Final Research Project
Khanh-Tung Nguyen-Ba (kn2465)
December 21, 2019

1 Overview

In the field of reinforcement learning, state aggregation is the problem of mapping the system's states into a small number of meta-states [Singh et al. (1995)]. The choice of aggregation map often depends on the data analysts' knowledge and is largely adhoc. There have been some previous work involving spectral decomposition [Zhang and Wang (2018)] or non-negative matrix factorization [Duan et al. (2018a)] that show promising answers this problem.

In this research project, we apply the Poisson Factorization models proposed by Gopalan et al. (2013) to find the latent structure of a transition matrix built from a set of 9 millions Manhattan traffic trajectories. The goal is to generate a data-driven state aggregation map that is geographically meaningful.

One of the remaining challenges of classical non-negative matrix factorization is determining the positive rank of a matrix - the minimal number of non-negative vectors of which a matrix can be written as a linear combination. There have been attempts from the linear algebra community to solve this problem, for example, borrowing the idea of nuclear norm relaxation in Duan et al. (2018b), the authors used an atomic regularizer as a convex surrogate for the non-negative rank and formulate a convex optimization problem. Taking a probabilistic approach, We assess the possibility of using Non-parametric Poisson Factor model [Gopalan et al. (2014)] to simultaneously find the aggregation map and the optimal size of the aggregate state.

2 The Problem

From the data set of 9 millions NYC Yellow cab trips in January 2016 that includes the pick-up and drop-off locations of each trip, we divide the map into 0.001' x 0.001' grid, group up the pick-up (drop-off) locations that are in the same vicinity and remove cells where there is no pick-up and no drop-off to create a transition grid of size 2020x2020. Assuming that this grid is a stationary Markov transition matrix, we want to find the latent structures that represent the aggregate state together with its aggregate and disaggregate distributions. We expect to see the map to be divided into areas corresponding to the aggregate state.

The transition matrix is sparse, only one third of of the entries are non-zero. It also has a large number of locations with very low activity [Fig.2d]. This makes Poisson Factorization a suitable model because it avoids the down-weighting effect of missing observations often seen in classical methods. In addition, the method only requires iterating over non-zero entries. These advantages are discussed in details in Gopalan et al. (2013).

3 Methods

3.1 The generative process

The generative process for the State Aggregation Hierarchical Poisson Factorization (HPF) is:

- For each pick-up location u in U pick-up locations:
  - Draw activity $\xi_u \sim \text{Gamma}(a_0, a_0/b_0)$
  - Draw $\theta_{uk} \sim \text{Gamma}(a_1, \xi_u)$ for $u = 1, ..., U$ and $k = 1, ...K$
- For each drop-off location d in D drop-off locations:
  - Draw activity $\eta_d \sim \text{Gamma}(c_0, c_0/d_0)$
  - Sample capacity $\beta_{dk} \sim \text{Gamma}(c_1, \eta_d)$ for $d = 1, ..., D$ and $k = 1, ..., K$
- For each trajectory from u to d:
  - Sample the number of occurrences $y_{ud} \sim \text{Poisson}(\sum_{k=1}^{K} \theta_{uk}\beta_{dk})$ for $u = 1, ..., U$ and $d = 1, ..., D$

A simpler model where the rate parameters ξ_u and η_d are fixed for all pick-up locations and all drop-off locations is called Bayesian Poisson Factorization (BPF):

- For each pick-up location u in U pick-up locations:
  - Sample capacity $\theta_{uk} \sim \text{Gamma}(a_1, b_1)$ for $u = 1, ..., U$ and $k = 1, ...K$
- For each drop-off location d in D drop-off locations:
  - Sample capacity $\beta_{dk} \sim \text{Gamma}(c_1, d_1)$ for $d = 1, ..., D$ and $k = 1, ..., K$

We also consider the Non-parametric Poisson Factorization (NPPF) model where the difference from BPF comes from an infinite number of factors and θ_{uk} is drawn from a Gamma process with a size-biased order of weights:

- For each pick-up location u in U pick-up locations:
  - Sample stick $s_u \sim \text{Gamma}(\alpha, c)$
  - Sample $v_{uk} \sim \text{Beta}(1, \alpha)$ for $k = 1, ..., \infty$
  - Set $\theta_{uk} = s_u.v_{uk} \prod_{i=1}^{k-1}(1 - v_{ui})$ for $k = 1, ..., \infty$
- The sampling process for β_{dk} and y_{ud} are similar to BPF, with the exception that k now goes to infinity.

