Online Factorization and Partition of Complex Networks by Random Walk

Lin Yang
Princeton University
lin.yang@princeton.edu

Zheng Yu
Princeton University
zhengy@princeton.edu

Vladimir Braverman
Johns Hopkins University
vova@cs.jhu.edu

Tuo Zhao
Georgia Institute of Technology
tourzhao@gatech.edu

Mengdi Wang
Princeton University
mengdiw@princeton.edu

Abstract

Finding the reduced-dimensional structure is critical to understanding complex networks. Existing approaches such as spectral clustering are applicable only when the full network is explicitly observed. In this paper, we focus on the online factorization and partition of implicit large lumpable networks based on observations from an associated random walk. We formulate this into a nonconvex stochastic factorization problem and propose an efficient and scalable stochastic generalized Hebbian algorithm (GHA). The algorithm is able to process random walk data in a streaming fashion and learn a low-dimensional representation for each vertex. By applying a diffusion approximation analysis, we show that the continuous-time limiting process of the stochastic algorithm converges globally to the "principal components" of the Markov chain. We also establish a finite-sample error bound that matches the nonimprovable state-of-art result for online factorization. Once learned the low-dimensional state representations, we further apply clustering techniques to recover the network partition. We show that when the associated Markov process is lumpable, one can recover the partition exactly with high probability given sufficient data. We apply the proposed approach to model the traffic flow of Manhattan as city-wide random walks. By using our algorithm to analyze the taxi trip data, we discover a latent partition of the Manhattan city that closely matches the traffic dynamics.

1 INTRODUCTION

Network data arise in many applications and research areas, including but not limited to social science, economics, transportation, finance, power grid, artificial intelligence, etc. Examples include protein-protein interaction networks [JS11], phone communication networks [New01], collaboration networks [BGLL08], and the gravitational interaction network of dark matter particles in cosmology [Pee80, Man99, NSMB05]. Due to the highly complex nature of these networks, many efforts have been devoted to investigating their reduced-order representations from high-dimensional data (e.g. [Chu97, Piz08, PBMW99, CY06]).

In this paper, we focus on learning from the dynamic "state-transition" data, which are snapshots of a random walk associated with the implicit network. For example, records of taxi trips can be used to reveal the traffic dynamics of a metropolitan. Each trip can be viewed as a fragmented sample path realized from a city-wide Markov chain that characterizes the traffic dynamics [LKG$^+$12, BGL17]. None of the existing works has considered how to recover the latent network partition of an urban area from the taxi trip data. For another example, reinforcement learning applications such as autonomous driving and game AI are modeled as Markov decision processes [SB98], which unfortunately suffer from the curse of dimensionality of the state space. Given trajectories of game snapshots or a game simulator, it is of vital interest to identify the low-dimensional representation of the "state" of game. For the general problem of finding reduced-order representations, popular approaches such as principal component analysis and spectral clustering do not utilize the Markov nature of state-transition data. Existing computational methods often require explicit knowledge and pre-computation of large matrices, which cannot scale to large-scale problems and is not even possible for online learning applications. Efficient methods are in demand.

Motivated by the need to analyze state-transition data, we propose an efficient and scalable approach for online factorization and partition of implicit complex networks. We start by employing a stochastic gradient-type algorithm, namely the generalized Hebbian algorithm (GHA), and tailor it towards processing Markov transition data. Then we show that the GHA learns low-dimensional representations of the network in an online fashion, and by further applying clustering techniques, we can recover the underlying partition structure with high probability. Our analysis is based on a diffusion approximation approach, which is widely used in stochastic analysis of complicated discrete processes such as queueing networks (see [HKY97] for more related literature on diffusion approximation). By properly rescaling of time, we approximate the discrete-time dynamics generated by the GHA algorithm using its continuous-time limiting process, which is the solution to an ordinary differential equation (ODE). Though the stochastic optimization problem is highly nonconvex, we show that the limiting stochastic process of the GHA converges geometrically to the global optima, even if the initial solution is chosen uniformly at random. We further show that the process after sufficiently large time is well approximated by an Ornstein-Uhlenbeck process, whose stochastic fluctuation can be precisely characterized. Despite of the spherical geometry and many unstable equilibria of the optimization problem, we establish global convergence with a near-optimal sample complexity guarantee in an asymptotic manner. Note that, for GHA, the only theoretical analysis we are aware of is a local convergence of a rank-1 algorithm given by [XLS15]. Thus our global analysis for GHA is of independent interests.

Our work is partly motivated by [WLVE08], which establishes the connection between networks and a class of *lumpable* Markov chains. It proposes an optimization framework to identify the partition structure when the transition matrix is known *a priori*. Our method is also related to the class of online eigenvalue decomposition methods for representation learning [AZL17, LWLZ16, XLS15, AMM16, JJK$^+$16]. However, none of the existing methods and analysis are applicable to Markov transition data and online network partition.

Notation: We denote $[n] = \{1, 2, \ldots, n\}$. Given two matrices $\boldsymbol{U} \in \mathbb{R}^{m\times r_1}, \boldsymbol{V} \in \mathbb{R}^{m\times r_2}$ with orthonormal columns, where $1 \leq r_1 \leq r_2 \leq m$, we denote the principle angle between two matrices by $\Theta(\boldsymbol{U},\boldsymbol{V}) = \operatorname{diag}\big[\cos^{-1}\left(\sigma_1(\boldsymbol{U}^\top\boldsymbol{V})\right), \cos^{-1}\left(\sigma_2(\boldsymbol{U}^\top\boldsymbol{V})\right), \ldots, \cos^{-1}\left(\sigma_{r_1}(\boldsymbol{U}^\top\boldsymbol{V})\right)\big]$, where $\sigma_i(\boldsymbol{A})$ is the i-th largest singular value of matrix $\boldsymbol{A}$. We also use $\cos(\cdot)$ and $\sin(\cdot)$ to act on matrices and denote entry-wise functions. For a matrix $\boldsymbol{V}$, we denote by $\boldsymbol{V}_{*j}$ its j-th column vector and by $\boldsymbol{V}_{i*}$ its i-th row vector. We denote by $\boldsymbol{V}_{*1:r}$ the sub-matrix of the first r columns. We denote by $\|\cdot\|_{\mathrm{F}}$ the Frobenius norm of a matrix, and denote by $\|\cdot\|_2$ the Euclidean norm of a vector or the spectral norm of a matrix. We denote by $\boldsymbol{e}_i \in \mathbb{R}^s$ the i-th standard unit vector for any $s \geq i$: $(\boldsymbol{e}_i)_i = 1$ and $(\boldsymbol{e}_i)_j = 0$ for $j \neq i$. We also denote by $0_{m\times n} \in \mathbb{R}^{m\times n}$ the matrix with all 0 entries.

2 PRELIMINARIES

Let us review the basics of networks and the associated Markov chains.

Networks and Associated Markov Chains: Let $G = (S, E)$ be a *connected* network with m vertices (a weighted directed graph), where $S = \{s_1, s_2, \ldots, s_m\}$ denotes the vertex set, $E = \{w_{i,j} \geq 0 : i, j \in [m]\}$ denotes the edge set, and $w_{i,j}$ denotes the weight of the edge (s_i, s_j).

Consider the random walk that is naturally associated with the network G: We denote by $\boldsymbol{P} = (p_{i,j}) \in \mathbb{R}^{m\times m}$ its probability transition matrix, where each state of the Markov chain corresponds to a vertex in G. Since G is a connected network, all states of the Markov chain are recurrent. The Markov chain generated by the network G satisfies $\mathbb{P}\left[s^{(t)} = s_j \middle| s^{(t-1)} = s_i\right] = p_{i,j}$. Suppose that G is *undirected* (i.e., $w_{ij} = w_{ji}$), then $\forall i, j : p_{i,j} = \frac{w_{i,j}}{w_i}$ and $w_i = \sum_{j\in[m]} w_{i,j}$. The stationary distribution of the Markov chain is $\mu_i = \frac{w_i}{\sum_{j\in[m]} w_j}$. The corresponding Markov chain is *reversible* and satisfies the following *detailed balance condition*

$$\forall i \neq j,\ \mu_i p_{i,j} = \mu_j p_{j,i} \text{ and } \textstyle\sum_{i\in[m]} \mu_i p_{i,j} = \mu_j, \quad (1)$$

i.e., $\boldsymbol{DP} = \boldsymbol{PD}$, where $\boldsymbol{D} = \operatorname{diag}(\mu_1, \mu_2, \ldots, \mu_m)$. Note that our subsequent analysis does not require the undirectedness assumption of the underlying network. In this paper, we focus on connected and undirected networks where $\mu_i > 0$ for all $i \in [m]$. For a non-connected network, our method still applies with the caveat that it recovers the structure of a connected component determined by the initial state.

