Dimensionality Reduction and Learning on Networks

by

Prakash Balachandran

Department of Mathematics
Duke University

Date: _____

Approved:

_____
Mauro Maggioni, Supervisor

_____
Jonathan Mattingly

_____
Sayan Mukherjee

_____
J. Thomas Beale

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Mathematics
in the Graduate School of Duke University
2011

Abstract

(Mathematics: Machine Learning)

Dimensionality Reduction and Learning on Networks

by

Prakash Balachandran

Department of Mathematics
Duke University

Date: _____

Approved:

_____
Mauro Maggioni, Supervisor

_____
Jonathan Mattingly

_____
Sayan Mukherjee

_____
J. Thomas Beale

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the Graduate School of Duke University
2011

Abstract

Machine learning is a powerful branch of mathematics and statistics that allows the automation of tasks that would otherwise require humans a long time to perform. Two particular fields of machine learning that have been developing in the last two decades are dimensionality reduction and semi-supervised learning.

Dimensionality reduction is a powerful tool in the analysis of high dimensional data by reducing the number of variables under consideration while approximately preserving some quantity of interest (usually pairwise distances). Methods such as Principal Component Analysis (PCA) or Isometric Feature Mapping (ISOMAP) do this by embedding the data, equipped with a nonnegative, symmetric, similarity kernel or adjacency matrix into Euclidean space and finding a linear subspace or low dimensional submanifold which best fits the data, respectively.

When the data takes the form of network data, how to perform such dimensionality reduction intrinsically, without resorting to an embedding, that can be extended to the case of nonnegative, non-symmetric adjacency matrices remains an important open problem. In the first part of my dissertation, using current techniques in local spectral clustering to partition the network using a Markov process induced by the adjacency matrix, we deliver an intrinsic dimensionality reduction of the network in terms of a non-Markov process on a reduced state space that approximately preserves transitions of the original Markov process between clusters. By iterating the process, one obtains a family of non-Markov processes on successively finer state spaces

representing the original network ands its diffusion at different scales, which can be used to approximate the law of the original process at a particular time scale. We give applications of this theory to a variety of synthetic data sets and evaluate its performance accordingly.

Next, consider the case of detecting astronomical phenomenon solely in terms of the light intensities observed. There already exists a large database of prior recorded phenomena that has been categorized by humans as a function of the observed light intensity. Given these so-called class labels then, how can we automate the procedure of extending these class labels to the massive amount of data that is currently being observed? This is the problem of concern in semi-supervised learning.

In the second part of my thesis, we consider data sets in which relations between data points are more complex than simply pairwise. Examples include gene networks where the the data points are random variables, and similarities of a subset are measured by non-independence of the corresponding random variables. Such data sets can be illustrated as a hypergraph, and the natural question for diagnosis becomes: how does one perform transductive inference (a particular form of semi-supervised learning)? Using the simple case of pairwise and threewise similarities, we construct a reversible random walk on undirected edges induced by threewise relations (faces). By pulling the random walk back to a random walk on the vertex set and mixing it with the random walk induced by pairwise similarities, we perform diffusive transductive inference. We present applications and results of this technique, and analyze its performance on a variety of data sets.

To my family and friends for their love and support and my advisors for their experience and patience.

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Symbols

$\mathbb{R}$	Real Numbers.
$\mathbb{R}^+$	Non-Negative Real Numbers.

Abbreviations

PCA	Principal Component Analysis.
MDS	Multidimensional Scaling.
ISOMAP	Isometric Feature Mapping.
SSL	Semi-Supervised Learning.
CV	Cross-Validation.
PSA	Prostate-Specific Antigen.
NED	No Evidence of Disease.
TV	Total Variation.
FAKS	Function Dependent Kernel Smoothing.
FAEF	Function Dependent Eigenfunctions.
RHS	Right Hand Side.
LHS	Left Hand Side.
i.i.d.	Independent and Identically Distributed.
SVD	Singular Value Decomposition.

Acknowledgements

I'd like to thank SAMSI's year long program in Complex Networks during 2010-2011 for their support.

I'd also like to thank Harvard University, Boston University and Duke University for the opportunity to visit and exchange knowledge with their affiliates.

Finally, I'd like to acknowledge my advisors Mauro Maggioni and Jonathan Mattingly without whom this would not be possible.

1

Introduction

1.1 Machine Learning

The importance of technologies that automate everyday tasks in human life can hardly be understated. We see evidence of this today from smart phones and the internet that allow us to freely send and receive information to supercomputers and automated assembly lines that allow the production of goods and services without human supervision.

Essential to the *quality* of these innovations and services are the algorithms that run them. Cellular phones, the internet, databases and assembly lines have been around for awhile, but they've become smarter by completing our sentences, searching the internet for queries customized to the user, and dynamically changing according to their environments. They've also spawned technologies that just 10 years ago seemed unimaginable: electronic signatures that verify the authenticity of the signor, pacemakers that deliver the appropriate voltage during a particular emergency and cars that tell us when we're too close to one another car while parallel parking.

In creating such smarter algorithms, one faces a computational difficulty: with

the advent of more efficient computers, one has a massive amount of input data of different types that require processing before the optimal output is delivered. This is referred to as the *curse of dimensionality* where there is an exponential increase in the number of states in the number of state variables.

Development of such new, dynamic, algorithms that can resolve the curse of dimensionality are encompassed in an active field of research known as *machine learning*. As the name suggests, machine learning is programming computers to optimize a performance criterion using example data or past experience in the regime where the input data is large. Two particular subfields of machine learning are *dimensionality reduction* and *semi-supervised learning*.

To get a handle on what dimensionality reduction does, consider specifying the state of air molecules in a room. In order to do this, one requires three variables specifying the position of each particle and three variables specifying the momentum, so that for N particles in the room there are 6^N possible configurations!

However, statistical physics tells us that in order to determine the state of a gas, we need only keep track of the pressure, volume and temperature. Thus, while at first sight there are 6^N many independent degrees of freedom in the system, there are three *effective* degrees of freedom that approximately characterize the system. This is the name of the game in dimensionality reduction: given a massive data set, find the effective degrees of freedom characterizing it. This also has a considerable computational impact, since one achieves a substantial degree of compression while preserving fidelity in the data, and so also allows a more efficient analysis of the data itself.

In semi-supervised learning, one has a massive amount of data which one wants to label, say, with the numbers $\{-1, 1\}$. This problem occurs frequently, say, in the construction of an algorithm that can identify emails as spam or not. The problem here is that it might be too expensive to label all of the data. Given that we can

label a subset of the data, when can we expect to be able to learn the labels on the rest of the data accurately (and efficiently)? This is what semi-supervised learning tries to tackle.

These classical areas of machine learning face some difficulties, however, when the data comes to us as network data, such as biological networks, social networks, telecommunication networks, the internet, or citation networks. Modern research strives to rectify this, and develop efficient algorithms that allow one to perform dimensionality reduction and semi-supervised learning on networks, which is what this dissertation concerns.

This chapter is an introduction that serves three purposes. The first is to introduce dimensionality reduction and semi-supervised learning in its classical framework more precisely to give the reader an idea as to the current state of research in these areas. The second is to introduce the subject of analysis: networks, and why classical dimensionality reduction and semi-supervised learning might not always work on these data sets. The third introduces the contents of this thesis.

1.2 Dimensionality Reduction and Semi-Supervised Learning

In this section, we give a more detailed treatment of dimensionality reduction and semi-supervised learning.

1.2.1 Dimensionality Reduction

Dimensionality reduction is considered part of *unsupervised learning* where one has a massive amount of data available, but knows nothing else about it, and wants to infer structure such as its intrinsic dimensionality as in dimensionality reduction or how points cluster as in density estimation. The two are very related, and so we begin with the latter, it being the simpler of the two.

Density Estimation

A basic question about the data is: can we infer how the data clusters, or to estimate its density?

The answer is surprisingly yes depending on the circumstances. As an example, consider a company with demographic information and past transactions of its customers. The company may want to see the distribution of the profiles of its customers to see what kinds of customers frequently/infrequently occur.

At first glance, it might seem like the company could get any type of customer. However, if the company is very specific in its services, only people with a certain type of demographic situation would approach the company to help solve their problems. This could be thought of as the "mean" profile of a customer. The question then remains to understand the *distribution* about the mean of these profiles.

Learning distributions for data taking on numeric values is an active and ongoing field of research in statistics referred to as *density estimation*. As the names suggest, the goal is to infer dense regions in the data for the purposes of understanding the distribution or trends in the data. The optimization criteria here will be a cost function penalizing large deviations which we want to minimize. Note that in this circumstance, there is *no* example or historic training data.

Density estimation techniques split up into three categories: *parametric, semi-parametric, and nonparametric methods* [Alpaydin (2010)].

Parametric Density Estimation

Let $X_1, \ldots, X_n$ be independent, identically distributed real-valued random vectors whose common distribution has a density with respect to Lebesgue measure on $\mathbb{R}^n$, $p(x)$. The problem then is to estimate p. An estimator of p is a function

$$x \mapsto p_n(x) = p_n(x, X_1, \ldots, X_n).$$

If we know *a priori* that p belongs to a family $\mathcal{F} = \{p(x|\theta) : \theta \in \Theta\}$ (that is, the family $\mathcal{F}$ is *parametrized* by θ) then the density estimation problem is a *parametric* one.

An example of a parametric estimation technique is the method of *maximum likelihood* (ML). Let $p(x|\theta)$ be the parametric family, say $\theta = (\mu, \sigma)$,

$$p(x|\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

If $\{x_i\}_{i=1}^n$ are the data, define the *likelihood function* by

$$L(D|\theta) = \prod_{i=1}^{n} p(x_i|\theta)$$

and the *log-likelihood function* by

$$L(\theta) = -\log(L(D|\theta)) = -\sum_{i=1}^{n} \log p(x_i|\theta).$$

The likelihood function prescribes how likely the observed data is as a function of θ. The goal then is to maximize this likelihood by maximizing $L(D|\theta)$ or minimizing $L(\theta)$. One can do this by standard calculus or gradient descent methods.

