[1] `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`

[2] Additional information orthogonal to predicting trip duration—such as passenger counts and tip—is discarded

which in general have less traffic conditions than city roads, there is evidence that there exists a high level of non-linearity in the data, which contributes to the good performance of models such as RNN and SVR.

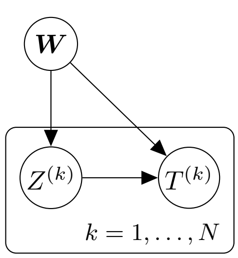
With the ubiquity of taxi GPS data, recent work focused on estimating the traffic conditions in cities using sparce probe data and used travel time predictions as the metric as the performance of the model. (Hunter et al., 2009) uses a Bayesian framework and used an expectation maximization algorithm that simultaneously learns the likely paths taken by probe vehicles as well as the travel time distributions through the network; (Herring et al., 2010) used a Coupled Hidden Markov Model (CHMM) and determines the most likely path by allocating the travel time between two consecutive location observations to roads in the 'M' step.

One thing that distinguishes our attempt to the previous work is that we have much more missing information. Instead of sparse probe data, we are only working with start and end locations, which presents much more challenges in actual path inference and parameter optimization. (Zhan et al., 2013) provided some very helpful insights since it used the same TLC dataset of taxi ride. It contructed a faithful representation of the manhattan streets network, but limited the discussion to a much smaller district in Manhattan to reduce computational complexity.

4. Model

We represent a city's road network with a connected graph $G = (V, E)$. Assume that each vertex $i \in V$ is associated with a weight w_i, representing the cost of traversing vertex i. A trip is represented by a path in G, and the distribution of the trip's duration depends on the weights w_i of vertices included in the path. Note that the choice of the path can in general depend on the collection of weights $\boldsymbol{W}$. In full generality, the model is represented by Figure 3, where trips in the data are indexed by (k), $T^{(k)}$ is the observed trip duration, and $Z^{(k)}$ is the path taken by trip k, a latent variable. Our primary interest is to perform inference on $\boldsymbol{W}$, so as to learn the levels of congestion associated with each vertex in G.

Figure 3. Representation of model as a directed graph

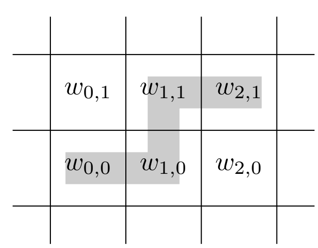


4.1. Parameterization

In principle, in our application to the New York City taxi data, we may take G to be a graph representing the exact road network in New York City, where each vertex is a *road segment* and the directed edge $(i, j) \in E$ if one can drive directly onto road segment j from road segment i; such a parameterization allows w_i to be directly interpretable as a measure of congestion on road segment i. However, such a detailed construction presents serious computational challenges when training on a large dataset, since solving path-finding problems and computing minimal paths are non-trivially expensive.[3]

To avoid these challenges, we parameterize G as an undirected rectangular grid. Despite not being able to pinpoint weights w_i to congestion of specific road segments, we are nonetheless able to interpret the weights w_i as representative of congestion on a small patch of land. We may now represent a path $Z^{(k)}$ as a set of indices i of grid points traversed by the path. In full generality, there are an infinite number of paths connecting any two points i, j on the grid, but the vast majority of these paths are not sensible. Thus we restrict the set of possible paths for trip k to a *set of reasonable paths* $\boldsymbol{Z}^{(k)}$, where each path in $\boldsymbol{Z}^{(k)}$ travels strictly in the direction of the destination. For instance, if the destination of j is to the northeast of the starting location i, then the set of reasonable paths $\boldsymbol{Z}$ are the set of paths that only involve northward or eastward movements (e.g. Figure 4 shows a reasonable path from $(0, 0)$ to $(2, 1)$). Such a parameterization is more general than many in the literature; Zhan et al. (2013), for instance, uses the K-shortest path algorithm and considers the shortest 20 paths as a set of reasonable paths.