Our Problem of Interest Given a sample trajectory $\{s^{(0)}, s^{(1)}, \ldots, s^{(t)}, \ldots\}$ of state transitions of the unknown Markov chain, our objective is to develop an online learning method to extract reduced-order information about the Markov chain and recover the latent network partition.

We are interested in complex networks that can be approximated using reduced-order representations. To be general, we consider networks with associated Markov chains having a *spectral gap* or being *nearly low-rank*, which is defined as follows:

Definition 1 (Nearly Low-Rank Markov Chains)**.** *A Markov chain with transition matrix* $\boldsymbol{P}$ *is nearly low-rank if there exist matrices* $\boldsymbol{F}_1, \boldsymbol{F}_2 \in \mathbb{R}^{m\times m}$*, where* $\operatorname{rank}(\boldsymbol{F}_1) = r$ *and* $\|\boldsymbol{F}_2\|_2 < \sigma_r(\boldsymbol{F}_1)$ *such that*

$$\boldsymbol{DP} = \boldsymbol{F}_1 + \boldsymbol{F}_2 \quad \textit{and} \quad \boldsymbol{F}_1^\top \boldsymbol{F}_2 = 0_{m\times m}, \qquad (2)$$

and $\boldsymbol{F}_1 = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, *where* $\boldsymbol{\Sigma} = \mathrm{diag}\left(\sigma_1, \sigma_2, \dots, \sigma_r\right)$ *is a diagonal matrix with* $1 \geq \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$, *and* $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{m\times r}$ *are matrices with orthonormal columns.*

Consider the following *representation matrix*

$$\boldsymbol{M} := \boldsymbol{D}^{-1}\boldsymbol{V} \in \mathbb{R}^{m\times r}, \tag{3}$$

each row of which can be viewed as an r-dimensional representation of a vertex of G. The matrix $\boldsymbol{M}$ gives a set of approximate "principal components" of the Markov chain, which has a similar spirit as spectral clustering [Chu97]. Note that Markov chains that are nearly low-rank are not necessarily reversible. When a Markov chain is both nearly low-rank and reversible, the conditions in Definition 1 shall hold with $\boldsymbol{U} = \boldsymbol{V}$.

In particular, we also consider an important special case of nearly low-rank Markov chains - "lumpable" Markov chains, which is introduced by [MS01] and formally in [WLVE08] as follows.

Definition 2 (Special Case: Lumpable Markov chains [WLVE08])**.** *A reversible Markov chain on states S with transition matrix $\boldsymbol{P}$ is* lumpable *with respect to the partition $S = S_1 \cup S_2 \dots \cup S_r$ if the top r eigenvectors of $\boldsymbol{DP}$ are piecewise constant with respect to the $S_1, \dots, S_r$.*

We can view S_1,...,S_r as "meta states" of the Markov chain. When the lumpability condition holds, the transitions between these sets satisfy the strong Markov property, i.e., for arbitrary $s_k, s_h \in S_i$, and arbitrary j, $\sum_{s_\ell \in S_j} p_{k,\ell} = \sum_{s_\ell \in S_j} p_{h,\ell}$ always holds. Intuitively speaking, the meta states suffice to characterize the macro dynamics of a complex Markov chain. When the Markov chain is lumpable, it is nearly-low rank as in Definition 1 with $\boldsymbol{U} = \boldsymbol{V}$. In this case, the matrix $\boldsymbol{U}$ becomes a block matrix. For any $i, j \in [r]$, the vector $\boldsymbol{U}_{*i}$ restricted on coordinates S_j has constant values across all entries. The work [WLVE08] showed when the Markov chain is lumpable with respect to a partition $S = S_1 \cup S_2 \dots \cup S_r$, one can recover the exact partition by clustering its r-dimensional representations (rows of $\boldsymbol{M} = \boldsymbol{D}^{-1}\boldsymbol{V}$).

3 METHOD

Recall that we are interested in learning from Markov transition data. In particular, consider the scenario where we only observe state-to-state transitions of a Markov process over S: $s^{(1)}, s^{(2)}, s^{(3)}, \dots, s^{(n-1)}, s^{(n)}, \dots$, without knowing the transition matrix $\boldsymbol{P}$ in advance. For notational convenience, we simplify the notation of the states to $S = \{1, 2, \dots m\}$.

3.1 NONCONVEX OPTIMIZATION FOR MARKOV CHAIN FACTORIZATION

To handle the dependency of the Markov process, we need to downsample the data. Specifically, we divide the trajectory of n state transitions into b blocks with block size τ for some $\tau \geq 2$:

$$\underbrace{s^{(1)}, s^{(2)}, ..., s^{(\tau)}}_{\text{the 1-st block}}, \underbrace{s^{(\tau+1)}, s^{(\tau+2)}, ..., s^{(2\tau)}}_{\text{the 2-nd block}}, \dots,$$
$$\underbrace{s^{(b-1)\tau+1}, s^{(b-1)\tau+2}, \dots s^{(b\tau)}}_{\text{the b-th block}}.$$

For the k-th block, we select the last two samples and construct $\boldsymbol{Z}^{(k)} \in \mathbb{R}^{m\times m}$ to be the matrix with one entry equaling 1 and all other entries equaling 0, i.e.,

$$\boldsymbol{Z}^{(k)}_{s^{(k\tau-1)}, s^{(k\tau)}} = 1 \quad \text{and} \quad \boldsymbol{Z}^{(k)}_{s,s'} = 0$$
$$\text{for all } (s, s') \neq \left(s^{(k\tau-1)}, s^{(k\tau)}\right). \tag{4}$$

Here we choose a large enough τ such that $\forall k \geq 1$, $\mathbb{E}\left[\boldsymbol{Z}^{(k)} \middle| s^{(0)}\right] \approx \boldsymbol{DP} = \boldsymbol{F}_1 + \boldsymbol{F}_2$, where $\boldsymbol{F}_1 = \boldsymbol{U}^\top\boldsymbol{\Sigma}\boldsymbol{V}$ and $\boldsymbol{F}_2$ are given in Definition 1. Intuitively, the choice of τ shall be related to how fast the Markov chain mixes. We will specify the choice of τ in Section 4.