3.2 Inference

First, we use the classic trick that introduces auxiliary latent variables z_{udk} such that $z_{uik} \sim \text{Poisson}(\theta_{uk}\beta_{dk})$, this leads to $y_{ud} = \sum z_{udk}$.

For BPF, we use a mean-field variational family to approximate $p(z, \beta, \theta)$ (for HPF it is $p(z, \beta, \theta, \xi, \eta)$). We use Gamma for the variational distributions, and use coordinate ascent to maximize the ELBO. The conjugate pair Poisson-Gamma yields efficient update to each of β_{uk} and θ_{dk} (and each of ξ_d and η_u for HPF).

For the non-parametric model, in order to handle the infinite number of factors, we use a mean-field variational family with a truncation level T. That way we can have an infinite number of components for the variational distribution, but a finite number of variational pararameters. We use Gamma for the variational distributions of the stick scale s_u and β_{dk}. Updating the stick scales s_u, the drop-off budget β_{uk} and auxillary variables z_{uik} are straightforward thanks to the Gamma-Poisson conjugacy. For each of the stick proportions v_{uk}, we use a degenerate delta distribution, and update it using a MAP estimate.

4 Results

4.1 On simulated data

We simulate a matrix of size 100x100 using BPF generative process with $K = 10, a_1 = b_1 = c_1 = d_1 = 0.3$, and assess the ability of BPF, HPF and NPPF to find the latent structure of the simulated matrix. We are not able to reproduce the observation from Gopalan et al. (2014), our NPPF implementation performs much inferior compared to BPF [Fig.1d and Fig.1b]. Note that we try to mitigate the impact of hyperparameters sensitivity by setting $c_1 = d_1 = 0.3$ for both BPF and NPPF, $b_0 = d_0 = 0.3$ for HPF. We observer that HPF is the best among the three models both in term of performance (quickly converged) and accuracy (higher logjoint) [Fig.1a]. HPF's superior performance is also indicated in the posterior predictive check result [Fig.1c].

For sanity check, we look at the weights v_{uk} of the NPPF model to make sure it follow a size-biased order [Fig.1e]. With the truncation level $T = 15$, the weight mostly fall into the first 5 locations. We remain unsure why NPPF performs much worse than its parametric counterpart, this is left for further investigation.

4.2 On real transition matrix

As HPF perform the best among the three models for simulated data, we continue using the model on real transition matrix. We fix the hyperparameters $a_1 = a_0 = b_1 = c_0 = c_1 = d_1 = 0.3$ and choose the number of factors K from $\{5, 7, 10\}$. We see the level of logjoint at convergence slightly

increase as K gets higher, and the model converges after 80 iterations for all 3 values of K [Fig.2a, Fig.2b, Fig.2c]. The posterior predictive result show that the model successfully capture the pick-up and drop-off budget distributions of the data [Fig.2d].

In the final step, we normalize the disaggregate distribution β_{dk} where $k = 1, ..., K$ and $d = 1, ..., 2020$, and perform K-Means clustering to find the cluster for each drop-off location. The results presented in [Fig. 3] show the clusters of the drop-off locations on the map. We can see that starting from K = 5, the clusters already represent geographically meaningful regions. With K = 7, we can see Upper West Side, Upper East Side, Midtown West, Greenwich and East Village being captured. With K = 10, the map is further partitioned, Harlem is split into East Harlem and West Harlem, and Lower East Side becomes more evident.

5 Code

All relevant code can be found at: https://github.com/kn27/BNPF

6 Conclusion

We successfully apply Hierachical Poisson Factorization model to solve the state aggregate problem in reinforcement learning. As far as we know, this is the first time a probabilistic model is applied to such problem. The algorithm is suitable for the sparse transition matrix and yields a clustered map that is geographically meaningful.

For future work, we will reassess our Non-parametric Poisson Factorization model as the inferior performance remains unexplainable to us.

References

Yaqi Duan, Zheng Tracy Ke, and Mengdi Wang. 2018a. State Aggregation Learning from Markov Transition Data. *ArXiv* abs/1811.02619 (2018).