Figure 4. An example of a reasonable path



We parameterize the conditional distribution of $T^{(k)}$ as Normal, in the following reformulation of the directed

[3]Manhattan has on the order of 10^4 road segments, and the dataset contains the order of 10^7 trips for January 2009 alone.

graphical model:

$$\boldsymbol{W} \sim p(\boldsymbol{W})$$
$$Z^{(k)} \sim p(Z^{(k)}|\boldsymbol{W})$$
$$T^{(k)}|\boldsymbol{W}, Z^{(k)} \sim \mathcal{N}\left(\sum_{i \in Z^{(k)}} w_i, \sigma^2\right),$$

where $p(Z^{(k)}|\boldsymbol{W})$ is a distribution over $\boldsymbol{Z}^{(k)}$. We consider two different ways to parameterize $p(Z^{(k)}|\boldsymbol{W})$: softmax regression and uniform. In the *softmax regression* model, a type of generalized linear model for discrete choice problems (McFadden et al., 1973), we parameterize the route choice such that

$$p(Z^{(k)}|\boldsymbol{W}) \propto \exp\left(-\sum_{i \in Z^{(k)}} w_i\right),$$

in order to encode the fact that drivers avoid routes that take a long period of time. In the *uniform* model, we simply assume that route choice is independent and uniform on the set of reasonable paths:

$$p(Z^{(k)}|\boldsymbol{W}) \propto 1.$$

The uniform model trades off realism in modeling for improvement in computation and training, as we see in Section 5.

In our application to the Manhattan dataset, we perform MLE inference, or, equivalently, MAP inference with $p(\boldsymbol{W}) \propto 1$. In principle, it is not difficult to parameterize the prior of $\boldsymbol{W}$ as an undirected graphical model, since we need only to supply edge and unary potentials. For instance, to impose a correlated prior on $\boldsymbol{W}$, as suggested by some (Hunter et al., 2009), we simply penalize large differences in neighboring weights in the edge potential, effectively assuming a prior model that is similar to a continuous version of the Ising model.

5. Inference

We perform maximum likelihood inference, maximizing

$$\max_{\boldsymbol{W}} \log p(\{T^{(k)}\}_{k=1}^{N}|\boldsymbol{W}) = \max_{\boldsymbol{W}} \sum_{k=1}^{N} \log p(T^{(k)}|\boldsymbol{W}).$$

The log-likelihood is

$$\begin{aligned}
&\log p(T^{(k)}|\boldsymbol{W}) \\
&= \log\left(\sum_{Z^{(k)} \in \boldsymbol{Z}^{(k)}} p(T^{(k)}|Z^{(k)}, \boldsymbol{W}) p(Z^{(k)}|\boldsymbol{W})\right) \\
&= \log\left(\mathbb{E}_{Z^{(k)}}\left[p(T^{(k)}|Z^{(k)}, \boldsymbol{W})\right]\right).
\end{aligned}$$

The expectation is a sum of the size $|\boldsymbol{Z}^{(k)}|$, which, for an $m \times n$ trip[4], is of $\binom{m+n}{n} \approx \frac{(n+m)^n}{n^n} e^n$ terms. Computing this expectation is the main inference challenge of our project.

5.1. Uniform Model

By assuming the uniform model $p(Z|\boldsymbol{W}) \propto 1$, we gain the ability to work with an expectation over $\boldsymbol{W}$ instead of an expectation over $\boldsymbol{Z}$, since the probability that a particular weight is included in a path is readily computable from elementary combinatorics. We maximize an approximate lower bound of the log-likelihood by applications of Jensen's inequality:

$$\begin{aligned}
\ell(\boldsymbol{W}) &= \log\left(\mathbb{E}_{Z^{(k)}}\left[p(T^{(k)}|Z^{(k)}, \boldsymbol{W})\right]\right) \\
&= \log\left(\mathbb{E}_{Z^{(k)}}\left[\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(T^{(k)} - \sum w_i)^2}\right]\right)
\end{aligned}$$

Note that the Normal density is concave for $|T^{(k)} - \sum w_i| < \sigma$ and convex otherwise. Applying Jensen's inequality locally, we obtain that

$$B(\boldsymbol{W}) = \frac{1}{2\sigma^2}\left(T^{(k)} - \sum_i w_i \pi_i\right)^2,$$

where π_i is the marginal probability of node i being included in a uniform route,[5] is a local lower bound for the negative log-likelihood for trips that we predict poorly and is a local upper bound for trips that we predict well, up to a constant. Thus $B(\boldsymbol{W})$ is a good approximation of the objective function that is easily computable, involving only mn as opposed to $\binom{m+n}{n}$ terms.