Let us formulate the Stochastic Transition Matrix Decomposition Problem as

$$(\boldsymbol{U}^*, \boldsymbol{V}^*) = \underset{\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{V}} \in \mathbb{R}^{m\times r}}{\mathrm{argmax}} \ \mathrm{tr}\left[\widetilde{\boldsymbol{U}}^\top \mathbb{E}\boldsymbol{Z}\widetilde{\boldsymbol{V}}\right]$$
$$\text{subject to} \quad \widetilde{\boldsymbol{U}}^\top\widetilde{\boldsymbol{U}} = \widetilde{\boldsymbol{V}}^\top\widetilde{\boldsymbol{V}} = I_r, \tag{5}$$

where the expectation

$$\mathbb{E}\boldsymbol{Z} := \lim_{n\to\infty} n^{-1}\sum_{k=1}^{n}\boldsymbol{Z}^{(k)} = \boldsymbol{DP}$$

is taken over the invariant distribution of the Markov chain. Note that global optima $\boldsymbol{U}^*$ and $\boldsymbol{V}^*$ are not identifiable in (5). For instance, for any orthonormal matrix $\boldsymbol{O} \in \mathbb{R}^{r\times r}$, $\boldsymbol{U}^* = \boldsymbol{UO}$ and $\boldsymbol{V}^* = \boldsymbol{VO}$ are still global optimal. Our goal is to show that our algorithm is able to find at least one solution. By using a self-adjoint dilation, we recast (5) into a symmetric decomposition problem as follows

$$\boldsymbol{W}^* = \underset{\boldsymbol{W} \in \mathbb{R}^{2m\times r}}{\mathrm{argmax}} \ \mathrm{tr}\left[\boldsymbol{W}^\top \mathbb{E}\boldsymbol{A}\boldsymbol{W}\right] \quad \text{subject to} \quad \boldsymbol{W}^\top\boldsymbol{W} = I_r, \tag{6}$$

where $\mathbb{E}\boldsymbol{A} = \begin{bmatrix} 0_{m\times m} & \mathbb{E}\boldsymbol{Z} \\ \mathbb{E}\boldsymbol{Z}^\top & 0_{m\times m} \end{bmatrix} \in \mathbb{R}^{2m\times 2m}$ and $\boldsymbol{W} = \frac{1}{\sqrt{2}}\left[\boldsymbol{U}^\top, \boldsymbol{V}^\top\right]^\top \in \mathbb{R}^{2m\times r}$.

3.2 ALGORITHM FOR ONLINE FACTORIZATION OF MARKOV CHAINS

To solve (6), we adopt the Generalized Hebbian Algorithm (GHA) which was originally developed as a *heuristic* for training neural nets and principal component analysis [San89]. GHA, also referred as Sanger's rule, is essentially a stochastic primal-dual algorithm. Specifically, let $\mathcal{L}\left(\boldsymbol{W}, \boldsymbol{L}\right)$ be the Lagrangian function of Eq. (6) given by

$$\mathcal{L}\left(\boldsymbol{W}, \boldsymbol{L}\right) = \operatorname{tr}\left[\boldsymbol{W}^\top \mathbb{E}\boldsymbol{A}\boldsymbol{W}\right] - \operatorname{tr}\left[\boldsymbol{L}\left(\boldsymbol{W}^\top\boldsymbol{W} - \boldsymbol{I}_r\right)\right],$$

where $\boldsymbol{L} \in \mathbb{R}^{r\times r}$ is the Lagrangian multiplier matrix. By checking the Karush-Kuhn-Tucker (KKT) conditions of the problem $\max_{\boldsymbol{W}} \min_{\boldsymbol{L}} \mathcal{L}\left(\boldsymbol{W}, \boldsymbol{L}\right)$, we obtain

$$\mathbb{E}\boldsymbol{A}\boldsymbol{W}^* + \boldsymbol{W}^*\boldsymbol{L}^* = 0 \text{ and } \boldsymbol{W}^{*\top}\boldsymbol{W}^* - \boldsymbol{I}_r = 0, \quad (7)$$

where $\boldsymbol{L}^*$ is the optimal Lagrangian multiplier. The above KKT conditions further imply

$$\boldsymbol{L}^* = -\boldsymbol{W}^{*\top}\mathbb{E}\boldsymbol{A}\boldsymbol{W}^*. \quad (8)$$

GHA is essentially a stochastic approximation method for the solving the equations (7) and (8). Specifically, we use the k-th block of transition data to compute the sample matrix

$$\boldsymbol{A}^{(k)} = \left[\begin{array}{cc} 0_{m\times m} & \boldsymbol{Z}^{(k)} \\ \boldsymbol{Z}^{(k)\top} & 0_{m\times m} \end{array}\right] \in \mathbb{R}^{2m\times 2m}. \quad (9)$$

Then the k-th iteration of GHA takes the form

$$\begin{aligned} &\text{Dual Update :} \\ &\boldsymbol{L}^{(k)} = \underbrace{\boldsymbol{W}^{(k)\top}\boldsymbol{A}^{(k+1)}\boldsymbol{W}^{(k)}}_{\text{Markov sample of } \boldsymbol{W}^{(k)\top}\mathbb{E}\boldsymbol{A}\boldsymbol{W}^{(k)}} \quad (10) \\ &\text{Primal Update :} \\ &\boldsymbol{W}^{(k+1)} = \boldsymbol{W}^{(k)} + \eta\underbrace{\left(\boldsymbol{A}^{(k+1)}\boldsymbol{W}^{(k)} - \boldsymbol{W}^{(k)}\boldsymbol{L}^{(k)}\right)}_{\text{Markov sample of } \nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}^{(k)},\Lambda^{(k)})} \quad (11) \end{aligned}$$

where $\eta > 0$ is the learning rate. Combing (10) with (11), we get a dual-free update of GHA as follows,

$$\boldsymbol{W}^{(k+1)} = \boldsymbol{W}^{(k)} + \eta(\boldsymbol{A}^{(k)}\boldsymbol{W}^{(k)} - \boldsymbol{W}^{(k)}\boldsymbol{W}^{(k)\top}\boldsymbol{A}^{(k)}\boldsymbol{W}^{(k)}).$$

Note that the columns of $\boldsymbol{W}^{(k)}$ are not necessarily orthogonal. But when $\boldsymbol{W}^{(0)}$ has orthonormal columns, then $\boldsymbol{W}^{(k)}$ tends to have orthonormal columns as $\eta \to 0$. The formal procedure is presented in Algorithm 1.

Algorithm 1 GHA for Online Factorization of Markov Chains

```
Input:  A stream of Markov transition data $s^{(1)}, s^{(2)}, s^{(3)}, \dots, s^{(n-1)}, s^{(n)}, \dots$
Initialize:
    Sample matrix $G \in \mathbb{R}^{2m\times r}$ with i.i.d. entries from $\mathcal{N}(0,1)$;
    $\boldsymbol{W}^{(0)} \leftarrow QR(G)$, $k \leftarrow 0$;
Repeat:
For every $\tau$ state transitions, obtain $\boldsymbol{A}^{(k)}$ using Eqs. (4),(9);
    $\boldsymbol{W}^{(k+1)} \leftarrow$
    $\boldsymbol{W}^{(k)} + \eta\left[\boldsymbol{A}^{(k+1)}\boldsymbol{W}^{(k)} - \boldsymbol{W}^{(k)}\boldsymbol{W}^{(k)\top}\boldsymbol{A}^{(k+1)}\boldsymbol{W}^{(k)}\right]$;
    $k \leftarrow k+1$;
Until stopping condition is satisfied
Output $[\widehat{\boldsymbol{U}}; \widehat{\boldsymbol{V}}] \leftarrow \sqrt{2}\boldsymbol{W}^{(k)}$
```

Algorithm 1 is a globally convergent method which does not require any warm-up initialization or prior knowledge. The initial solution $\boldsymbol{W}^{(0)}$ is drawn uniformly from the set of all orthonormal matrices by applying a QR decomposition to a matrix with i.i.d. Gaussian entries. Algorithm 1 makes update online and uses $O(mr)$ space, while a batch method needs $O(m^2)$ space to store the explicit transition matrix.