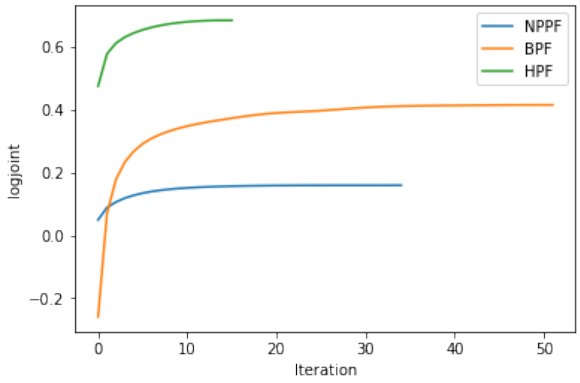
Yaqi Duan, Mengdi Wang, Zaiwen Wen, and Ya-Xiang Yuan. 2018b. Adaptive Low-Nonnegative-Rank Approximation for State Aggregation of Markov Chains.

Prem Gopalan, Jake M. Hofman, and David M. Blei. 2013. Scalable Recommendation with Poisson Factorization. *ArXiv* abs/1311.1704 (2013).

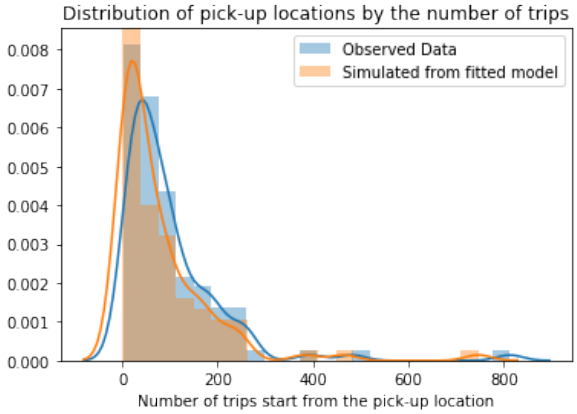
Prem Gopalan, Francisco J.R. Ruiz, Rajesh Ranganath, and David M. Blei. 2014. Bayesian nonparametric poisson factorization for recommendation systems. *Journal of Machine Learning Research* 33 (1 1 2014), 275–283.

Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. 1995. Reinforcement Learning with Soft State Aggregation. In *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.). MIT Press, 361–368. http://papers.nips.cc/paper/981-reinforcement-learning-with-soft-state-aggregation.pdf

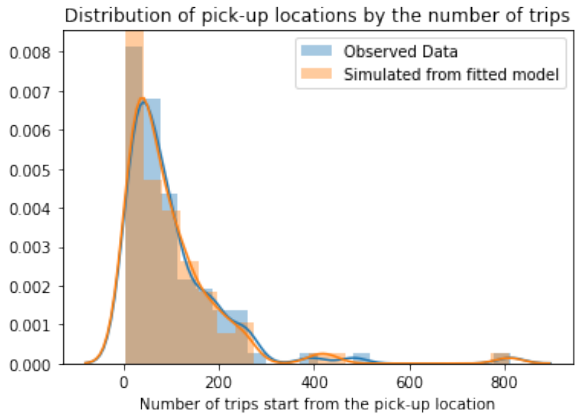
Anru Zhang and Mengdi Wang. 2018. Spectral State Compression of Markov Processes.



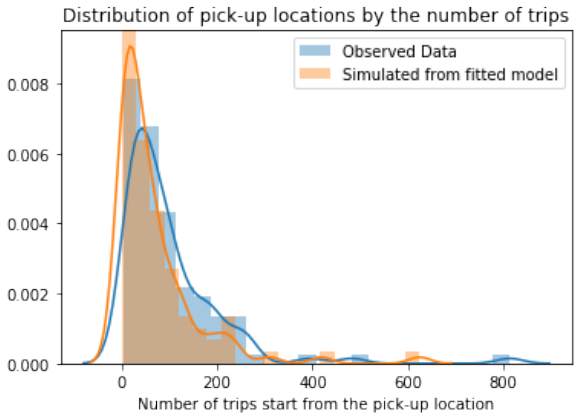
(a) Train logjoint shows NPPF performed worst