5.2. Softmax Regression Model

We now consider a more realistic but more complex model, where we parameterize $Z|\boldsymbol{W}$ as a GLM, namely as a softmax regression where $p(Z|\boldsymbol{W}) \propto \exp\left(-\sum_{i \in Z} w_i\right)$, encoding drivers' preferences for shorter trips. The log-

[4] By an $m \times n$ trip, we mean a trip with east-west distance n and north-south distance m

[5] π_i can be computed analytically. Suppose the source and destination of the trip are (n, m) apart and vertex i is (a, b) away from the source. Then, by elementary combinatorics,

$$\pi_i = \frac{\binom{a+b}{a}\binom{n+m-a-b}{n-a}}{\binom{n+m}{n}}$$

likelihood in this model is

$$\begin{aligned}&\ell(\boldsymbol{W})\\&=\log\left(\mathbb{E}_{Z^{(k)}}\left[\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2\sigma^2}(T^{(k)}-\sum_i w_i)^2}\right]\right)\\&=\log\left(\sum_{Z^{(k)}\in\boldsymbol{Z}^{(k)}}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2\sigma^2}(T^{(k)}-\sum_i w_i)^2}p(Z^{(k)}|\boldsymbol{W})\right)\end{aligned}$$

Before we discuss our inference strategy, we first discuss a few difficulties of the model. At first glance, the latent variable structure seems lend well to the Expectation-Maximization algorithm (Dempster et al., 1977). However, the EM algorithm requires computing the term

$$Q(\boldsymbol{W}|\boldsymbol{W}^{(t)})=\mathbb{E}\left(\log(T^{(k)},Z^{(k)}|\boldsymbol{W})|T^{(k)},\boldsymbol{W}^{(t)}\right),$$

and computing the expectation requires the conditional distribution $p(Z|T,\boldsymbol{W})=\frac{p(T|Z,\boldsymbol{W})p(Z|\boldsymbol{W})}{p(T|\boldsymbol{W})}$, where the denominator is intractable to compute. We might also consider techniques in variational inference (Blei et al., 2017). However, the latent variable $Z^{(k)}$ is a random set of indices with the property that the indices form a path on the grid, and thus cannot be reasonably decomposed into independent components, ruling out the mean-field algorithm. Another promising option is stochastic gradient variational Bayes (SGVB) and variational autoencoders (Kingma & Welling, 2013), which is designed for large datasets for which the EM algorithm fails. However, SGVB requires a reparameterization of the latent variable drawn from an approximate distribution, in order for the gradients to be properly computed. In SGVB, one independently draws some ϵ from a distribution and obtains z via $z=g(\epsilon)$ for some continuous function g.[6] This is difficult to do in our context, since Z is not continuous nor scalar-valued. Thus it is difficult to find an ϵ and a continuous transformation to approximate samples from the distribution of Z.

Our inference strategy is based on a sampling-based approximating to the expectation over $\boldsymbol{Z}$. The key trick we use is that

$$\mathbb{E}_Z\left[f(Z^{(k)})\right]=|\boldsymbol{Z}^{(k)}|\mathbb{E}_{\tilde{Z}}\left[f(\tilde{Z}^{(k)})p_Z(\tilde{Z}^{(k)}|\boldsymbol{W})\right],\quad(1)$$

where $\tilde{Z}$ is drawn uniformly from $\boldsymbol{Z}$. This is the same technique as in importance sampling, except here the objective is to derive computationally tractable approximations, rather than maximizing the efficiency of the approximations. Exchanging log and expectation operator and applying (1) yields the following upper bound for negative log-likelihood, up to a constant,

$$\frac{1}{2L\sigma^2}\sum_{j=1}^{L}\left[T^{(k)}+\sum_{i\in\tilde{Z}_j^{(k)}}w_i\right]^2+\frac{1}{L}\sum_{i=1}^{L}\sum_{i\in\tilde{Z}_j^{(k)}}w_i\quad(2)$$
$$+\operatorname*{logsumexp}_{i=1,\ldots,L}\left(-\sum_{\tilde{Z}_i}w_i\right),$$

where we sample uniformly and independently $\{\tilde{Z}_j^{(k)}\}_{j=1}^{L}$. The pseudocode of the implementation is detailed in Algorithm 1.