One critique of the maximum likelihood method is that it tends to overfit the data (that is, it could fit noise). One way around this to use the *maximum a posteriori* (MAP) method. Here, we include a *prior* distribution $\pi(\theta)$ which is the probability of θ before seeing any data. π incorporates prior knowledge in the problem or beliefs as to what the right regime of θ should be. The goal is then to use Bayes' rule to maximize the *a posteriori* probability, $\pi(\theta)L(D|\theta) = \pi(\theta)\prod_{i=1}^n p(x_i|\theta)$.

Regardless of the method one chooses, the problem in parametric density estimation is to determine what class of distributions $p(x|\theta)$ to use in a given problem, and what algorithm to use to maximize the likelihood.

Semi-Parametric Density Estimation

Suppose now that the data doesn't admit a single class of parametrized probability distributions. Such an example occurs in optical character recognition where there are two ways of writing the number 7 [the American writing has no bar through the middle, while the European one does]. When the data contains instances of both, the digit 7 should be represented as the disjunction of the two groups. In this case, we might need two parametrized families, in which case the problem becomes a *semiparametric* density estimation problem.

The prefix "semi" stems from the fact that while we still use parametric density estimation for each group, we still need some probability distribution on the *number* of groups themselves, independent of the data. We could then try to estimate the mixture density

$$p(x) = \sum_{i=1}^{k} p(x|G_i)p(G_i)$$

when we know there are k-groups. Aside from the [usual] problem of choosing a parametric family for each group, the most important question in this setting is how can we estimate the number of groups k? This is so-called *clustering* problem: separating the data set into regions of similar data points away from dissimilar points.

There exist several clustering algorithms, most exploiting some sort of embedding into a Euclidean space which are *extrinsic* and other which don't and are *intrinsic*. Mapping the data into a Euclidean space while preserving its clustering properties is intimately related to dimensionality reduction which is taken up in more detail in the Dimensionality Reduction portion of this section. We give examples of both extrinsic and intrinsic clustering algorithms.

The most basic extrinsic algorithm is the so-called *k-means* algorithm. Given the data as a subset of Euclidean space, $\{x_i\}_{i=1}^n \subseteq \mathbb{R}^d$, assume *a priori* that there are

k clusters with initial (randomly chosen) centroids located at $\{m^t\}_{t=1}^k \subset \mathbb{R}^d$. The idea is then to loop over all the data points x_i, find its nearest centroid, say m^j, and assign x_i to the j^{th} cluster. Once this is done, new centroids are computed using the these new clusters, and the process is repeated until the centroid locations $\{m^t\}_{t=1}^k$ stabilize (converge). The algorithm is given in table 1.1.

Table 1.1: k-means clustering algorithm on a data set $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$.

Initialize with $\{m^t\}_{t=1}^k$
Repeat:
For all x_i:

$$b_i^t = \begin{cases} 1 & \text{if } ||x_i - m^t|| = \min_j ||x_i - m^j|| \\ 0 & \text{Otherwise.} \end{cases}$$

Reassign m^t according to

$$m^t = \sum_i \frac{b_i^t x_i}{\sum_i b_i^t}$$

Until $\{m^t\}$ converge.

k-means clustering exploits the embedding of the data set into a Euclidean space in order to seed the initial k centroid locations in $\mathbb{R}^d$. One way around this is to *hierarchically cluster* the data set, which does not require an embedding of the data into a Euclidean space, but instead, a metric or distance function on the data.

In intrinsically analyzing a data set, graph theory is often employed. A *graph* or *network* is a collection of interacting entities where entities are represented by nodes and similarities or strength of interaction between them as edges.

The edge data on the network can be described through the *adjacency matrix* W. On a network with n entities V, it is an $n \times n$ matrix or array whose entry $W(i, j)$ stores the relationship of j relative to i. Notationally: $N = (V, A)$. The entries can be 0/1 if a relationship is present or absent, $\mathbb{R}^+$ to reflect the strength of relationships, signed to denote regulation, etc. There can be multiple adjacency

matrices if the edge data are of different types.

A network is undirected if all the adjacency matrices are symmetric. It is directed otherwise.

In the framework of graph theory then, hierarchical cluster simply requires a weighted, undirected, graph whose nodes are the data points and weighted edges are the distance between points. We cover networks more in depth in section 1.3.

An example of a hierarchical clustering algorithm is *agglomerative clustering*. As its name suggests, the algorithm starts with N groups and at each stage chooses two closest groups to merge. The algorithm continues until there is a single one. There are two metrics that researchers use to determine the distance between groups:

In *single link clustering*,

$$d(G_1, G_2) = \min_{x_i \in G_1, x_j \in G_2} W(i, j), \tag{1.1}$$

and in *complete link clustering*,

$$d(G_1, G_2) = \max_{x_i \in G_1, x_j \in G_2} W(i, j). \tag{1.2}$$

We summarize agglomerative clustering in table 1.2.

In agglomerative clustering, a useful diagram mapping merging clusters is the *dendrogram*. We give an example in figure 1.1. Here, the y-axis is the height of the tree. At height h, if two clusters are grouped together if the distance between them if less than h. By cutting the dendrogram at height h, we obtain a partition of the data at the selected precision.

One major objection to the previous methods are that they prescribe the initial number of clusters beforehand instead of learning them directly from the data. We now survey some state-of-the-art intrinsic algorithms (at the time this thesis was written) for which the number of clusters is learned from the data [Spielman and

Table 1.2: Agglomerative clustering algorithm using a metric d such as (1.1) or (1.2).

Initialize with N groups $\{G_1, \ldots, G_N\}$
Repeat:
Between each G_i and G_j, compute $d(G_i, G_j)$.
Let $(k, \ell) = argmin_{i,j} d(G_i, G_j)$, and merge G_k, G_ℓ, deleting G_k and G_ℓ.
Until there is one cluster.

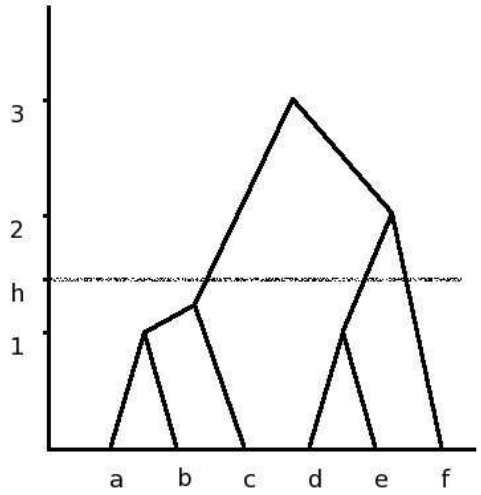


FIGURE 1.1: Dendrogram for agglomerative, single-link clustering. x-axis parametrizes initial clusters, y-axis the height of the tree. At height h, if two clusters are grouped together if the distance between them if less than h. By cutting the dendrogram at height h, we obtain a partition of the data at the selected precision.

Teng (2004), Anderson and Chung (2006), Mahoney (2010), Anderson and Chung (2007)]. These algorithms are driven by a common idea. The first three can only be applied to networks where the metric is symmetric, while the latter concerns cases otherwise. The notation required to present these algorithms is rather intensive, so we present only the ideas here restricted to the case where the metric is symmetric.

The setup is that the data set is equipped with a symmetric metric in the form of a weighted, symmetric, adjacency matrix, W. Given this matrix, the *conductance*

of a subset S is given by,

$$\Phi(S) = \frac{\sum_{i \in S, j \in S^c} W(i,j)}{\min(Vol(S), Vol(S^c))} \tag{1.3}$$

where $Vol(S) = \sum_{i \in S} d(i)$ and $d(i) = \sum_j W(i,j)$. $Vol(S)$ measured the strength of the connections contained in S. $\Phi(S)$ scores how well S is bottlenecked from S^c: the smaller $\Phi(S)$ is, the fewer edges connect S and S^c while the more balanced the volumes S and S^c are. Thus, the smaller $\Phi(S)$, the better cluster it is.

Given a starting node x and maximum conductance ϕ these algorithms all try to find a set S with small conductance and moderate volume containing x. We'll cover diffusions on a network in section 1.3, but for the time being, in order to find this set S, the authors construct a random walk X_n on the data set which starts at $X_0 = x$. By taking a few steps according to this random walk, and at each step keeping track of those vertices with the most mass, they find a cluster containing x with moderate volume and $\Phi(S) < \phi$.

It's important to note that these last four algorithms differ greatly from the previous ones in that they're *spectral*: they exploit the eigenfunctions and eigenvalues of functions of the adjacency matrix to efficiently find their clusters. We'll see more of spectral clustering in chapter 2 of this thesis.

Non-Parametric Density Estimation

Suppose now we have no prior information to use parametric or semi-parametric density estimation. If not such prior information is prescribed, then the density estimation problem is a non-parametric one. Nonparametric density estimation will be crucial in Chapter 2 when we estimate the density of exit times in a Markov process from a cluster.

An example of non-parametric techniques are those using kernel density estimators. Here, we assume that we're given $\{(X_i, Y_i)\}_{i=1}^n$ i.i.d. random variables such

that

$$Y_i = f(X_i) + \xi_i$$

where $X_i \in [0,1]$ and ξ_i represents noise contaminating the samples. f will typically be the cdf of the density of $\{Y_i\}_{i=1}^n$, $p(x)$, and $X_i \sim unif[0,1]$ and so the goal is to estimate f and its derivative.

If $F(x)$ is the distribution function for $p(x)$, then

$$p(x) \approx \frac{F(x+h) - F(x-h)}{2h}$$

so that if

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I(X_i \leq x)$$

is the empirical distribution function with I the indicator function on the corresponding set,

$$p_n(x) = \frac{F_n(x+h) - F_n(x-h)}{2h}$$

is the estimator of the density $p(x)$.

If we define $K_0(u) = \frac{1}{2} I(-1 < u \leq 1)$, then we can write,

$$p_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K_0 \left(\frac{X_i - x}{h} \right).$$

In general, we can use other kernels K. The theory then tries to minimize the error, typically either the mean-square error,

$$\mathbb{E}\left[|p - p_n|^2\right]$$

or the L^1 error,

$$\mathbb{E}\left[|p - p_n|\right]$$

in terms of the the number of samples n, the bandwidth h, and the choice of kernel K, so that for a given precision ϵ, we can compute an estimator p_n that is within ϵ of the true density by judiciously choosing n, h and K.