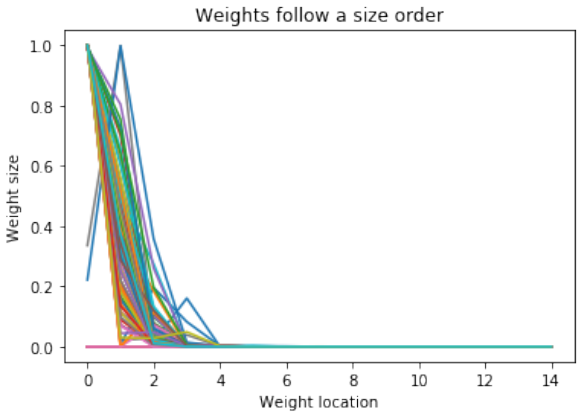
(b) Posterior Predictive Check for BPF



(c) Posterior Predictive Check for HPF

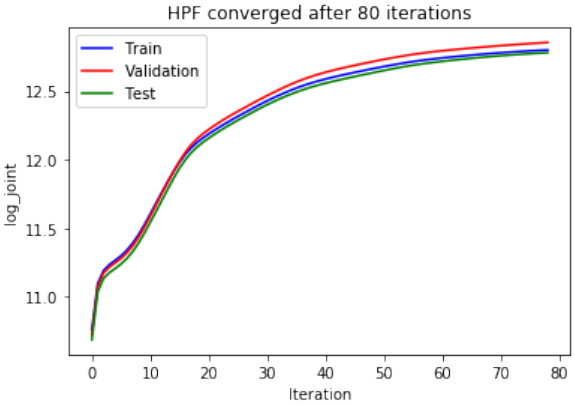


(d) Posterior Predictive Check for NPPF

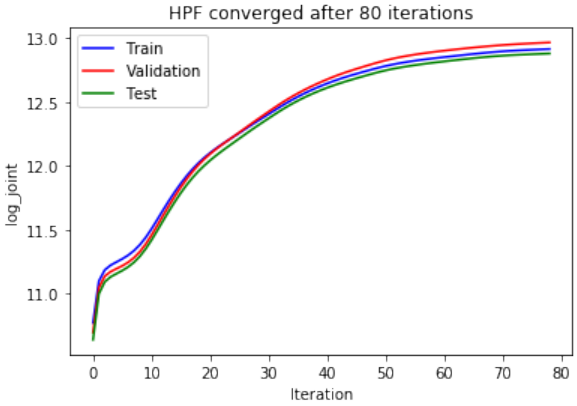


(e) Weights v_{uk} follow a size-biased order

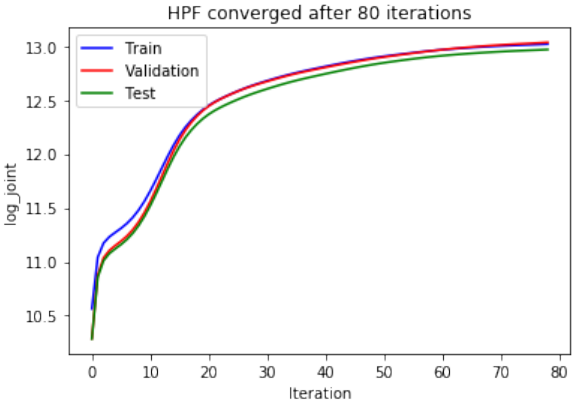
Figure 1: Performance of three models on simulated data