3.3 RECOVERING NETWORK PARTITION FROM RANDOM WALKS

Recall that in Definition 1 the $m \times r$ matrix $\boldsymbol{M} = \boldsymbol{D}^{-1}\boldsymbol{V}$ gives a reduced-order representation for each vertex of the network. As long as we can estimate $\boldsymbol{D}, \boldsymbol{V}$, we would be able to partition the network by applying a clustering algorithm such as the k-means. We describe the overall procedure:

Algorithm 2 Recovering The Network Partition from Random Walks

```
1: Run Algorithm 1 on the Markov transition data and obtain $[\widehat{\boldsymbol{U}}; \widehat{\boldsymbol{V}}]$.
2: Let $\widehat{\boldsymbol{\mu}}$ be the empirical estimate of the stationary distribution, i.e., $\widehat{\mu}_i = \sum_{k=1}^n \mathbb{I}(s^{(k)} = i)/n$. Let $\widehat{\boldsymbol{D}} = \operatorname{diag}(\widehat{\mu}_1, \widehat{\mu}_2, \dots, \widehat{\mu}_m)$. Now each row of $\widehat{\boldsymbol{M}} = \widehat{\boldsymbol{D}}^{-1}\widehat{\boldsymbol{V}}$ gives an approximate $r$-dimensional representation for the corresponding state/vertex.
3: Find a set of centers $C = \{c_1, c_2, \dots, c_r\} \subset \mathbb{R}^r$ by solving the following problem:
       $\widehat{C} = \operatorname{argmin}_C \sum_{i=1}^m \min_{c\in C} d^2(\widehat{\boldsymbol{M}}_{i*}, c)$,     (12)
   where $d(\widehat{\boldsymbol{M}}_{s_i*}, c_j) = ||\widehat{\boldsymbol{M}}_{s_i*} - c_j||_2$ is the Euclidean distance.
4: Output the partition by assigning each state to its closest center.
```

4 THEORY

We analyze the convergence of Algorithm 1 for solving the nonconvex factorization problem. We use the idea of diffusion approximation (e.g. [HKY97]). (1) We show that the dynamics of our algorithm can be approximated by an ordinary differential equation (ODE); (2) To analyze the convergence rate, we show that after proper rescaling of time, the algorithm's dynamics can be characterized by the solution of a Stochastic Difference Equation (SDE). The SDE allows us to analyze the error fluctuation when the iterates are within a small neighborhood of the global optimum.

4.1 REDUCING DEPENDENCY BY DOWN SAMPLING

Recall our goal is to estimate Markov chain factorization from random walks. In the online learning setting, the data comes in a stream and are highly dependent. To handle the dependency, our algorithm replies on down sampling the data points. Next we introduce some important measures for a Markov chain. For notational convenience, we denote $\mu(\Omega) = \sum_{i\in\Omega} \mu_i$ for any subset of states $\Omega \subset S$. We introduce the *merging conductance* [Mih89] of a Markov chain by

$$\Phi = \min_{\Omega\subset[m]} \frac{\sum_{j\in\Omega,\ell\in\Omega^c} \sum_{i\in[m]} \frac{\mu_j p_{j,i} \mu_\ell p_{\ell,i}}{\mu_i}}{\sum_{j\in\Omega} \mu_j}$$
$$\text{subject to} \quad \mu(\Omega) \le 1/2,$$

where Ω^c is the complement of Ω. The parameter Φ is a generalization of the Cheeger's constant, which characterizes the bottleneck of a network. For recurrent Markov chains that are rapidly mixing, Φ can be treated as a constant. Besides, we let

$$\mu_{\max} = \max_{i\in[m]} \mu_i, \mu_{\min} = \min_{i\in[m]} \mu_i.$$

We then choose block length τ in Algorithm 1 as follows:

$$\tau \ge \left[\frac{2}{\Phi^2} \log \left(\sqrt{\frac{\mu_{\max}}{\mu_{\min}}} \frac{1}{\eta} \right) \right].$$

As shown in the full version[YBZW17], by choosing such a down-sampling block length, our data samples are sufficiently close to i.i.d. samples drawn from the stationary distribution of the underlying Markov chain. This allows us to approximate algorithm by another auxiliary procedure of fully independent samples. In the next two sub-sections, we show the limiting procedure of the algorithm based on this down-sampling rate.

4.2 ODE CHARACTERIZATION OF ALGORITHM 1

Let $\boldsymbol{R} \in \mathbb{R}^{2m\times 2m}$ be the matrix of eigenvectors of $\mathbb{E}\boldsymbol{A}$ in (6). We consider a transformation by $\boldsymbol{R}$:

$$\overline{\boldsymbol{W}}^{(k)} = \boldsymbol{R}^\top \boldsymbol{W}^{(k)} \quad \text{and} \quad \overline{\boldsymbol{A}}^{(k)} = \boldsymbol{R}^\top \boldsymbol{A}^{(k)} \boldsymbol{R}.$$

Let $\boldsymbol{\Lambda} = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{2m}) = \mathbb{E}\overline{\boldsymbol{A}}$ with that $\sigma_1 \ge \sigma_2 \ge \ldots \sigma_{2m}$. To demonstrate an ODE characterization for the trajectory of the algorithm, we introduce a continuous time t. Recall where η is the learning rate. We denote $\overline{\boldsymbol{W}}(t) = \overline{\boldsymbol{W}}^{(\lfloor t/\eta \rfloor)}$. For notation simplicity, we may drop (t) if it is clear from the context. For $r+1 \le i \le 2m$, we define the cosine subspace angle as

$$\gamma_i^{(\eta)}(t) = \left\| \boldsymbol{e}_i^\top \boldsymbol{R}^\top \boldsymbol{W} \right\|_2 = \left\| \boldsymbol{e}_i^\top \overline{\boldsymbol{W}} \right\|_2,$$

where $\boldsymbol{e}_i \in \mathbb{R}^{2m}$ is the i-th standard unit vector. We use (η) as a superscript to emphasize the dependence on η. To show a global convergence of $\gamma_i^{(\eta)}(t)$, we characterize its upper bound in the following lemma.

Lemma 1 (Principle Angle Upper Bound). *Let $\boldsymbol{E} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_r) \in \mathbb{R}^{2m\times r}$. Suppose that $\boldsymbol{W}$ has orthonormal columns and $\boldsymbol{E}^\top\boldsymbol{W}$ is full rank. For any $\boldsymbol{X} \in \mathbb{R}^{2m\times s}$ with $s \ge 1$, we have $\left\|\boldsymbol{X}^\top\boldsymbol{W}\right\|_{\mathrm{F}} \le \left\| \boldsymbol{X}^\top \boldsymbol{W} \cdot \left(\boldsymbol{E}^\top\boldsymbol{W}\right)^{-1} \right\|_{\mathrm{F}}$.*

Accordingly, we define

$$\widetilde{\gamma}_i^{(\eta)} = \left\| \boldsymbol{e}_i^\top \overline{\boldsymbol{W}} \cdot \left(\boldsymbol{E}^\top \overline{\boldsymbol{W}} \right)^{-1} \right\|_2.$$

Since $\boldsymbol{W}^{(0)}$ has orthonormal columns, for any fixed $t > 0$, the columns of $\overline{\boldsymbol{W}}(t)$ are orthonormal almost surely as $\eta \to 0$. Thus $\widetilde{\gamma}_i^{(\eta)}$ becomes a uniform upper bound of $\gamma_i^{(\eta)}$ almost surely as $\eta \to 0$. The next theorem establishes the continuous time limit for $\widetilde{\gamma}_i^{(\eta)}$.

Theorem 1 (ODE Convergence). *Given $\overline{\boldsymbol{W}}^{(0)}$ with orthonormal columns and that $\boldsymbol{E}^\top\overline{\boldsymbol{W}}^{(0)}$ is invertible, for all $r < i \le 2m$, $\widetilde{\gamma}_i^{(\eta)}(t)$ converges weakly to the solution of the following ODE,*

$$d\widetilde{\gamma}_i^2(t)/dt = b_i \widetilde{\gamma}_i^2(t)$$

as $\eta \to 0$, where b_i is some constant satisfying $b_i \le 2(\sigma_i - \sigma_r)$.