For more details on this, we refer the reader to Tsybakov (2010) and Devroye and Gyorfi (1985).

Dimensionality Reduction

In the previous section, we discussed density estimation: that is, given an independent, identically distributed sample from a probability distribution with density $p(x)$ on $\mathbb{R}^n$, estimate $p(x)$ by either using parametric, semi-parametric, or non-parametric methods. What happens if there is no such probability density?

Such an example occurs in the spectacular discovery in Tenenbaum (2000), where 4096 64x64 pixel images of a persons face were found to live near a three dimensional nonlinear submanifold of $\mathbb{R}^{4096^2}$. Since this submanifold doesn't "fill" up the ambient space (there is no density with respect to Lesbesgue measure), any sort of density estimation technique previously discussed is inapplicable.

This example shows how clustering is inextricably linked to dimensionality reduction since the existence of regions where the data clusters allows us to infer the continuous object (in this case, the three-dimensional submanifold) that approximates that region which might not be "filling up" the ambient space, and reduce the intrinsic dimensionality from 4096^2 to 3. In dimensionality reduction, this will always be the name of the game: given the data $\{x_j\}_{j=1}^n$ that at first glance might have d degrees of freedom (they needn't be embedded in a Euclidean space), find an embedding of the data onto a submanifold of $\mathbb{R}^d$ whose dimension $m << d$.

By reducing the dimensionality of high dimensional data, one achieves substantial compression, and also learns structure in the data that can be useful in analysis. However, this compression should try to preserve some quantity of interest in the

data [pairwise distances or some quadratic loss function, for example]. In this way, dimensionality reduction reduces the complexity of the original data while remaining faithful. We give some basic examples starting with *Principal Component Analysis* (PCA) before moving on to the classical *Multidimensional Scaling* (MDS) of Borg and Groenen (2005), *Isometric Feature Map* of (ISOMAP) Tenenbaum (2000) and finally to the spectacular result of *Laplacian Eigenmaps* of Belkin and Niyogi (2003) linking (spectral) clustering to dimensionality reduction.

In PCA, the goal is to find a linear subspace that well-approximates the data. Given a data set of n points that lies in a d dimensional Euclidean space, let X be the $d \times n$ matrix consisting of those points. Here, we'll assume that the empirical mean of the n samples are subtracted from each column.

Let $X = W\Sigma V$ be the Singular Value Decomposition (SVD) [Horn and Johnson (1985)], where Σ is a $d \times n$ nonnegative, diagonal matrix and W and V are $d \times d$ matrix and $n \times n$ orthogonal matrices [$WW^T = I$]. The diagonal elements of Σ are called the *singular values* of X, and since they're nonnegative, we'll order them in decreasing order $\sigma_1 > \ldots$. Let W_L be the $L \times d$ matrix obtained by keeping the first L rows of W. Then,

$$Y_L = W_L X$$

is an $L \times n$ matrix which reduces the dimension of X from d to L.

The idea behind this construction is that each singular value is proportional to the portion of the variance of the data correlated with each row of W. By projecting the data onto the first L rows, the construction therefore preserves most of the variance, so that since the remaining singular values are small, we can throw away the corresponding components with minimal loss of information. All this can be shown rigorously [Alpaydin (2010)]. The algorithm is presented in table 1.3.

MDS generalizes PCA to the case where the data might not come as a subset of

Table 1.3: The algorithm for Principal Component Analysis (PCA).

Initialize with n observations of vectors in $\mathbb{R}^d$ stored in a $d \times n$ matrix X. Compute the SVD of X, $X = W\Sigma V$. Order the singular values of Σ in decreasing order $\sigma_1 > \ldots$. For a reduction that preserve σ of the total variance, determine L such that $\sum_{i=1}^{L} \sigma_i \leq \sigma \leq \sum_{i=1}^{L+1} \sigma_i$. Compute the $L \times n$ matrix $Y = W_L X$.

a Euclidean space, but is equipped with a distance function encompassed, as usual, in a nonnegative, symmetric matrix W. The goal is to obtain an embedding of the data into a Euclidean space whose configuration minimizes a loss function (usually, one wants to preserve distances). If the coordinates are $\{x_j\}_{j=1}^{n}$ are the coordinates in such an embedding, the cost function we wish to minimize is then,

$$\min_{x_1,\ldots,x_n} \sum_{i<j} \left| ||x_i - x_j|| - W(i,j) \right|^2 \tag{1.4}$$

although one could certainly use another cost function. To minimize this cost, one uses a numerical optimization technique.

The previous two results were linear in that we obtained an reduction of the dimensionality of the data set by delivering a linear approximation to it. ISOMAP builds on MDS, but seeks to preserve the intrinsic geometry of the data by preserving distances *locally* and then "patching" them together to recover distances globally. This is in stark contrast to (1.4) where one tries to preserve all distances everywhere simultaneously.

ISOMAP requires three steps given in table 1.4. The first step in the algorithm involves constructing an undirected weighted graph using the distance function. One may do this in two ways:

1. Connect each point to all points within some fixed radius ϵ,

2. Connect each point to its k nearest neighbors.

ISOMAP can also be shown [Tenenbaum (2000)] to asymptotically recover the true dimensionality and geometric structure of a large class of nonlinear manifolds.

Table 1.4: The algorithm for ISOMAP.

Construct an undirected graph G using either (1) or (2) above. For all pairs (i, j), compute: $d_G(i, j)$ = the shortest path distance in the graph G. Apply MDS to $\{d_G(i, j)\}$.

The final result we present is that for Laplacian eigenmaps. This part of the discussion presumes some knowledge of differential geometry, and connects spectral clustering algorithms presented in section 1.2.1 to dimensionality reduction, but it is not essential, so the reader may want to skip it.

Let $\mathcal{M}$ be a Riemannian submanifold of $\mathbb{R}^n$ equipped with the usual L^2 metric $||\cdot||$. Let $\mathcal{S}_N$ be a nested increasing subset of points sampled from $\mathcal{M}$. For each N, construct an undirected network on $\mathcal{S}_N$ with weights

$$W_N(i, j) = \exp\left(-\frac{||x_i - x_j||_2^2}{\epsilon^2}\right)$$

where, for $x_i, x_i \in \mathcal{M}$, $||\cdot||_2$ denotes the norm in $\mathbb{R}^n$.

On an undirected weighted network with adjacency matrix W, define the *normalized Laplacian*,

$$\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}},$$

and the *combinatorial Laplacian*,

$$L = D - W$$

where D is the diagonal matrix of the degrees $d(i) = \sum_j W(i, j)$, and where the diagonal entry of D is defaulted to 0 if $d(i) = 0$.

If $\mathcal{L}_N$ is the normalized Laplacian on $\mathcal{S}_N$, then, with a judiciously chosen $\epsilon(N)$ such that $\epsilon \to 0$ as $N \to \infty$, Belkin and Niyogi (2003) were able to show that if f is a function on $\mathcal{M}$ and f_N is f evaluated on $\mathcal{S}_N$,

$$\langle f_N, \mathcal{L}_N f_N \rangle \to \langle f, \mathcal{L} f \rangle \quad \text{as } N \to \infty.$$

The reason that this theorem is so spectacular is that it related spectral clustering techniques from section 1.2.1 to the manifold obtained via dimensionality reduction. Spectral techniques exploit the eigenfunctions and eigenvalues of $\mathcal{L}$ on an undirected network, so this theorem proves, intuitively at least, that clusters in the data survive as bottlenecked regions in the manifold.

We leave our final dimensionality reduction discussion, Coifman and Maggioni (2006), to section 1.4 for the departure to chapter 2 of this thesis.

1.2.2 Semi-Supervised Learning

Dimensionality reduction and density estimation are classified as part of *Unsupervised learning*, meaning they do not take advantage of any labeling or additional information that might be present in the data. *Semi-supervised learning* as we saw in section 1.1 uses some a subset of labeled data to learn the labeling on unlabeled data. We give some detail of this kind of machine learning here, taken from Alpaydin (2010) and Chapelle (2010).

Classification

Consider a bank that's in the business of giving loans. For the bank to make a profit, it's important to identify which customers are high risk and which are low risk when they walk in the door. Low risk loans will typically be given a low interest rate - these are the safe bets for the bank, but the bank won't make much off of them. The high risk loans are where the bank really makes money since they typically charge a high

interest rate. On the other hand, if they have too many high risk loans, customers can default and the bank could go out of business.

In this problem then, the optimization criterion becomes that of maximizing profit. We refer to the labels of low risk and high risk as *class labels*.

For the input data, we first note that before giving out any loan, the bank makes a meticulous record of the customer's financial history including information such as income, savings, collateral, age, financial history, etc. If we could infer a rule mapping customer attributes to class labels with high probability, then we could diagnose a customer as low risk or high risk when they walk in the door. Inferring this rule is the *classification problem*.

To learn this rule, we must have existing data, which is taken from a record of past loans. The purpose for classification then is to take this historic data, compute this rule mapping customer financial attributes to class labels, and use it to predict whether a customer is low risk or high risk when they walk in the door.

Pattern Recognition

Similar to the problem of classification is that of *pattern recognition*. Here, we want to teach a computer to recognize patterns in the data with a high degree of accuracy. Accuracy becomes our optimization criteria, and training the computer with a *training set* of previously stored patterns becomes our example data.

A particular type of example comes from *optical character recognition*. Consider electronic signatures taken by USPS: when a package is delivered, the recipient is required to electronically sign his or her name on a pad. We would like a computer to automatically read the characters in this signature to verify the authenticity or identity of the signor with a high degree of accuracy.

Of course, the accuracy, or percentage error, becomes our cost function that we're trying to minimize. Instead of two classes, however, as in the problem of

classification, we have a family of classes consisting of alphanumeric characters. The problem then becomes that of classification: given previous signatures consisting of characters we know, learn a rule mapping the attributes of these characters to the characters themselves. Then use this algorithm in a computer to recognize characters in a real time setting.