Algorithm 1 Training algorithm for softmax path selection

{Optimizing using `torch.optim.Adam`}
for trip k in $1,\ldots,N$ **do**
  Sample L random paths uniformly.
  Compute the objective function as in (2) for observation k.
  Compute the gradients of the objective function with respect to $\boldsymbol{W}$.
  Update $\boldsymbol{W}$.
end for

5.3. Prediction

We conclude this section by briefly discussing prediction in our models. In the uniform model, prediction is straightforward. We calculate the expected time $\widehat{T}=\sum_i w_i\pi_i$, where π_i is the marginal probability that node i is included in a uniformly randomly chosen path. Computing prediction is slightly more subtle in the softmax regression model. In the softmax regression model, we sample L_{pred} uniformly chosen paths and take the weighted average of the sums of w_i in each path, where the weights are proportional to $\exp(-\sum_{i\in Z}w_i)$ for path Z.

6. Methods

6.1. Data Processing

For our dataset, we started with the TLC dataset of all NYC taxi trips taken in January 2009. From the raw data, we added several computed columns. We one-hot-encoded columns for hour of day (24 columns) as well as for day of the week (7 columns). We also explicitly computed the duration of the trip in seconds as the dependent variable to predict.

In addition to adding columns, we filtered the original dataset. We only consider trips whose start- and end-locations that falls inside a rectangle that bounds Manhattan. We also removed trips with duration less than two minutes or more than two hours.

[6] Here we are using the notation in (Kingma & Welling, 2013).

We performed a principal component analysis on all de-meaned and standardized geographic coordinates (both start and end locations) from the filtered data. We thus transform the coordinates into the first and second principal components. This allowed us to compute the rotational matrix so that Manhattan lies approximately orthogonal to the x-axis. For the discrete grid models, these coordinates were further discretized into a 20×70 grid. Each node in the grid approximately corresponds to a $400\text{ft} \times 400\text{ft}$ area. We chose the grid size as a compromise between computational tractability and model expressiveness.

We saved the dataset as a whole, and we also partitioned into (day of week, hour) groups, using the one-hot encoded vectors. Some models were trained on the entire dataset (e.g. Neural Network), and some were trained individually on the partioned datasets (e.g. Structural Models).

6.2. Baselines

We calculated three baselines. The first was a linear regression predicting trip duration from trip distance (calculated using the Pythagorean Theorem and the start and end coordinates), and trained on a partitioned data set.

The second was a neural network trained on the entire dataset taking as input the start and end coordinates as well as the day of week and hour one-hot encoded vectors. The neural network had one hidden layer with 50 neurons and a $\tanh$ activation function.

The third baseline was constructed by querying the Google Maps Distance Matrix API.[7] While there was no need to train this baseline, there was a rate limit constraint on how many test data points it could be tested on. We assess its performance on 2,500 data points in a representative partioned test dataset.

6.3. Training

Training implementations varied from model to model. In training the neural network baseline, SGD was used with a learning rate of 10^{-7} run for 12 epochs on the full data set. This took between 30 minutes and an hour. The uniform paths structural model was similarly trained with SGD, with a learning rate of 10^{-4}.

The uniform model was only trained on a subset of the dataset corresponding to all trips on a given day of the week and on a given hour. Convergence appeared after roughly two epochs, and the model was run for 5 epochs taking just under ten minutes.

The softmax regression model was trained using the Adam optimizer (Kingma & Ba, 2014) with learning rate 10^{-2} for 10 epochs, taking approximately an hour. Additionally, the softmax regression model take hyperparameters L (sample size in each stochastic draw) and σ^2 (variance of trip time). We chose $L = 20$ and $\sigma^2 = 100$. These values are again chosen to strike a bargain between realism and computation feasibility. The model appears to converge by the second epoch.