Theorem 1 suggests the global convergence of the algorithm. Specifically, the solution to the above ODE is,

$$\widetilde{\gamma}_i(t) = \widetilde{\gamma}_i(0) e^{b_i t/2} \le \widetilde{\gamma}_i(0) e^{(\sigma_i - \sigma_r)t}, \quad \forall\, r < i \le 2m,$$

which implies $\gamma_i^{(\eta)}(t) \to 0$ for any $r < i \le 2m$ as $\eta \to 0$ and $t \to \infty$. Since $\left\|\sin\Theta\left(\boldsymbol{E}, \overline{\boldsymbol{W}}(t)\right)\right\|_{\mathrm{F}}^2 = \sum_{i>r} \gamma_i^{(\eta)2}(t)$, we obtain

$$\begin{aligned} \left\|\sin\Theta\left(\widehat{\boldsymbol{U}}(t), \overline{\boldsymbol{U}}\right)\right\|_{\mathrm{F}}^2 + \left\|\sin\Theta\left(\widehat{\boldsymbol{V}}(t), \overline{\boldsymbol{V}}\right)\right\|_{\mathrm{F}}^2 & \\ \le 2\left\|\sin\Theta\left(\boldsymbol{E}, \overline{\boldsymbol{W}}(t)\right)\right\|_{\mathrm{F}}^2 & \\ \le 2\sum_{i>r} \widetilde{\gamma}_i(0) e^{(\sigma_{r+1}-\sigma_r)t} \to 0. & \quad (13) \end{aligned}$$

4.3 SDE CHARACTERIZATION OF ALGORITHM 1

Our ODE approximation of the algorithm shows that after sufficiently many iterations with sufficiently small η, the algorithm solution can be arbitrarily close to the true subspace, $\operatorname{span}(\overline{\boldsymbol{R}}_{*1}, \overline{\boldsymbol{R}}_{*2}, \ldots, \overline{\boldsymbol{R}}_{*r})$. To obtain the "rate of convergence", however, we need to study the variance of the trajectory at time t. Note that such a variance is of order $\mathcal{O}(\eta)$, and vanishing under the limit of $\eta \to 0$. To characterize the variance, we need to rescale the updates by a factor of $\eta^{-1/2}$, i.e., after rescaling, the variance is of order $\mathcal{O}(1)$. Specifically, the rescaled update is defined as

$$\zeta_i^{(\eta)}(t) = \eta^{-1/2} \cdot \overline{\boldsymbol{W}}^{(\lfloor t/\eta \rfloor)\top} \cdot \boldsymbol{e}_i \in \mathbb{R}^r.$$

Note that given $\boldsymbol{W}^{(k)}$ such that $\operatorname{span}(\boldsymbol{W}^{(k)}) = \operatorname{span}(\overline{\boldsymbol{R}}_{*1}, \overline{\boldsymbol{R}}_{*2}, \ldots, \overline{\boldsymbol{R}}_{*r})$, we have

$$\mathbb{E}\left(\boldsymbol{W}^{(k+1)} | \boldsymbol{W}^{(k)}\right) = \boldsymbol{W}^{(k)}.$$

We consider a regime, where the algorithm has already run for sufficient many iterations such that

$$\|\sin\Theta\left(\boldsymbol{R}_{*1:r}, \boldsymbol{W}^{(N_1)}\right)\|_{\mathrm{F}}^2 = \|\sin\Theta(\boldsymbol{E}, \overline{\boldsymbol{W}}^{(N_1)})\|_{\mathrm{F}}^2 \le \eta^{1/c},$$

for some constant $c > 1$. By restarting the counter, we denote $\overline{\boldsymbol{W}}^{(0)} := \overline{\boldsymbol{R}}^{\top} \boldsymbol{W}^{(N_1)}$. Now we define $\overline{\boldsymbol{W}}^{(0)\top} \boldsymbol{\Lambda} \overline{\boldsymbol{W}}^{(0)} = \boldsymbol{\Gamma}^{\top} \cdot \widetilde{\boldsymbol{\Lambda}}_r \cdot \boldsymbol{\Gamma}$, where $\boldsymbol{\Gamma} \in \mathbb{R}^{r \times r}$ is an orthonormal matrix and $\widetilde{\boldsymbol{\Lambda}}_r = \operatorname{diag}(\sigma'_1, \sigma'_2, \ldots, \sigma'_r)$, with $\sigma'_1 \geq \sigma'_2 \geq \ldots \geq \sigma'_r \geq 0$.

Denote $\zeta_{i,j}^{(\eta)}(t) = \eta^{-1/2} \left(\boldsymbol{e}_j'^{\top} \boldsymbol{\Gamma} \cdot \overline{\boldsymbol{W}}^{(\lfloor t/\eta \rfloor)\top} \cdot \boldsymbol{e}_i \right)$, for $i = r+1, r+2, \ldots, 2m$ and $j = 1, 2, \ldots, r$, where $\boldsymbol{e}'_j \in \mathbb{R}^r$ denotes the j-th standard unit vector in $\mathbb{R}^r$. We establish the following theorem.

Theorem 2 (SDE Convergence)**.** *Given* $\left\| \sin \Theta \left(\boldsymbol{E}, \overline{\boldsymbol{W}}^{(\lfloor t/\eta \rfloor)} \right) \right\|_{\mathrm{F}}^2 \leq \mathcal{O}(\eta^{1/c})$ *for all* $t \geq 0$, *then for any* $i > r$ *and* $j \in [r]$, *the trajectory of* $\zeta_{i,j}^{(\eta)}(t)$ *weakly converges to the solution of the following SDE, as* $\eta \to 0$,

$$d\zeta_{i,j} = K_{i,j} \zeta_{i,j} dt + G_{i,j} d\mathcal{B}_{i,j} \tag{14}$$

where $\mathcal{B}_{i,j}$ *is the standard Brownian motion (not necessarily i.i.d. across* i, j*) and constants* $K_{i,j} \leq (\sigma_i - \sigma_r)$, $\sum_{i>r}^{2m} G_{i,j}^2 \leq B$ for any $j \in [r]$, *with some absolute constant* B.

Notice that (14) is a Fokker-Plank equation, which admits the following solution,

$$\begin{aligned} \zeta_{i,j}(t) &= \zeta_{i,j}(0) \exp\left[K_{i,j} t\right] \\ &\quad + G_{i,j} \int_0^t \exp\left[K_{i,j}(s-t)\right] d\mathcal{B}_{i,j}(s). \end{aligned} \tag{15}$$

Therefore, we show that each $\zeta_{i,j}^{(\eta)}(t)$ weakly converges to an Ornstein-Uhlenbeck (OU) process, which is widely studied in existing literature [Meu09]. Since the drifting term is driven by $K_{i,j} < 0$, the OU process eventually becomes a pure random walk, i.e., the first term of R.H.S. in (15) goes to 0. Recall that $\zeta_{i,j}^{(\eta)}(t)$ characterizes the sin angle of the subspaces, i.e.,

$$\left\| \sin \Theta \left(\boldsymbol{E}, \overline{\boldsymbol{W}}^{(\lfloor t/\eta \rfloor)} \right) \right\|_{\mathrm{F}}^2 = \eta \textstyle\sum_{i>r}^{2m} \sum_{j=1}^{r} \zeta_{i,j}^{(\eta)2}(t). \tag{16}$$

Thus the fluctuation of $\zeta_{i,j}^{(\eta)}(t)$ is essentially the error fluctuation of the algorithm after sufficiently many iterations. By (15), we obtain

$$\begin{aligned} &\mathbb{E} \| \sin \Theta \left(\boldsymbol{E}, \overline{\boldsymbol{W}}^{(\lfloor t/\eta \rfloor)} \right) \|_{\mathrm{F}}^2 \asymp \eta \sum_{i>r}^{2m} \sum_{j=1}^{r} G_{i,j}^2 \int_0^t \exp\left[2K_{i,j} t\right] dt \\ &= \mathcal{O}\left(\frac{\eta r}{|K_{i,j}|} \right) = \mathcal{O}\left(\frac{\eta r}{\sigma_r(\boldsymbol{F}_1) - \|\boldsymbol{F}_2\|_2} \right). \end{aligned}$$

Given the error parameter $\epsilon > 0$, we need η to satisfy $\mathcal{O}\left(\frac{\eta r}{\sigma_r(\boldsymbol{F}_1) - \|\boldsymbol{F}_2\|_2} \right) \asymp \epsilon$. Combining with a Markov inequality and Equation (13), we obtain the following lemma,