We now proceed to give some graph based algorithms for semi-supervised learning. First, we make the following discussions a little more precise. Given a large amount of data X with a small fraction of it $\bar{X}$ as labeled with class labels $c_1, \ldots, c_n$, the goal of semi-supervised learning is to infer a *classifier* f that labels $\bar{X}$ accurately, and extends the to labels on $X - \bar{X}$. The criteria we minimize is error rate of prediction.

The crucial assumption that semi-supervised learning runs on is the following: if two points x_1, x_2 in a high density region are close, then their labels should be the same.

The first algorithm we present is one of *label propagation* on an undirected network. First, one uses either of the metrics (1) or (2) as in ISOMAP in section 1.2.1 to construct an undirected network on the data, whose weights are stored in the adjacency matrix W. Then, one constructs the matrix $P = D^{-1}W$, where $D = \text{diag}(d(i))$ and $d(i) = \sum_j W(i,j)$.

It's not hard to see that $\sum_j P(i,j) = 1$ for all i, so that since $W(i,j) \geq 0$, the rows of P define probability distributions for each data point x_i. This defines the *probability transition matrix* for a Markov chain on the data. That is, it defines a random walk on the data $\{X_n\}$ whose transitions are "memory forgetting," i.e.

$$\mathbb{P}\left[X_n = x_n | X_{n-1} = x_{n-1}, \ldots, X_0 = x\right] = \mathbb{P}\left[X_n = x | X_{n-1} = x_{n-1}\right] = P(x_{n-1}, x_n).$$

By this property, we also have:

$$P^n(x, y) = \mathbb{P}\left[X_n = y | X_0 = x\right]$$

so that if ν is a probability distribution on the data, and f is a function on the data,

$$\left(\nu P^n\right)(y) = \mathbb{P}\left[X_n = y\right], \quad \left(P^n f\right)(x) = \mathbb{E}\left[f(X_n) | X_0 = x\right].$$

The idea behind label propagation is to use this random walk to diffuse class labels from the training set to the test set. For simplicity, suppose the labels are simply $\{-1, 1\}$. One initializes with a guess $\hat{Y}^{(0)}(y_1, \ldots, y_\ell, 0, \ldots, 0)$ with the first ℓ points being those of $\bar{X}$ and the remaining those of $X - \bar{X}$, and then takes a step in the random walk propagating the labels to $\hat{Y}^{(1)} = P\hat{Y}^{(0)}$. However, since one always knows the true labels of $\bar{X}$, $Y_\ell = (y_1, \ldots, y_\ell)$, one replaces the first ℓ entries of $\hat{Y}^{(1)}$ with Y and then repeats the procedure until the label vectors $\hat{Y}^{(t)}$ converge. The algorithm is presented in table 1.5.

Table 1.5: Label propagation algorithm of Zhu and Chahramani. Taken from Chapelle (2010).

Compute an affinity matrix W using (1) and (2) as in ISOMAP in section 1.2.1.
Compute $d(i) = \sum_j W(i, j)$.
Initialize $(y_1, \ldots, y_\ell, 0 \ldots, 0) \mapsto \hat{Y}^{(0)}$.
Iterate:
$\quad D^{-1} W \hat{Y}^{(t)} \mapsto \hat{Y}^{(t+1)}$.
$\quad Y_\ell \mapsto \hat{Y}_\ell^{(t+1)}$.
Until convergence to $\hat{Y}^{(\infty)}$.
Label the point x_i by i^{th} component of $\hat{Y}^{(\infty)}$.

The next algorithm we present is that of *discrete regularization*. Again, we assume we're given a data set equipped with a nonnegative, symmetric, adjacency matrix W on n data points, V. Let $\mathcal{H}(V)$ and $\mathcal{H}(E)$ denote the Hilbert space of functions

on the edges and vertices of our n data points, equipped with the inner products,

$$\langle f, g\rangle = \sum_{v\in V} f(v)g(v), \quad \langle \phi, \chi\rangle = \sum_{[u,v],u,v\in V} \phi([u,v])\chi([u,v]).$$

Define the *gradient operator*, $\nabla : \mathcal{H}(V) \to \mathcal{H}(E)$, by

$$(\nabla f)([u,v]) = \sqrt{\frac{W(u,v)}{d(v)}} f(v) - \sqrt{\frac{W(u,v)}{d(u)}} f(u)$$

and the norm of the gradient ∇f at v by,

$$||\nabla_v f|| = \left(\sum_{u,v\in V} (\nabla f)^2 ([u,v]) \right)^{\frac{1}{2}}.$$

and the *p-Dirichlet form* of the function f by,

$$\mathcal{S}_p(f) = \frac{1}{2} \sum_{v\in V} ||\nabla_v f||^p .$$

If $Y = (y_1, \ldots, y_\ell)$ are the labels on $\bar{X}$, then the solution to the semi-supervised learning problem is to find a classifier, $f = f^*$ which is the solution to,

$$f^* = argmin_{f\in\mathcal{H}(V)} \left\{ \mathcal{S}_p(f) + \mu||f - Y||_2^2 \right\}.$$

For $p = 2$, this function can be written as the solution to,

$$\mathcal{L}f^* + \mu(f^* - Y) = 0$$

where $\mathcal{L}$ is the normalized Laplacian. The idea here is that $\mu \in (0, \infty)$ is a parameter specifying the tradeoff between the two competing terms, the first being regularization, and the second forcing f to label $\bar{X}$ correctly. The algorithm is presented in table 1.6.

We leave the final semi-supervised learning algorithm, Szlam (2008), to section 1.4 for the departure to chapter 3 of this thesis.

Table 1.6: The discrete regularization algorithm of Zhu and Scholkopf, taken from Chapelle (2010).

Compute an affinity matrix W using (1) and (2) as in ISOMAP in section 1.2.1. Compute $f^* = argmin_{f \in \mathcal{H}(V)} \{\mathcal{S}_p(f) + \mu ||f - Y||_2^2\}$.

1.3 Networks

We saw in Chapter 1.1 various kinds of machine learning on data sets, and found that undirected, weighted networks were a crucial part of an intrinsic analysis of the data set. However, all these algorithms were on *undirected* networks where the weighted adjacency matrix was symmetric in its entries. There are numerous, important, examples of data sets where this does not hold, and so where the above machine learning techniques break down. This is exactly the issue this thesis tries to tackle. This section serves as a more detailed introduction to networks, much of which is taken from Newman (2010).

1.3.1 Definitions and Examples

A *network* or *graph* is a collection of interacting entities where

1. Entities are represented by nodes and,

2. Similarities or strength of interactions between them as edges.

It could occur that in a network, there are relationships more complex than those simply consisting of pairwise. If there is a k-wise interaction among k-nodes, we represent this as a a $k-1$ simplex with vertices consisting of the k-nodes. This structure is referred to as a *hypergraph*. We remark that a hypergraph is distinct from a *simplicial complex*: in the latter, given a k simplex, all of its boundaries of $k-1$ sub-simplices, or *faces*, are included. In a hypergraph, there could be, say, a

triangle with none of its boundary edges present. With a slight abuse of notation, in the discussion that follows, a graph could also refer to a hypergraph.

The edge data on a network can be described through the *adjacency matrix*, A. On a network N consisting of n entities V, it is an $n \times n$ matrix or array whose entries $A(i_1, i_2)$ stores the relationship of i_2 relative to i_1. Higher order relationships can be stored in an *adjacency tensor* whose $A(i_1, \ldots, i_n)$ stores the information of $i_2, \ldots, i_n$ relative to i_1.

Adjacency matrices or tensors can by 0/1 valued to represent if a relationship is present or absent, $\mathbb{R}^+$ (*weighted*) to reflect the strength of relationships, signed to denote up/down regulation, etc. There could also be multiple adjacency matrices/tensors if more than one type of relationship is present.

A network is *undirected* if all its adjacency matrices are symmetric. It is *directed* otherwise.

We've already seen how weighted, undirected, networks occur in machine learning in section 1.2.1. We now proceed to expand the reader's familiarity of networks by giving further examples and questions researchers have regarding them in order to understand how the classical tools of machine learning in section 1.2.1 might not be applicable depending on the data.

The Internet

The most well-known example of a network is the internet, where the nodes are computers and edges are (symmetric) physical data connections between them. Of course, the function of the internet is to transport data between computers in the form of pieces, or *packets*, and an active area of research is how should one choose the best route by which to transport the data. For instance, is the shortest route always the best? How can one avoid bottlenecks in traffic flow that might slow down communication? The adjacency matrix here is 0/1-valued to represent if two

computers communicate with each other.

Understanding the structure of the internet also allows one to develop new communications standards since new protocols are continuously being developed.

The World Wide Web

The next most common example of a network is the world wide web. Here, the nodes are webpages, and $A(i, j)$ represents if there is a link from website i to website j (0/1-valued). Note this network is directed: i might have a link to j but not vice versa.

An active area of research is that of *link structure*. People tend to add links between pages with related content, so that understanding link structure sheds light on content structure. Google, Bing, Amazon as well as other internet companies make billions of dollars by understanding this structure.

Social Networks

In a social network, people are nodes and edges represent connections of some kind between them. On Facebook, for example, undirected edges represent if two people mutually "friend" each other, although in a general social network, relationships could be directed and even weighted to denote the strength of the relationship between two people as a function of other measured variables.

For Facebook, understanding how people choose to make friends is important to understanding how the network changes over time. This in turn, helps in advertisement of various products on the website, which results in revenue for the company. In more general social networks, sociologists are interested in understanding subgroups of people that function as a single unit, or a given person's social circle [which might turn out to be independent from how many friends that person has!].

Biological Networks

This class of networks is rather extensive, so we give just a few.

Neural networks occur in Neurology where nodes are neurons and (directed) edges exist if two neurons communicate with each other. Understanding functional areas of the brain where neurons collectively act as a single unit is a very important area of research in understanding how the brain functions and might aid in the development of cures of several neurological diseases (such as learning a core network that exists during the onset of epilepsy).

Another example is a food web, which is an ecological example where nodes are species in an ecosystem and (directed) edges represent relationships between them. Understanding food webs helps us understand and quantify ecological phenomena such as energy consumption or carbon flows.