[7] https://developers.google.com/maps/documentation/distance-matrix/

7. Results

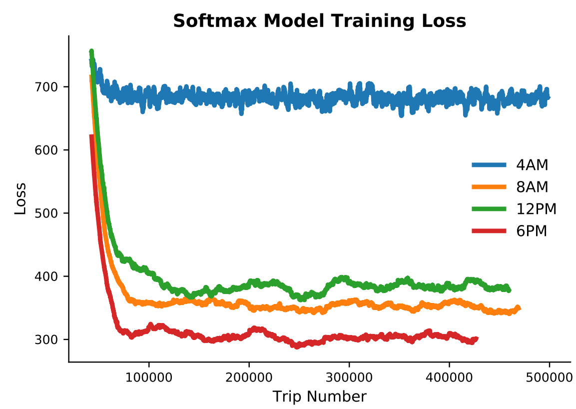


Figure 5. Softmax model training loss, rolling average

We train our model and test the model on an out-of-sample test-set using the methods discussed in Section 5.3. As a simple check of the training process, we plot in Figure 5 the loss obtained when fitting the softmax regression model. We see that the model converges relatively fast.[8] The convergence plot for the uniform model is extremely similar and thus omitted. Table 1 contains the main prediction results of our model relative to some benchmark predictions such as the Google Maps Distance Matrix API, neural network, and linear regression. We note that, in terms of predictive metrics, our models were able to outperform Google Maps and linear regression, but were not able to outperform the neural network, even though the difference is not very large. Despite the disappointing outcome, the advantage of our framework is that the parameters have structural interpretations, whereas the neural network is a black box.

8. Discussion

Table 1 is encouraging. Our models successfully leverage the power of geography and picks up more signal in the data than distance of the trip alone, as demonstrated by the significantly improved performance over linear regression. Perhaps more strikingly, our model slightly outperforms

[8] The full fitting takes approximately 30 minutes per dataset (2.9 GHz Intel Core i5, 8GB RAM), but the model appears to converge in only 20% of that time.

	Baseline			Model	
	Google Maps	Neural Network[2]	Linear Regression[3]	Uniform Model	Softmax Model
$\mathrm{SD}(T)$	7.40	6.85	6.13	6.22	6.22
$\widehat{\mathbb{E}}[\epsilon]$[1]	-0.83	-0.07	-0.05	0.61	0.99
$\mathrm{SD}(\epsilon)$	4.98	3.88	5.01	4.28	4.28
$\widehat{\mathbb{E}}[\lvert\epsilon\rvert]$	3.42	2.44	3.60	2.82	2.86
$\mathrm{Median}(\lvert\epsilon\rvert)$	2.55	1.67	2.93	1.92	1.92
$\mathrm{Percentile}(\lvert\epsilon\rvert, 99)$	16.54	12.79	16.62	14.72	15.24
Test-set[4] R^2	0.54	0.68	0.33	0.53	0.53

Table 1. Prediction results on test set

[1] Here $\epsilon =$ (Actual trip duration) $-$ (Predicted duration) in minutes. $\widehat{\mathbb{E}}$ denotes the sample mean.
[2] We restrict the test-set to 8:00–9:00 AM on Tuesdays in January 2009 for all models *except* the neural network, for which we use an out-of-sample test set on the *entire* range of trip times, since the neural network encodes time-of-day and day-of-week data.
[3] In linear regression, we simply regress T on straight-line distance between the source and the destination.
[4] Test-set R^2 is calculated as $1 - \frac{\mathrm{Var}(\epsilon)}{\mathrm{Var}(T)}$ on the underlying dataset being tested on. Note that the test-set R^2 is an out-of-sample R^2.

the predictions generated by Google Maps, which means that the predictive accuracy of the model is on par with the standard of a widely used consumer product. Even though we were not able to outperform the neural network, we reiterate that the main advantage of our approach is to have interpretable parameters, which are visualized in Figure 6. We turn to the figure in the next section.

8.1. Comparing Models

Figure 6 shows that the parameter values for the uniform and softmax regression models behave similarly, with similar areas of high and low congestion. The patterns seem to largely agree with conventional wisdom, both in spatial and temporal dimensions. We identify that 4:00 AM is indeed less congested than rush hour, that Midtown Manhattan seems to be particularly congested during the day, and that the congested areas have an average speed of 4–6 miles per hour. The visualization also confirms some less-than-well-known features; the areas that demonstrate congestion at 4:00 AM largely confirms with anecdotal evidence.[9] As the models seem to produce largely similar output, we conclude that they are fairly robust.