Lemma 2 (Error Analysis of the Limiting Process)**.** *Given a sufficiently small* $\epsilon > 0$, *let*

$$N = \mathcal{O}\left(\frac{rB}{\epsilon \left(\sigma_r(\boldsymbol{F}_1) - \|\boldsymbol{F}_2\|_2 \right)^2} \log \frac{\sum_{i>r} \widetilde{\gamma}_i^2(0) \cdot B}{\epsilon \left(\sigma_r(\boldsymbol{F}_1) - \|\boldsymbol{F}_2\|_2 \right)} \right)$$

and $t = N\eta$. *Let* $[\widehat{\boldsymbol{U}}(t), \widehat{\boldsymbol{V}}(t)] \leftarrow \boldsymbol{W}(t)$. *We then have*

$$\lim_{\epsilon \to 0} \mathbb{P}[\| \sin \Theta \left(\widehat{\boldsymbol{U}}(t), \boldsymbol{U} \right) \|_{\mathrm{F}}^2 + \| \sin \Theta \left(\widehat{\boldsymbol{V}}(t), \boldsymbol{V} \right) \|_{\mathrm{F}}^2 > \epsilon] \leq \frac{1}{10}.$$

Remark 1. *With standard characterizations of the random matrices (e.g. [Tao12]), we obtain the value of* $\sum_{i>r} \widetilde{\gamma}_i^2(0) = \operatorname{poly}(m)$ *with probability close to 1 when* m *is large. If the ODE and SDE faithfully approximates the algorithm at sufficiently small* η, *i.e., the approximation error of ODE/SDE to the algorithm updates is smaller than the desired precision* ϵ, *then the number of down-sampling steps of the algorithm is*

$$N = \mathcal{O}\left(\frac{rB}{\epsilon \left(\sigma_r(\boldsymbol{F}_1) - \|\boldsymbol{F}_2\|_2 \right)^2} \log \frac{\sum_{i>r} \widetilde{\gamma}_i^2(0) \cdot B}{\epsilon \left(\sigma_r(\boldsymbol{F}_1) - \|\boldsymbol{F}_2\|_2 \right)} \right).$$

Remark 2. *To analyze the algorithm more rigorously, we also show a discrete analysis of the algorithm with a slight modification in the full version[YBZW17].*

4.4 RECOVERY OF NETWORK PARTITION BY CLUSTERING

We have established bounds for obtaining sate embeddings in the last two sections. Next we show that one can recover the partition structure of the underlying network, provided that the Markov chain is lumpable and the sample size is sufficiently large.

Theorem 3 (Recovery of Partition Structure for Lumpable Markov Chains)**.** *Suppose that the estimated eigen-matrices* $\widehat{\boldsymbol{U}}$, $\widehat{\boldsymbol{V}}$, *and empirical distribution* $\widehat{\boldsymbol{\mu}}$ *satisfy*

$$\begin{aligned} &|| \sin \Theta(\widehat{\boldsymbol{U}}, \boldsymbol{U}) ||_F^2 + || \sin \Theta(\widehat{\boldsymbol{V}}, \boldsymbol{V}) ||_F^2 \leq \epsilon \textit{ and} \\ &\qquad \max_{i \in [m]} |\widehat{\mu}_i - \mu_i| \leq \sqrt{\epsilon} \mu_i. \end{aligned} \tag{17}$$

for some $\epsilon \in (0,1)$. *Let* $\widehat{\boldsymbol{M}} := \operatorname{diag}(\widehat{\boldsymbol{\mu}})^{-1} \widehat{\boldsymbol{V}}$ *and* $\boldsymbol{M}$ *as defined in Definition 1. Then for any* $s_i, s_j \in S$,

$$\left| \left\| \widehat{\boldsymbol{M}}_{s_i *} - \widehat{\boldsymbol{M}}_{s_j *} \right\|_2^2 - \left\| \boldsymbol{M}_{s_i *} - \boldsymbol{M}_{s_j *} \right\|_2^2 \right| \leq \tfrac{C\epsilon}{\mu_{\min}^2}.$$

Moreover, suppose that the Markov chain is lumpable with respect to the partition $S_1, \ldots, S_r$. *Then the procedure of Algorithm 2 exactly recovers the network partition as long as*

$$\forall l, s_i \in S_l, s_j \in S_l^c : \left\| \boldsymbol{M}_{s_i *} - \boldsymbol{M}_{s_j *} \right\|_2^2 > \tfrac{2C\epsilon}{\mu_{\min}^2}.$$

Theorem 3 implies that our proposed partition approach can exactly recover the partition of a lumpable Markov chain, as long as the random walk trajectory is long enough to tell the blocks apart. It is possible to extend our analysis to approximately lumpable Markov chains, which is left for future research. The proof is given in the full version[YBZW17].

5 EXPERIMENTS

We experiment with the proposed method on both simulated and real-world data sets.

5.1 SIMULATED DATA

We first simulate random sample paths of nearly low-rank Markov chains and test the performance of the proposed generalized Hebbian algorithm. The model is generated as follows. Let $\boldsymbol{P}_{0[1:r,1:r]} \propto \boldsymbol{B}$, where $\boldsymbol{B}$ is a $r \times r$ matrix with i.i.d. standard normal entries in absolute values. We let $\boldsymbol{P}_{0[r+1:m,r+1:m]}$ be a random permutation matrix to ensure the ergodicity. Then we randomly connect these two parts by letting $\boldsymbol{P}_{0[1:r,r+1:m]} = \boldsymbol{P}_{0[r+1:m,1:r]}' \overset{iid}{\sim} \text{Bernoulli}(0.15)$. Lastly, we normalize each row to obtain a nearly r-rank stochastic matrix $\boldsymbol{P}$, i.e., $\boldsymbol{P}_{[i,:]} = \boldsymbol{P}_{0[i,:]} / \sum_{j=1}^{m} (\boldsymbol{P}_0)_{ij}$.

Convergence and Comparisons. We test the performance of the proposed algorithm on models of different m and r, comparing with the classical Oja's algorithm and the direct SVD of the frequency matrix using the same simulated sample data. We first pick a constant stepsize for an appropriate number of iterations as the warmup phase, then pick a decreasing stepsize $\propto 1/k$ to accelerate the convergence. The same stepsizes are also used in Oja's algorithm for comparison due to the similarity between two algorithms. All tests are done on the simulated transition data for 100 independent trials and we plot the 90% confidence error bar. Figure 2 shows that the convergence rate of subspace angle of the proposed algorithm matches the batch SVD and the Oja's algorithm.

We compare the GHA with batch SVD, which is to compute the empirical transition matrix from all past data and perform SVD. Figure 2 shows that the convergence speed of the online GHA method is nearly the same as that of batch SVD (up to a constant factor). It suggests that the online GHA method is utilizing data as efficiently as the batch method.

We also observe that GHA and Oja's iteration have similar performance when processing online random walk data. And in some cases, the proposed GHA outperforms Oja's algorithm. We note that GHA has better runtime than the Oja's iteration because it does not perform the QR factorization which is required per each Oja's iteration.

Down-sampling. We test the convergence performance of the proposed generalized Hebbian algorithm using different down-sampling block lengths for the case $m = 10$, $r = 3$. Choices of stepsizes are picked as before. Figure 5 shows the comparisons of both the iteration complexity and the sample complexity. As we can see, the down-sampling step does handle the dependency of the streaming data and accelerates the convergence in terms of number of iterations. The overall sample complexity has not been significantly afftected by downsampling.

Recovery of lumpable networks. We test the network partition method for recovering the meta states of lumpable Markov processes. We generate associated Markov chains from undirected graphs, whose adjacency matrix $\boldsymbol{W} = \boldsymbol{Z}\overline{\boldsymbol{W}}\boldsymbol{Z}^\top$. Here, $\boldsymbol{Z} \in \mathbb{R}^{m \times r}$ is a randomly generated membership matrix where each row has exactly one non-zeros entry equaling 1; $\overline{\boldsymbol{W}}$ is a symmetric matrix with diagonal entries equaling 0 and upper triangle entries i.i.d. draw from absolute standard normal distribution. It can be verified the associated Markov chain is lumpable with respect to the partition of r groups. For various value of m and r, we test the network partition method proposed in Section 3.3 on simulated sample paths. For each setting, we repeat the experiment for 100 independent trials and calculate the mean of clustering error rate with 90% confidence error bars. Figure 3 shows the decay of clustering error rate as number of iterations gets larger, compared with network partition using direct SVD of the sample frequency matrix, consistent with the theoretical result in Section 4.4. Figure 4 shows the dynamics of clustering of the estimated low-rank representations for the case $m = 250$, $r = 5$.