Our last example of a biological network is a metabolic network (although similar descriptions will hold for protein-protein interaction networks and gene regulatory networks). This is a biochemical network where vertices are metabolites (the substrates and products of metabolism) and directed edges are reactions that turn one metabolite into another. Of course, these networks are important in order to understand how the body functions.

All of the adjacency matrices above could be directed, undirected, weighted or 0/1-valued depending on the circumstances. Metabolic networks could also have adjacency matrices that are $+/-$-valued to denote up/down regulation of one metabolite on another.

One approach in gene networks is to treat the genes themselves as random variables with a correlation structure that can be measured. If higher order dependencies can also be measured accurately, one can also put higher simplices into the network to make it a hyergraph.

1.3.2 Spectral Graph Theory and Diffusions on Networks

In the previous section, we introduced networks and gave many examples and questions researchers have about them. However, it's clear that when the data arises as network data, and relationships are directed, the tools of machine learning in 1.2.1 aren't applicable. In some circumstances, one might just be interested in whether or not directed edges are present or not, in which case the problem becomes an undirected one again. However in many applications, this leads to a loss of information, and it's crucial to be able to develop machine learning techniques that can handle the directedness of a network.

This is precisely what this thesis concerns, at least in the realms of semi-supervised learning and dimensionality reduction. This section serves to give a broad introduction to the tools and ideas we will be using in the intrinsic analysis of network data, although for the sake of completeness, we'll cover the necessary mathematical formalism in the relevant chapters. We outline exactly the contents in the next section 1.4.

Spectral Graph Theory

We saw in discrete regularization in semi-supervised learning that one has many of the operations on graphs that are available in classical calculus: namely differentiation (the gradient) and the Laplacian. This section covers how these operations can be used to solve some hard graph theoretic problems that arise in classical machine learning, as well as establishing some notation and results that will be useful in understanding how these operations relate to diffusions in the next section 1.3.2. These ideas, along with those of the next section, will be the core of this thesis. This exposition is taken from Chung (1997).

Much of the work in classical spectral graph theory examines the relationship of the eigenvalues and eigenvectors of the graph Laplacian to the properties of the

graph itself (here, we only consider symmetric, nonnegative, pairwise relationships). To review, define the *normalized Laplacian* and *combinatorial Laplacian* by

$$\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}, \quad \text{and} \quad L = D - W$$

where W is a symmetric, nonnegative $n \times n$ matrix, $D = diag(d(i))$, and $d(i) = \sum_j W(i,j)$ is the *degree* of i.

By the spectral theorem for symmetric matrices, we have that the eigenvalues of either Laplacian are real and that there exists an orthonormal basis of eigenvectors. In the discussion that follows, we'll assume that the eigenvalues of both of these Laplacians are ordered in increasing fashion.

Both of these operators have an eigenspace corresponding to an eigenvalue of $\lambda_0 = 0$: for the combinatorial Laplacian, it consists of the function of all 1's, and for the normalized Laplacian the function $f(i) = \sqrt{d(i)}$. So, immediately, we have that the dimensionality of the eigenspace for $\lambda_0 = 0$ encodes the number of connected components, since we'll have such a function for each component defined to be zero everywhere else. Recall that a subgraph S is connected if for every two vertices i, j there exist a sequence of edges $[i, i_1], [i_1, i_2], \ldots, [i_m, j]$ connecting them with $i_j \in S$, and a connected component is a maximal such subgraph.

Recall from section 1.2.1 that the *conductance* of a subset of vertices S is defined by

$$\Phi(S) = \frac{\sum_{i \in S, j \in S^c} W(i,j)}{\min(Vol(S), Vol(S^c))}$$

where $Vol(S) = \sum_{i \in S} d(i)$. The Cheeger constant of a graph G is defined by $g_G = \min_S \Phi(S)$. Recall that $Vol(S)$ measures the strength of the connections contained in S, and $\Phi(S)$ scores how well S is bottlenecked from S^c: the smaller $\Phi(S)$ is, the fewer edges connect S and S^c while the more balanced the volumes S and S^c are. Thus, the smaller $\Phi(S)$, the more S is clustered. Thus, the Cheeger constant measures the

best possible cluster in the entire graph.

What the next eigenspace for λ_1 characterizes is given by the celebrated Cheeger's inequality:

$$\frac{g_G^2}{2} < \lambda_1 \leq 2g_G. \tag{1.5}$$

In a nutshell: look at the eigenvector(s) for λ_1, f_1 and let $S = \{i : f_1(i) < 0\}$. Then, S is approximately the set which achieves Cheeger's constant with an error bounded by 1.5. We remark that finding the set that achieves Cheeger's constant is NP hard since one needs to sweep through the set of all subsets of vertices in the network. Computing the f_1's on the other hand, is known not to be NP hard, so this is a rather good approximation to finding the best cluster (cut) in a graph. For those readers that have kept up with us thus far, this is intimately related to the local spectral algorithms mentioned in 1.2.1.

One can ask at this stage if the higher eigenfunctions encode anything similar. Based on the above considerations, the answer is yes, although not nearly as direct as the previous results.

To proceed further, we need to define a Markov chain. Recall that from section 1.2.2 that $P = D^{-1}W$, where $D = \text{diag}(d(i))$ and $d(i) = \sum_j W(i,j)$.

It's not hard to see that $\sum_j P(i,j) = 1$ for all i, so that since $W(i,j) \geq 0$, the rows of P define probability distributions for each data point x_i. This defines the *probability transition matrix* for a Markov chain on the data. That is, it defines a random walk on the data $\{X_n\}$ whose transitions are "memory forgetting," i.e.

$$\mathbb{P}\left[X_n = x_n | X_{n-1} = x_{n-1}, \ldots, X_0 = x\right] = \mathbb{P}\left[X_n = x | X_{n-1} = x_{n-1}\right] = P(x_{n-1}, x_n).$$

Now,

$$\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} = I - D^{\frac{1}{2}} P D^{-\frac{1}{2}}$$

so that $I - P$ and $\mathcal{L}$ are conjugate, and therefore share the same eigenvectors and eigenvalues.

In Coifman and Lafon (2006), the authors construct a dimensionality reduction for an undirected network by using the eigenfunctions of $\mathcal{L}$ (and hence P) which (approximately) preserve the *diffusion distances*,

$$D_t(x,y) = ||P^t(x,\cdot) - P^t(y,\cdot)||_2 = \sum_u \frac{|P^t(x,u) - P^t(y,u)|^2}{d(u)}.$$

To see how they produce this reduction using higher eigenvectors of $\mathcal{L}$, let λ_j be the eigenvalues of P so that $1 = \lambda_0 > |\lambda_1| > \cdots$ with corresponding eigenfunctions ψ_j (note that $\lambda_j = 1 - \mu_j$ where μ_j are the eigenvalues of $\mathcal{L}$).

For a judiciously chosen value of δ, the idea is to look at:

$$s(\delta, t) = \max\{\ell \in \mathbb{N} : {\lambda_\ell}^t > \delta |\lambda_1|^t\}.$$

Then, the map Ψ mapping the graph to $\mathbb{R}^{s(\delta,t)}$ is given by,

$$\Psi_t(x) = \left(\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x), \ldots, \lambda_{s(\delta,t)}^t \psi_{s(\delta,t)}(x)\right)$$

and one can show that

$$||\Psi_t(x) - \Psi_t(y)||_2 = \left(\sum_{\ell=1}^{s(\delta,t)} \lambda_\ell^{2t} \left(\psi_\ell(x) - \psi_\ell(y)\right)^2\right)^{\frac{1}{2}} \approx D_t(x,y).$$

The moral behind this construction is that $P^t f = \sum_j \lambda_j^t \langle f, \psi_j \rangle \psi_j$ so that one can cut off this approximation after $s(\delta, t)$ with minimal loss of information to preserve the dynamics of the random walk by time t, encoded by the first $s(\delta, t)$ eigenfunctions ψ_j.

Diffusions

We saw in the last section the relationship of the spectral theory of the Laplacian $\mathcal{L}$ to describing the state of a Markov chain P at time t. In this section, we take this discussion further with a more detailed analysis of random processes on networks. This exposition is taken from Levin and Peres (2009) and Norris (1997).

In the previous section, we started out with an adjacency matrix and constructed a random walk. The more natural direction is the other way.

To review, a *Markov Chain* on the finite state space Ω is a random walk $\{X_n\}_{n=0}^{\infty}$ on V satisfying the "memory forgetting" property, or *Markov property*,

$$\begin{aligned}\mathbb{P}\left[X_{n+1} = y | X_n = x, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1\right] \\ = \mathbb{P}\left[X_{n+1} = y | X_n = x\right] = P_n(x, y) \quad \forall n \in \mathbb{N}, \quad x, y, x_j \in \Omega. \quad (1.6)\end{aligned}$$

The distributions $P_n(x, \cdot)$ are called the *transition probabilities* at time n. When $P_n = P$ for all n, the random walk is *time-homogeneous*, and throughout this thesis, all chains are time-homogeneous chains unless otherwise specified.

Given an initial distribution $\pi(x) = \mathbb{P}\left[X_0 = x\right]$, interpreted to be a non-negative row vector indexed by Ω summing to one, we have that for a time-homogeneous chain:

$$\mathbb{P}\left[X_n = x\right] = (\pi P^n)(x).$$

The *stationary distribution* of P is an initial distribution π such that $\pi P = \pi$; that is, it's a non-negative left eigenvector of P with eigenvalue 1. Its interpretation is that if we start in this distribution, we stay there:

$$(\pi P^n)(x) = \pi(x) = (\pi P^m)(x) \Leftrightarrow \mathbb{P}\left[X_n = x\right] = \mathbb{P}\left[X_m = x\right] \quad \forall n, m \in \mathbb{N}.$$

Conditions for the existence of a unique stationary distribution are that P must be:

1. *Irreducible*: Given $x, y \in \Omega$ there exists a path of length t connecting them. (read: $\forall x, y \in \Omega$, there exists $n \in \mathbb{N}$ such that $P^n(x, y) > 0$).

2. *Aperiodic*: The random walk cannot oscillate between distributions supported on different subsets of Ω (read: $gcd\{n : P^n(x, x) > 0\} = 1$).