However, the two models do have significant differences. The softmax regression model appears to have smoother parameter values than the uniform model. This is consistent with the softmax model ascribing higher probability

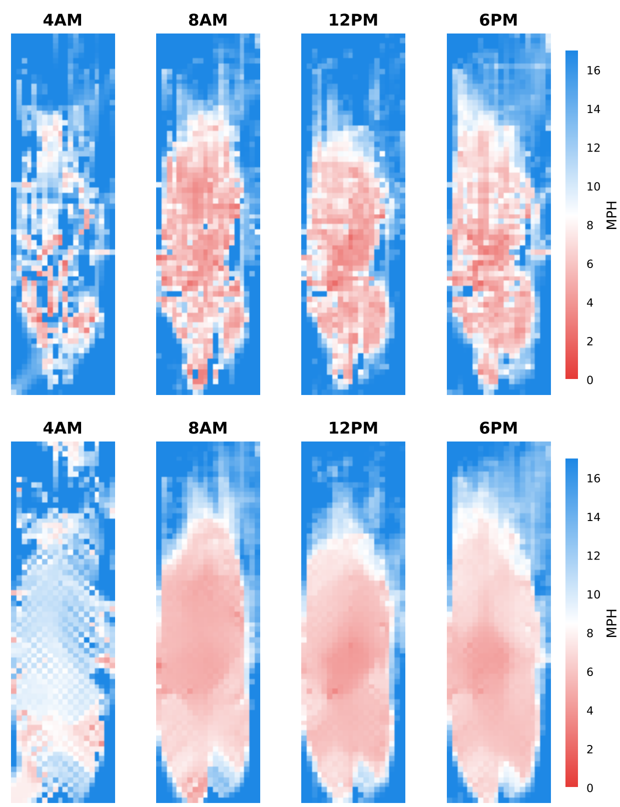


Figure 6. Learned Weights, Various Times Tuesday. Top Row: Uniform Model. Bottom Row: Softmax Model.

[9] According to *Business Insider* (`http://read.bi/164vSK1`), East Village and surrounding areas rank highly for nightlife.

to faster paths, and so we expect the parameter updates to spread out spatially.

It is difficult to select the model to use based on predictive metrics alone, as the models perform strikingly similarly. However, from a practical standpoint, the uniform model is significantly easier to implement and faster to train than the softmax regression model, even though the latter embraces more structural realism.

8.2. Remaining Puzzles

- What conclusions can you draw from the results section?
- Is there further analysis you can do into the results of the system? Here is a good place to include visualizations, graphs, qualitative analysis of your results.

9. Conclusion

References

Blei, David M, Kucukelbir, Alp, and McAuliffe, Jon D. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted), 2017.

Dempster, Arthur P, Laird, Nan M, and Rubin, Donald B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

Herring, Ryan, Hofleitner, Aude, Abbeel, Pieter, and Bayen, Alexandre. Estimating arterial traffic conditions using sparse probe data. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 929–936. IEEE, 2010.

Hunter, Timothy, Herring, Ryan, Abbeel, Pieter, and Bayen, Alexandre. Path and travel time inference from gps probe vehicle data. *NIPS Analyzing Networks and Learning with Graphs*, 12(1), 2009.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

McFadden, Daniel et al. Conditional logit analysis of qualitative choice behavior. 1973.

Van Lint, J, Hoogendoorn, S, and Van Zuylen, H. Freeway travel time prediction with state-space neural networks: modeling state-space dynamics with recurrent neural networks. *Transportation Research Record: Journal of the Transportation Research Board*, (1811):30–39, 2002.

Wu, Chun-Hsin, Ho, Jan-Ming, and Lee, Der-Tsai. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems*, 5(4): 276–281, 2004.

Zhan, Xianyuan, Hasan, Samiul, Ukkusuri, Satish V, and Kamga, Camille. Urban link travel time estimation using large-scale taxi data with partial information. *Transportation Research Part C: Emerging Technologies*, 33: 37–49, 2013.