5.2 MANHATTAN TAXI DATA

We experiment using a real dataset that contains 1.1×10^7 trip records of NYC Yellow cabs from January 2016 [TLC17]. Each entry records the coordinates of the pick-up and drop-off locations, distance and length of trip, and taxifares. Now the question is can we learn a good low-dimensional representation of different locations in Manhattan from these taxi data? To accomplish this, we discretize the map into a fine grid (with cell size roughly 100m) and model each taxi trip as a single state transition of a Markov chain. For example, a taxi picks up a customer at cell s_1 and drops off the customer at cell s_2, then picks up a customer at s_3... We can view $s_1, s_2, s_3 \ldots,$ as the path of states visited by an implicit city-wide random walk, as shown in Figure 1.

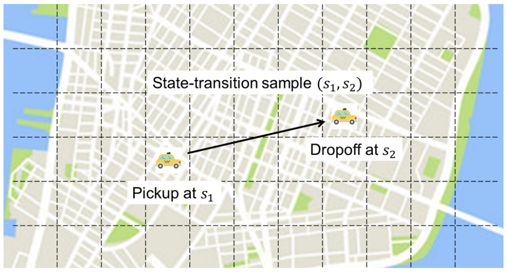
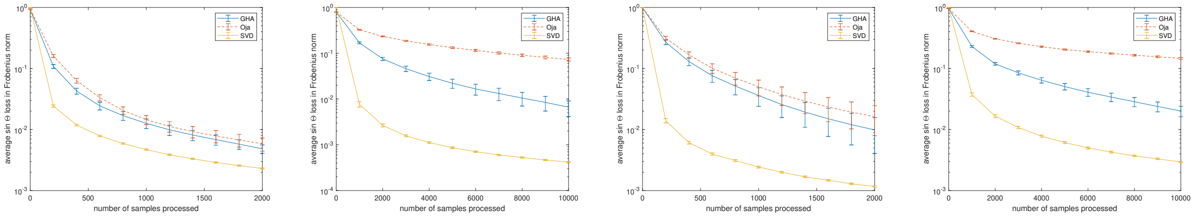


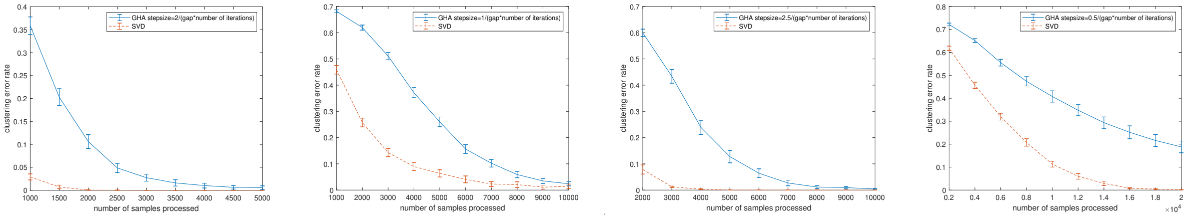
Figure 1: State-transition sample of the NYC taxi data

In order to guarantee recurrence of the random walk, we removed cells that are rarely visited. We end up with 2017 locations and a total of 10^7 effective trips. The first 200 singular values of the empirical transition matrix are shown in Figure 7, which implies a low-rank structure of the data. We apply Algorithm 1 and the partition proce-



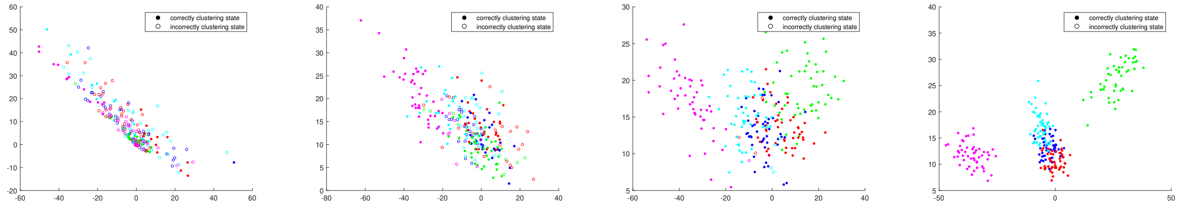
(a) $m = 50, r = 3$ (b) $m = 50, r = 10$ (c) $m = 250, r = 3$ (d) $m = 250, r = 10$

Figure 2: **Convergence of GHA, compared with batch SVD and Oja's iteration.** The convergence rate of the proposed GHA matches SVD in all cases. GHA has similar performance as Oja's iteration in (a) and (c), and outperforms Oja's iteration in the case of (b) and (d).



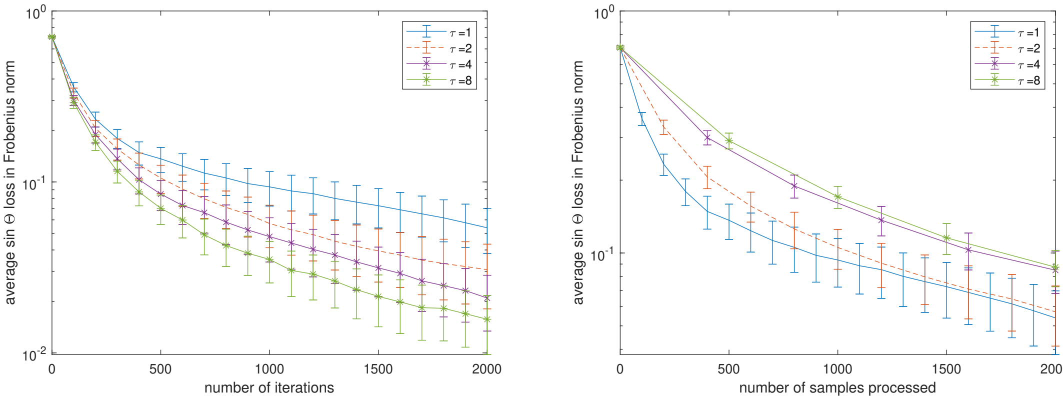
(a) $m = 50, r = 5$ (b) $m = 50, r = 10$ (c) $m = 100, r = 5$ (d) $m = 100, r = 10$

Figure 3: **Clustering Error Rate of Algorithm 2.** (a)-(d) shows the performance of the proposed network partition method on different Markov chain models, in comparison with the clusters obtained using batch SVD and K-means.