By the *ergodic theorem*, an irreducible, aperiodic chain has $\mu P^t \to \pi$ as $t \to \infty$ for any choice of initial distribution μ.

A Markov Chain P is *reversible* with respect to the stationary distribution π if

$$\pi(x)P(x, y) = \pi(y)P(y, x).$$

Intuitively, this property says that if we start in π, the dynamics of running the chain forwards for a length of time is the same as running the chain backwards for the same length of time. This allows us to recover an adjacency matrix W for the Markov chain determined by P:

$$W(x, y) = \pi(x)P(x, y) = \mathbb{P}[X_n = x, X_{n+1} = y]$$

which will be symmetric when P is reversible, and nonsymmetric otherwise.

Finally, we introduce the measurement of how close two probability distributions are on a finite state space, given by the *total variation distance*,

$$||\mu - \nu||_{TV} = \frac{1}{2}\sum_x |\mu(x) - \nu(x)| = \inf_{(X,Y), X\sim\mu, Y\sim\nu} \mathbb{P}[X \neq Y].$$

The latter equality is defined via the following notation: let (X, Y) be two random variables whose marginals are μ and ν respectively (i.e. $X \sim \mu$ and $Y \sim \nu$). Such a pair of random variables is called a *coupling* of the distributions μ and ν. The total variation distance then lower bounds the degree to which any coupling of μ and ν disagree (in fact, the infimum can be shown to be achieved).

With this notation, we have the fundamental convergence theorem for Markov chains:

Proposition 1. *Suppose that P is irreducible and aperiodic with stationary distribution π. Then, there exists $\alpha \in (0,1)$ and $C > 0$ such that*

$$d(t) = \max_x ||P^t(x,\cdot) - \pi||_{TV} \leq C\alpha^t.$$

The *mixing time* of such a Markov chain is defined to be,

$$t_{mix}(\epsilon) = \min\{t : d(t) \leq \epsilon\}$$

with *the mixing time* defined by $t_{mix} = t_{mix}\left(\frac{1}{4}\right)$. One can show that $d\left(\ell t_{mix}\right) \leq 2^{-\ell}$.

The whole theory of Markov chain mixing then is to bound the rate of convergence of an irreducible aperiodic Markov chain in order to bound the mixing time.

In the case where the P is reversible and the adjacency matrix W is symmetric, one can show [Levin and Peres (2009)],

$$\max_x ||P^t(x,\cdot) - \pi||_{TV} \leq \frac{e^{-(1-\lambda_*)t}}{\pi_{min}}$$

where

$$\lambda_* = \max\{|\lambda_j| : \lambda_j \text{is an eigenvalue of } P\}$$

is the *absolute spectral gap.*

In light of proposition 7, however, it turns out we can still get exponential convergence to stationarity for a nonreversible Markov chain. The trick is to define $W_{sym} = \frac{W+W^T}{2}$ and bound the rate of convergence in theorem 7 in terms of absolute spectral gap of its normalized Laplacian as in Chung (2005).

These relationships between spectral theory and diffusions on a finite data set allow us to intrinsically analyze a data set in a controlled way by diffusing local information to the rest of the network. We can now discuss how exactly do this in the general setting by summarizing the contents of this thesis.

1.4 Contents of this Thesis

The point of departure of this thesis are two recent results in dimensionality reduction and semi-supervised learning on data sets that use the random walks.

A recent result by Coifman and Maggioni (2006) allows one to perform a multiresolution analysis of a certain class of random walks on a network by efficiently computing, representing, and compressing powers of the transition matrix at different time scales. By doing so, one obtains an effective dimensionality reduction of the network, where instead of a subregion of Euclidean space, one obtains a subgraph of the original network effective representing the original network at a particular time scale of the diffusion.

The outline of this result is as follows. One has equipped a symmetric diffusion operator T that is conjugate to the heat kernel of a (possibly continuous time) Markov process on the network. At each stage of computing dyadic powers T^{2^j}, one finds a set of vectors that (approximately) spans the range of the operator at this time 2^j by employing a local orthogonalization procedure in the neighborhood of each point. That is, if $\mathcal{H}$ is a Hilbert space, $\epsilon > 0$ and $\{v_k\}_{k\in\mathcal{K}} \subset \mathcal{H}$ are vectors spanning the range of T^{2^j}, one tries to find a set of vectors $\{\xi_i\}_{i\in\mathcal{I}} \subset \mathcal{H}$ that ϵ-span $\{v_k\}$, so that for all $k \in \mathcal{K}$,

$$\left|\left|P_{\{\xi_i\}}v_k - v_k\right|\right|_{\mathcal{H}} < \epsilon$$

where P denotes projection of v_k onto the subspace spanned by $\{\xi_i\}$.

This allows compression of the information of T at time 2^j so that one may write T^{2^j} in this representation and iterate the procedure. Assuming the heat kernel converges to stationarity, large powers of T are effectively low rank, so this procedure allows one to describe large powers of T in a form that reflects its low rank structure.

One observation is that the metric used to ϵ-span is the L^2 metric instead of the more natural metric for diffusions, namely total variation distance, which delivers

strong probabilistic guarantees. Our contribution in chapter 2 of this thesis tries to do this.

The outline is as follows. We assume that the network comes to us with a *multiscale structure* in that it is equipped with a continuous time Markov chain that explores certain regions of the network much faster than it takes to explore the entire network. For each j, this information is stored in a multiscale partition $\{\mathcal{C}_{j,k}\}_{k\in\mathcal{K}_j}$ which defines in a nested increasing fashion in j, successively larger regions where the random walker has a low probability of leaving.

Due to the multiscale property, it might take the random walker a significant amount of time to explore the entire network, while a relatively short amount of time to explore each $\mathcal{C}_{j,k}$. For each fixed j, we compress out the random walk at time scales proportional to leaving $\mathcal{C}_{j,k}$ for all k, and deliver a non-Markov process which transitions between these regions at jumps in a way that respects the original process. Since the procedure is repeated for each j, one obtains a multiscale decomposition of the network into general jump processes on subsets reflecting the structure of the random walk at a particular time.

We can then use this multiscale decomposition to approximate the original transition density of the Markov process with strong probabilistic guarantees, so that one obtains a similar degree of compression as the original diffusion wavelets which allows one to compute the heat kernel (and hence any function of the heat kernel) efficiently and accurately at a given time scale. We also give examples.

Next, we saw in section 1.2.2 that one method of semi-supervised learning was label propagation, which used a random walk to push labels from the labeled set to the unlabeled set by running the random walk to stationarity. While this is certainly the smoothest classifier with respect to the random walk, reflecting stationarity, it might turn out there is a better one which is rougher that can be learned before the

random walker reaches stationarity. This is the idea taken up in Szlam (2008).

Here, one is given the same setup: a data set X with a nonnegative, symmetric, adjacency matrix W encoding distances, and a labeled subset of the data $\bar{X}$ with class labels $c_1, \ldots, c_n$ and one wants to infer class labels on $X - \bar{X}$.

For each c_i, an indicator of characteristic function of 1's and 0's, χ_i^{lab}, are constructed, and

$$\chi_i^{lab,t} = P^t \chi_i^{lab}$$

is computed, where t is determined by leave-K-out cross validation on the labeled data. Using this choice of t, the label on an unlabeled vertex x is

$$argmin_i \chi_i^{lab,t}.$$

Probabilistically, this algorithm can be rephrased as follows: the support of the functions $\{\chi_i^{lab}\}$, $\{\mathcal{S}_i\}$ define subsets of the data; to determine the label for an unlabeled point x, one runs the random walk t steps into the future determined by cross validation starting from x and sees which set $\mathcal{S}_i$ is most likely to be hit at that time. The label of x is then just the label of this set.

Our contribution to semi-supervised learning is to use this theory to deliver classifiers that incorporate higher order features in the data beyond pairwise when those are present. The outline is as follows.

For simplicity, we assume only the existence of pairwise and threewise interactions. We use threewise interactions to construct a reversible Markov chain on the set of edges, and use pairwise relations to construct a Markov chain on the set of vertices as usual P_1. We then pull back the edge process to a process on the vertex set, P_2, and perform the diffusive transductive inference described above learning now the *two* parameters c and t via cross-validation, where c is defined by,

$$P = cP_1 + (1 - c)P_2. \tag{1.7}$$

The idea is to probabilistically mix information arising from higher order interactions. In section 1.2.2, we noted that semi-supervised learning works only under the principle that when two points x_1 and x_2 lie in a high density and are close, then their labels should be the same. By mixing Markov chains in the form (1.7), we interpolate between two types of clusters: those clusters expressing pairwise correlations and those expressing threewise correlations, and attempt to diffuse class labels. In addition to giving examples, this gives rise to how well one can perform semi-supervised learning in the case where P is not reversible, as for almost all choices of c, this will be the case.

2

Homogenization of Markov Chains

2.1 Introduction

As mentioned in section 1.4, when an undirected network admits a multiscale partition $\{\mathcal{C}_{j,k}\}_{k\in\mathcal{K}_j}$ whose members at scale j represent regions in which the time scale a (continuous time) Markov process takes to explore them are much smaller than the time scale of exploring the entire network, we deliver a (continuous time) stochastic (non-Markov) process representing the original process' transitions at these scales. By using kernel density estimation for the corresponding hold times, we prove error bounds in total variation distance of such approximate processes to their real ones, and deliver concentration type inequalities determining how many sample paths are necessary to simulate to approximate the law of the process within a given precision, and evaluate the performance of our algorithms on several synthetic data sets.

We assume that the undirected network comes to us with a *multiscale structure* in that it is equipped with a continuous time Markov process X_t that explores certain regions of the network, Ω_0, much faster than it takes to explore the entire network. We store this information in terms of a partition of the network at different scales

$\{\mathcal{C}_{j,k}\}_{k\in\mathcal{K}_j}$ where j is a scale variables, and $k \in \mathcal{K}_j$ are indices at that scale indexing partition members. In our convention, larger j values correspond to a coarser scale, so that partition members become larger. Furthermore, we assume that a set at a coarser scale can be reconstructed from those sets one scale lower:

$$\mathcal{C}_{j+1,k} = \coprod_{k'} \mathcal{C}_{j,k'}$$

Due to the multiscale assumption, X_t might take a very long time to explore the entire network, but a comparatively shorter amount of time to explore $\mathcal{C}_{j,k}$. Given that this is the case, how can we force the random walker out of these regions and explore the rest of the network while respecting the probabilistic dynamics of X_t?