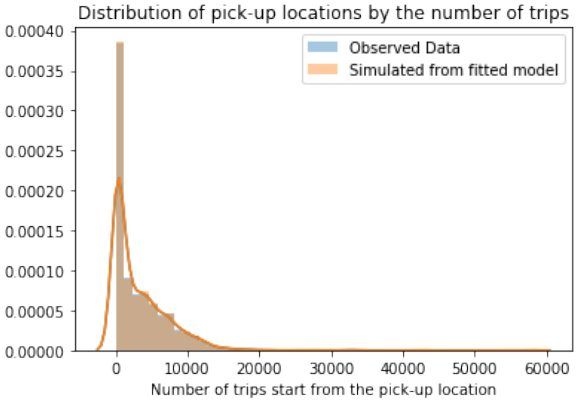
(a) K = 5



(b) K = 7

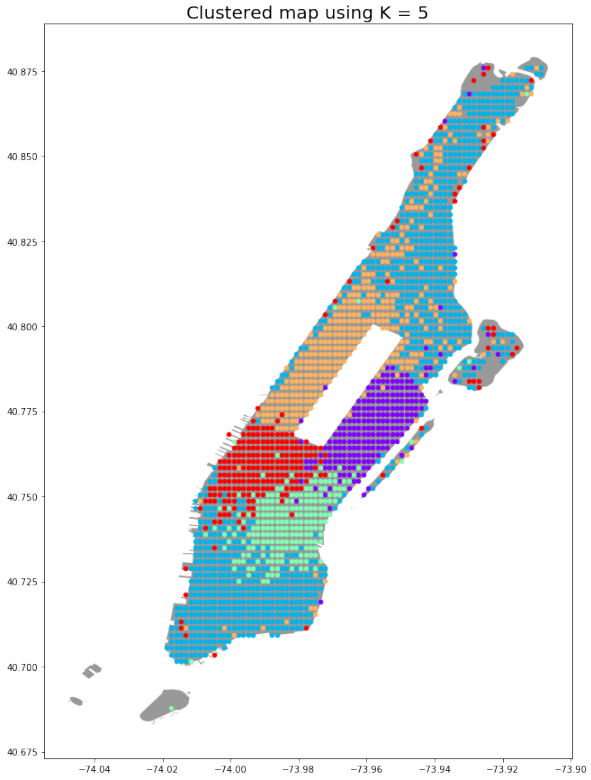


(c) K=10

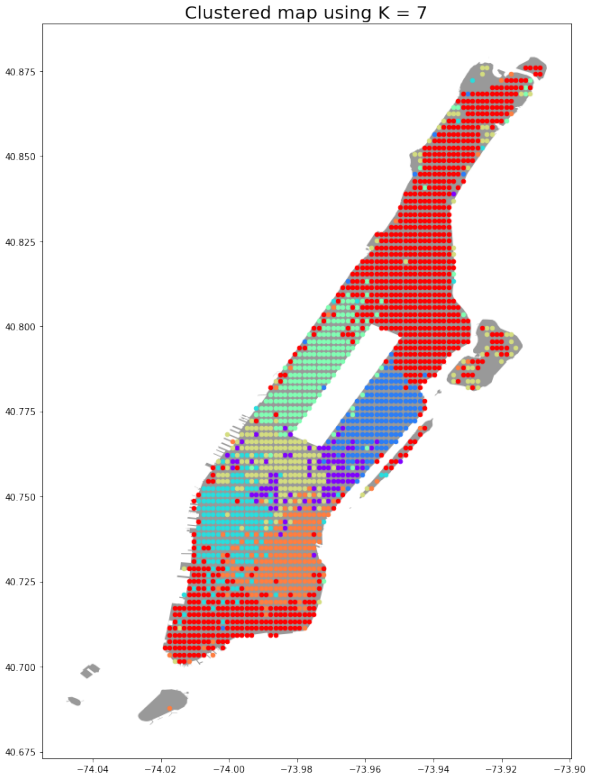


(d) Posterior Predictive Check for HBF for K = 5

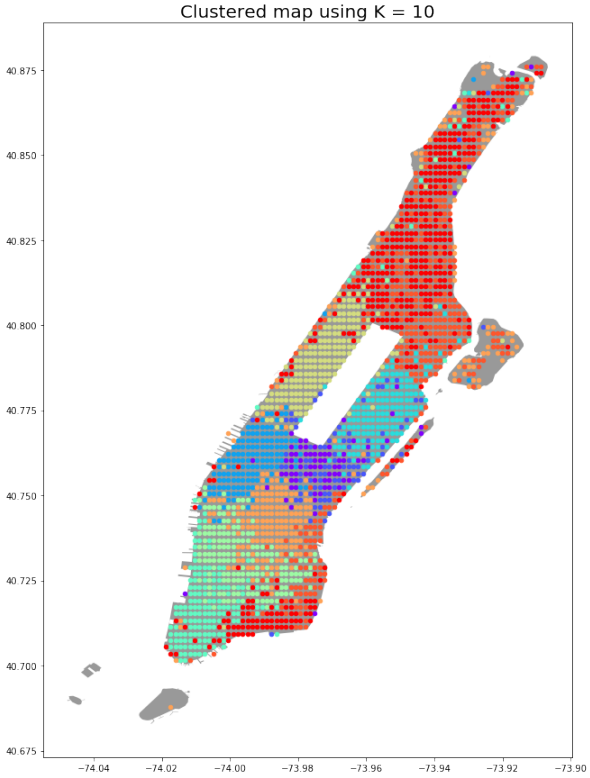
Figure 2: Optimization results for Hierarchical Poisson Factorization



(a) K = 5



(b) K = 7



(c) Clustering K=10

Figure 3: Clustering result of the disaggregate distribution