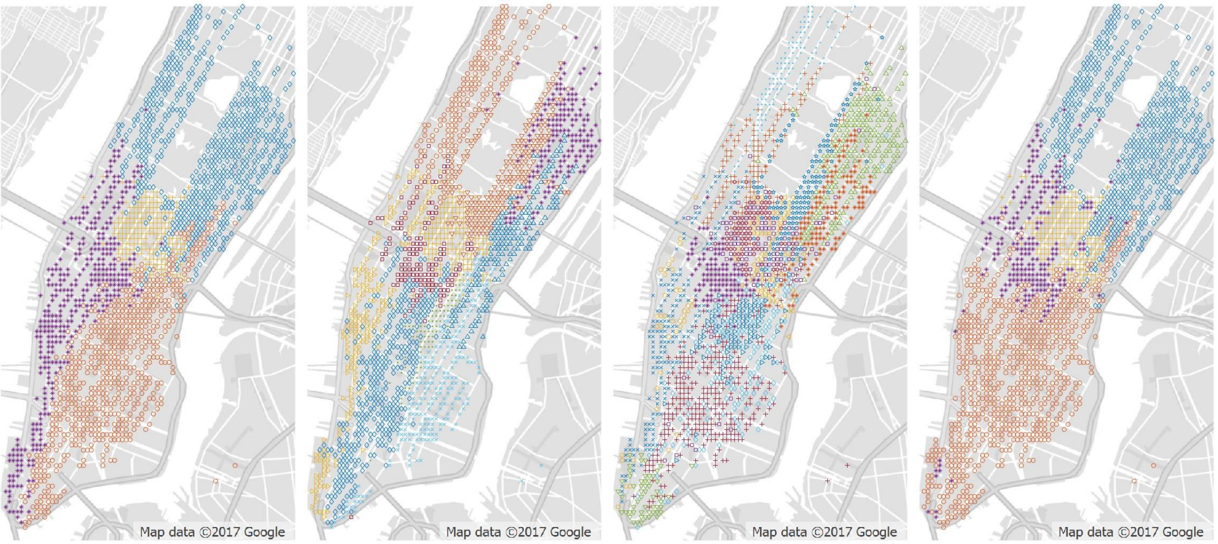
(a) 2000 samples (b) 6000 samples (c) 10000 samples (d) 20000 samples

Figure 4: **Clusters obtained by Algorithm 2.** (a)-(d) plots the first two coordinates of the clustering results of the estimated low-rank representations after certain number of iterations. The ground truth of the meta state of the lumpable Markov chain are colored in same color. The respective clustering error rates are 65.2%, 34.8%, 4% and 0%.



(a) iteration complexity (b) sample complexity

Figure 5: **Convergence of GHA, with different down-sampling block lengths.** $\tau = 1$ represents the case with no down-sampling. (a) shows the convergence performance of different down-sampling block lengths in terms of number of iterations; (b) shows the convergence performance of different down-sampling block lengths in terms of number of samples processed.



(a) GHA: 4 clusters; (b) GHA: 10 clusters; (c) GHA: 15 clusters; (d) SVD: 4 clusters.

Figure 6: The meta-states partition of Manhattan traffic network based on taxi trip records. Each color or symbol represents a meta-state. (a)-(c) are states partition using our online algorithm. (d) is obtained via a direct SVD over a full aggregated matrix.

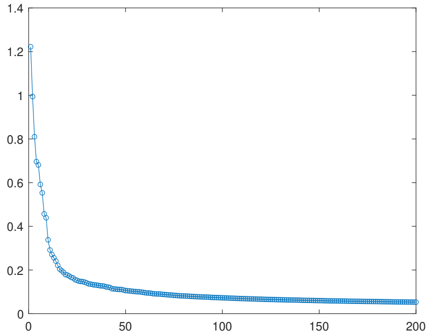


Figure 7: Singular values of the empirical transition matrix from the NYC taxi data

dure of Section 3.3 to the taxi trip data and illustrate the results in Figure 6(a)–6(c). Our method reveals a very informative partition of the Manhattan city according to traffic dynamics (it roughly matches with the comercial areas in Manhattan). We compare our online algorithm with a batch partition procedure and observe that they generate highly similar results (Figure 6(d)) when $r = 4$. A practically impressive observation is: our algorithm uses less than 1 Mbytes memory for $r = 4, 10, 15$. In contrast, the batch partition uses about 200 Mbytes memory even for $r = 4$.

6 CONCLUSIONS AND DISCUSSIONS

We have developed an online learning method for analyzing dynamic transition data generated by a random walk on a network. Out method finds the low-dimensional representation of an implicit network and reveals its latent partition structure. Our method has superior space and computation complexity. We show that it achieves near-optimal global convergence and sample complexity by using an ODE-SDE argument.

Our algorithm and analysis can be adapted to work for the more general online singular value decomposition poblem, which has been considered by [JJK$^+$16, LWLZ16, AZL17, XLS15, AMM16, CYLZ17]. The most critical distinction between our result and existing ones is that ours applies to random walk data and network partition. We summarize other technical improvements of our results:

(1) All the existing analyses require i.i.d samples, while ours applies to dependent Markov samples; **(2)** [JJK$^+$16, LWLZ16, AZL17, CYLZ17] analyzed Oja's algorithm for PCA problem, which conducts QR factorization in each iteration. Our algorithm does not require such decomposition - each iteration uses only vector-to-vector inner products; **(3)** [XLS15] analyzed a similar algorithm as ours but their results require a sufficiently near-optimal initial solution, which is not available in our problems. **(4)** [AMM16] investigated a method based on convex relaxation of SVD but they achieves a sub-optimal sample complexity - $\widetilde{O}(1/\epsilon^2)$. Moreover, their algorithm needs to compute an expensive Fantope projection with a computational complexity $O(m^3)$ per iteration.

In summary, Markov transition data carries rich information about the underlying structure of complex networks and stochastic systems. We hope this work will motivate more research in this area and faster algorithms for applications in networks and reinforcement learning.

References

Raman Arora, Poorya Mianjy, and Teodor Marinov. Stochastic optimization for multiview representation learning using partial least squares. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1786–1794, 2016.

Zeyuan Allen-Zhu and Yuanzhi Li. First efficient convergence for streaming k-pca: a global, gap-free, and near-optimal rate. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 487–492. IEEE, 2017.

Austin R Benson, David F Gleich, and Lek-Heng Lim. The spacey random walk: A stochastic process for higher-order data. *SIAM Review*, 59(2):321–345, 2017.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006.

Zhehui Chen, Lin F Yang, Chris Junchi Li, and Tuo Zhao. Online partial least square optimization: Dropping convexity for better efficiency and scalability. pages 777–786, 2017.

J Harold, G Kushner, and George Yin. Stochastic approximation and recursive algorithm and applications. *Application of Mathematics*, 35, 1997.

Prateek Jain, Chi Jin, Sham M Kakade, Praneeth Netrapalli, and Aaron Sidford. Streaming pca: Matching matrix bernstein and near-optimal finite sample guarantees for oja's algorithm. In *29th Annual Conference on Learning Theory*, pages 1147–1164, 2016.

Björn H Junker and Falk Schreiber. *Analysis of biological networks*, volume 2. John Wiley & Sons, 2011.

Yu Liu, Chaogui Kang, Song Gao, Yu Xiao, and Yuan Tian. Understanding intra-urban trip patterns from taxi trajectory data. *Journal of geographical systems*, 14(4):463–483, 2012.

Chris J Li, Mengdi Wang, Han Liu, and Tong Zhang. Near-optimal stochastic approximation for online principal component estimation. *arXiv preprint arXiv:1603.05305*, 2016.

Rosario N Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems*, 11(1):193–197, 1999.

Attilio Meucci. Review of statistical arbitrage, cointegration, and multivariate ornstein-uhlenbeck. 2009.

Milena Mihail. Conductance and convergence of markov chains-a combinatorial treatment of expanders. In *Foundations of computer science, 1989., 30th annual symposium on*, pages 526–531. IEEE, 1989.

Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. 2001.

Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

Sridhar Nerur, Riyaz Sikora, George Mangalaraj, and VenuGopal Balijepally. Assessing the relative influence of journals in a citation network. *Communications of the ACM*, 48(11):71–74, 2005.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Phillip James Edwin Peebles. *The large-scale structure of the universe*. Princeton university press, 1980.

Clara Pizzuti. Ga-net: A genetic algorithm for community detection in social networks. *Parallel problem solving from nature–PPSN X*, pages 1081–1090, 2008.

Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Society Providence, RI, 2012.

NYC TLC. Nyc taxi and limousine commission (tlc) trip record data, 2017. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`.

E Weinan, Tiejun Li, and Eric Vanden-Eijnden. Optimal partition and effective dynamics of complex networks. *Proceedings of the National Academy of Sciences*, 105(23):7907–7912, 2008.

Bo Xie, Yingyu Liang, and Le Song. Scale up nonlinear component analysis with doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 2341–2349, 2015.

Lin F Yang, Vladimir Braverman, Tuo Zhao, and Mengdi Wang. Online factorization and partition of complex networks from random walks. *arXiv preprint arXiv:1705.07881*, 2017.