The idea is as follows. At the first scale, $j = 1$, we compress out the random walk by projecting it onto those vertices at scale 0 that join scale 1 to scale 2, defining a new state space Ω_1. This projected process obeys the following rules:

1. Having entered a cluster at $i \in \mathcal{C}_{1,k}$, the random walker flips a coin to choose an exiting vertex $j \in \mathcal{C}_{1,k}$ with a probability equal to starting from i and leaving through j.

2. Having chosen j, he holds at i for the (random) amount of time it takes him to start at i and leave through j, and then transitions.

3. Having moved to j, he then chooses a vertex $\ell \in \mathcal{C}_{1,k'}$ for $k' \neq k$ with probability equal to having transitioned from j to ℓ at scale Ω_0 conditioned on leaving $\mathcal{C}_{1,k}$.

4. He then holds at j for the (random) amount of time it takes him to transition to ℓ, and the entire process is repeated at $\ell \in \mathcal{C}_{1,k'}$.

Since all of the data necessary for this new projected process, $X_t^{(1)}$, are constructed via the appropriate data of the original process X_t, it might not be too surprising that $X_t^{(1)}$ represents X_t well at at the time scale t of leaving the clusters $\{\mathcal{C}_{1,k}\}$.

The entire procedure is then repeated using the transition kernel of $X_t^{(1)}$ and the induced partition of $\{\mathcal{C}_{2,k}\}$ on Ω_1. This defines a new state space Ω_2 determined by transitions between vertices adjoining scales 2 and 3, and so on. The outcome is a multiscale decomposition of the original process into a family of general jump processes reflecting the structure of the random walk at a particular time. This can then be used to obtain a multiscale decomposition of the original heat kernel [transition density], and hence, various quantities $\mathbb{E}\left[f(X_t)\right]$.

We summarize this procedure by an giving an example.

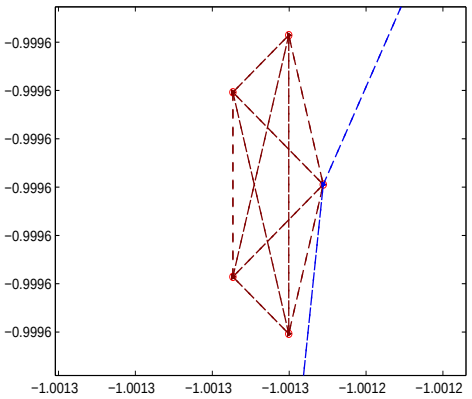
2.1.1 Example 1

Scale 1

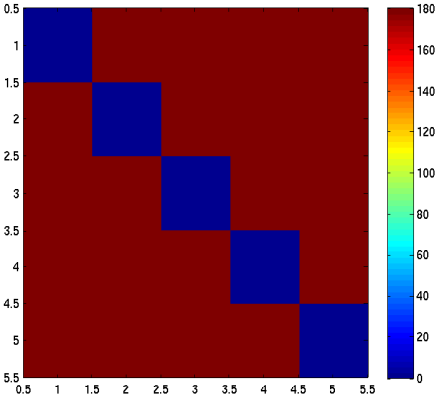
Consider a simple (undirected) network with three scales shown in figure 2.1. Scale 0 consists of each individual vertex as it appears in the graph, defining the state space at scale 0, Ω_0. At the first scale is a complete graph on 5 vertices with weights 180. Each of these are then attached to a decagon with weights 19 defining scale 2. Finally, each of these are attached to a dyadic tree with height 3 and weights 1 between vertices, defining scale 3.

Let W be the corresponding adjacency matrix of weights, and let $P = D^{-1}W$ where $D = diag(d_i)$, $d_i = \sum_j W(i,j)$ be the probability transition matrix for a discrete time Markov chain on the network, X_n. We can turn this Markov chain into a continuous time Markov process, X_t, by prescribing that that the process holds at each vertex for an amount of time $\tau \sim exp(1)$ before he jumps to another vertex determined by P.

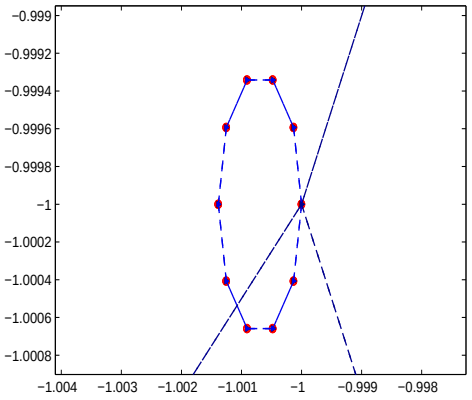
These weights were chosen for a reason. At any vertex, the random walker has a 95% chance of entering scale 1. For those vertices at the interface of scales 2 and 3, conditioned on not entering scale 1, the random walker has a 95% chance of entering scale 2.



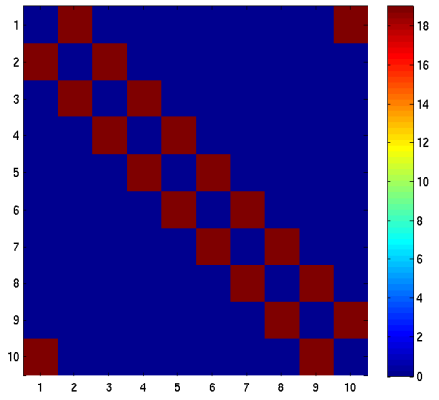
(a) Scale 1 with weights 180.



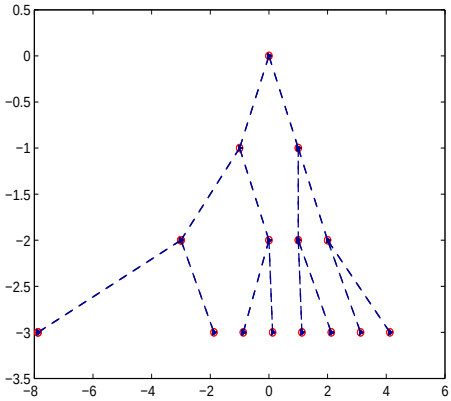
(b) Adjacency matrix for scale 1.



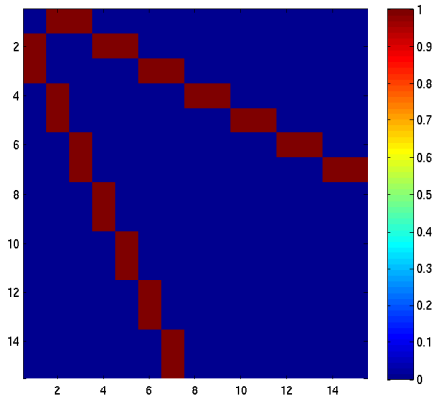
(c) Scale 2 with weights 19.



(d) Adjacency matrix for scale 2.



(e) Scale 3 with weights 1.



(f) Adjacency matrix for scale 3.

FIGURE 2.1: A 3-scale multiscale graph. (a) A complete graph on 5 vertices at scale 1. (b) The adjacency matrix for scale 1. (c) A decagon at scale 2. (d) The adjacency matrix for scale 2. (e) A tree at scale 3. (f) The adjacency matrix for scale 3.

With this setup then, $\{\mathcal{C}_{0,k}\}$ consists of exactly the vertices of the network as they are, $\{\mathcal{C}_{1,k}\}$ partitions the network into the set of 150 complete graphs of size 5, $\{\mathcal{C}_{2,k}\}$ into the set of 15 (decagons) $\times$ (complete 5 graphs), and $\{\mathcal{C}_{3,k}\}$ into the entire network itself.

Suppose for the sake of discussion that time is measured in seconds. Then, with this choice of weights, for a random walker starting at scale 1, it takes him an average of 95 seconds to leave scale 1, between $1.8239 \times 10^4 - 3.63 \times 10^4$ seconds to leave scale 2 to scale 3, and over 10^7 seconds to explore the entire network! Indeed then: the random walker explores scale 1 faster than scale 2, and explores scale 2 faster than scale 3.

We can quantify this notion of separation of scales [at least in the case of undirected networks] through the concept of *conductance* of a subset S:

$$\Phi(S) = \frac{\sum_{i\in S, j\in S^c} W(i,j)}{\min\left\{\sum_{i\in S,j} W(i,j), \sum_{i\in S^c,j} W(i,j)\right\}} := \frac{|E(S,S^c)|}{\min\{VolS, VolS^c\}}.$$

Intuitively, partitioning the network into sets with smaller conductance, we obtain a partition into coarser sets. In the above example,

$$\Phi\left(\mathcal{C}_{1,k}\right) = 1.1 \times 10^{-2}, \quad \Phi\left(\mathcal{C}_{2,k}\right) = 5.49 \times 10^{-5}.$$

Conceptually, $\Phi(S)$ is a bottleneck ratio measuring the ratio of the weights connecting S and S^c to the number of edges within S and S^c. The smaller this ratio is, the more balanced the volumes of S and S^c are, and the fewer weighted edges connect them, resulting in a set S with a strong bottleneck with respect to S^c, and a longer first exit time from S for either random process X_n or X_t.

Now that we've discussed how we define our separation of scale, we return to our example. If we could compress out scale 1, then we would obtain the dynamics of the random walker as he passes between vertices at scale 2. How shall we perform this compression?

The idea is as follows. Consider the geometry at the interface between scales 1 and 2, as in figures 2.1(a) and 2.1(c). We can collapse each copy of scale 1 onto its attaching boundary vertex which effectively represents that copy of scale 1. At this point, at least graphically, any random process would transition between vertices at scale 2 as desired. This new, homogenized, state space we call Ω_1.

In order to complete the compression then, we need to prescribe the rules 1-4 as in section 2.1 regarding hold times and transition probabilities. Of course, we should do this in a way that respects the dynamics of the original process.

(1) is taken care of immediately in this elementary example since with probability 1, X_t must enter and exit through the attaching boundary vertex.

For (2), the hold time is given by the random amount of time X_t takes to *enter and leave* scale 1. We emphasize that we force the walker to take a step into scale 1 before leaving in contrast to the first exit time from scale 1. The stronger the separation of scales, the closer these times will be with high probability anyways, but we choose our convention since if τ is the random time required to to leave after taking a step into scale 1, then the distribution of τ represents the time scale of exploring and leaving scale 1, as opposed to the first exit time where he may leave immediately.

For (3) and (4), once the random walker is about to leave scale 1 from $x \in \mathcal{C}_{1,k}$, he transitions to another boundary vertex $y \in \mathcal{C}_{1,k'}$ $k' \neq k$ with probability $P(x, y)$ conditioned on leaving $\mathcal{C}_{1,k}$ holding at x for a time $\tau \sim exp(1)$ before transitioning.

This then is the model we use to describe the random walk at the time scale at which he explores and leaves scale 1. We refer to this stochastic process as $X_t^{(1)}$. Any questions about it is encompassed in the heat kernel $p_t(x, y) = \mathbb{P}_x \left[X_t^{(1)} = y \right]$ describing the probability starting at x of being at y at time t. A familiar result from Markov processes shows that this process is *not* Markovian because the hold times

are not exponential random variables. Therefore, in order to compute $p_t(x, y)$, we must simulate sample paths and compute the fraction of them that are at y at time t starting from x.

In order to do so, we need two ingredients: a distribution of hold times to sample from approximating the original to a desired precision, and the transition probabilities of where to jump afterwards. The transition probabilities when simulating a sample path are easy enough to compute. What might not be so obvious is what the exit time distribution should be.

Proposition 2. *Consider a Markov process with exp(1) hold times and an irreducible transition matrix P. Let $\mathcal{C}$ be a connected proper subset of the state space, and $\gamma_\ell(t)$ be the density function of a $\Gamma(\ell, 1)$ random variable. If $Q = P(\mathcal{C}, \mathcal{C})$ then the density function of the entry-exit time starting from x and leaving through y having taken a step into $\mathcal{C}$ is given by,*

$$f(t) = \frac{1}{N} \sum_{n=2}^{\infty} \gamma_\ell(t) Q^{\ell-1}(x, y)$$

where $N = \sum_{\ell=1}^{\infty} Q^\ell(x, x) = [Q(I - Q)^{-1}](x, y)$.

Of course, sampling this distribution in our case would require keeping all $\left\{ \frac{Q^{\ell-1}(x,x)}{N} \right\}_{\ell=2}^{\infty}$ and sampling them before sampling the corresponding $\gamma_\ell(x)$. Instead, we use the following theorem to approximate $f(t)$ to a desired precision.

Proposition 3. *Under the same hypotheses and notation as proposition 2, suppose further that P is reversible [§2.4], in which case the eigenvalues of P, $\{\lambda_j\}$ are real, and that Q is irreducible and aperiodic. Order these eigenvalues in descending according to their absolute value. Define*

$$f_{x,n}(t) = \rho(n)\left[\frac{1}{\rho(n)}\sum_{\ell=2}^{n}\gamma_\ell(t)\frac{Q^{\ell-1}(x,x)}{N}\right] + (1-\rho(n))\sum_{\ell=0}^{\infty}\gamma_{n+1+\ell}(t)\lambda_1^\ell(1-\lambda_1)$$

where $\rho(n) = \sum_{\ell=2}^{n}\frac{Q^{\ell-1}(x,x)}{N}$. *Then,* $0 < \lambda_1 < 1$ *and given* $\epsilon > 0$,

$$\sup_A |f(A) - f_{x,n}(A)| = ||f - f_{x,n}||_{TV} < \epsilon$$

when $n > \frac{\log\left[\epsilon\sqrt{\frac{\sigma(i)}{\sigma(j)}}N(1-|\lambda_2|)\right]}{\log|\lambda_2|}$, *where* $\sigma(x) = \sum_{\pi(x)}\sum_{y\in\mathcal{C}}\pi(y)$ *and* $\pi(y)$ *is the stationary distribution of* P *at* y.

This proposition also has a nice interpretation of λ_1 for any such matrix Q: it is the (approximate) failure probability of leaving $\mathcal{C}$. For the proofs of these theorems, we refer the reader to theorem 1 proven in appendix A.1.

Using such a $f_{x,n}$ then, we can approximate to a desired accuracy the exit time distribution from scale 1 in our example, and hence approximate our model process at scale 1 to a given precision. We call $X_t^{(1)}$ the actual homogenized process with heat kernel $p_t^{(1)}(x,y)$, and $X_t^{(1,\epsilon)}$ the approximate one with hold times approximated and heat kernel $p_t^{(1,\epsilon)}(x,y)$.

We summarize this discussion at scale 1 in the following figures. Figure 2.2 presents the comparison of our estimator with $\epsilon = 0.01$ for the exit time distribution from scale 1 with the true distribution and compares how well it fits 20,000 samples. Figure 2.3(a) and 2.3(b) present the homogenized state space at scale 1, Ω_1.

Figure 2.4 computes $p_t^{(1,\epsilon)}$ at various times computed using 1,000 sample paths and $\epsilon = 10^{-6}$ in matching exit times from scale 1. It shows that at times at scales on the order of the exit time from scale $\mathcal{C}_{1,k}$ (100), the heat kernel of the homogenized process at scale 1 begins to diffuse along the decagons at scale 2, and at larger time scales, into the tree at sale 3.

In figure 2.5, we show how well $p_t^{(1,\epsilon)}$ for $\epsilon = 10^{-6}$ approximates the projection of the heat kernel of the original process at scale 0 by computing

$$\log \max_{x \in \Omega_1} \left|\left|(P_t A)(x, \cdot) - p_t^{(1,\epsilon)}(x, \cdot)\right|\right|_{TV}$$

as a function of t, in steps of size 100, using 1,000 sample paths of $X_t^{(1,\epsilon)}$ where $P_t = e^{(P-I)t}$ is the heat kernel at scale 0 and A is the $|\Omega_0| \times |\Omega_1|$ matrix for which

$$A(i,j) = \begin{cases} 1 & \text{if } i \in \mathcal{C}_{1,j} \\ 0 & \text{otherwise.} \end{cases}$$

The transition matrix $(P_t A)(x, y)$ for $x \in \Omega_1$ represents the probability starting at a boundary vertex that the random walker, *at scale 0*, is in $\mathcal{C}_{1,y}$ at time t.

Figure 2.6 illustrates the non-Markov property of $X_t^{(1,\epsilon)}$ with $\epsilon = 10^{-6}$ by computing

$$\log \max_{x \in \Omega_1} \left|\left|\left(p_{t_0}^{(1,\epsilon)}\right)^t (x, \cdot) - p_{t \cdot t_0}^{(1,\epsilon)}(x, \cdot)\right|\right|_{TV}$$

as a function of t for various choices of t_0. We note that the process is maximally Markovian when $t_0 \approx 100$ namely the mean transition time between the clusters $\mathcal{C}_{1,k}$.

Finally, we present how sensitive $X_t^{(1,\epsilon)}$ approximates $X_t^{(1)}$ in total variation distance as function of ϵ and the bin sizes Δt we look for jumps. Figure 2.7 plots

$$\left|\left|p_t^{(1)}(x, \cdot) - p_t^{(1,\epsilon)}(x, \cdot)\right|\right|_{TV}$$

as a function of t from $t = 0, \ldots, 60,000$ starting from the same point x, where we've partitioned $[0, 60, 000]$ into equal intervals of length $\Delta t = 10$. Here, proposition 3 was used with $\epsilon = 10^{-6}$ and $10,000$ sample paths. This shows that our approximation is indeed a good one.

To supplement this figure, we present figure 2.8, which plots

$$-\log \sup_{t \leq 20,000} \max_x \left|\left|p_t^{(1)}(x, \cdot) - p_t^{(1,\epsilon)}(x, \cdot)\right|\right|_{TV} = -\log \delta(\epsilon, \Delta t)$$

as a function of ϵ for choices of $\Delta t = 10^{-1}, 1$, and 10 using $1,000$ sample paths, where the maximum is taken over all starting positions x. The x-axis is $-\log_{10}\epsilon$ for $\epsilon = 10^{-1}, \ldots, 10^{-6}$ and the y-axis $-\log\delta$. This illustrates that as we increasing precision in ϵ, we require a smaller Δt for a given $\delta(\epsilon, \Delta t)$. In other words, for a given error in the actual and homogenized processes, choosing smaller ϵ causes the two processes to jump at closer times and we require a smaller Δt to detect these jumps.

Scale 2

Now that we've homogenized scale 1, how can we use it to homogenize scale 2, and compress out the decagons?

The idea is to use the *induced* partition of $\{\mathcal{C}_{2,k}\}$ on Ω_1. Recall that members of $\mathcal{C}_{2,k}$ consist of (decagons) $\times$ (complete 5 graphs). At scale 1, however, we've collapsed the complete graph on 5 vertices to the vertex attaching it to its decagon. Hence, members $\mathcal{C}_{2,k}$ transform into the set of decagons which partition Ω_1 into $\{\widetilde{\mathcal{C}}_{2,k}\}$.

Since we've constructed the homogenized process at scale 1 to respect dynamics of the underlying process, the random walker at scale 1 will explore $\widetilde{C}_{2,k}$ faster than he explores the entire homogenized state space, so that $\left\{\widetilde{C}_{2,k}\right\}$ is a multiscale partition of Ω_1.

Now, we have the a distribution of hold times and transition probabilities for the homogenized process at scale 1. To iterate the homogenization procedure on this state space then, and homogenized out the decagons, we perform the same steps as in 2.1.1. We then obtain a non-Markov process on the vertices of the tree at scale 3, the state space of which we call Ω_2.

There's a variant to propositions 2 and 3 for these clusters in theorem 1.6, however, theorems of this type won't hold in general when scale j isn't tangent to scale $j+1$. In lieu of this sort of analytic result then, we employ kernel density estimation

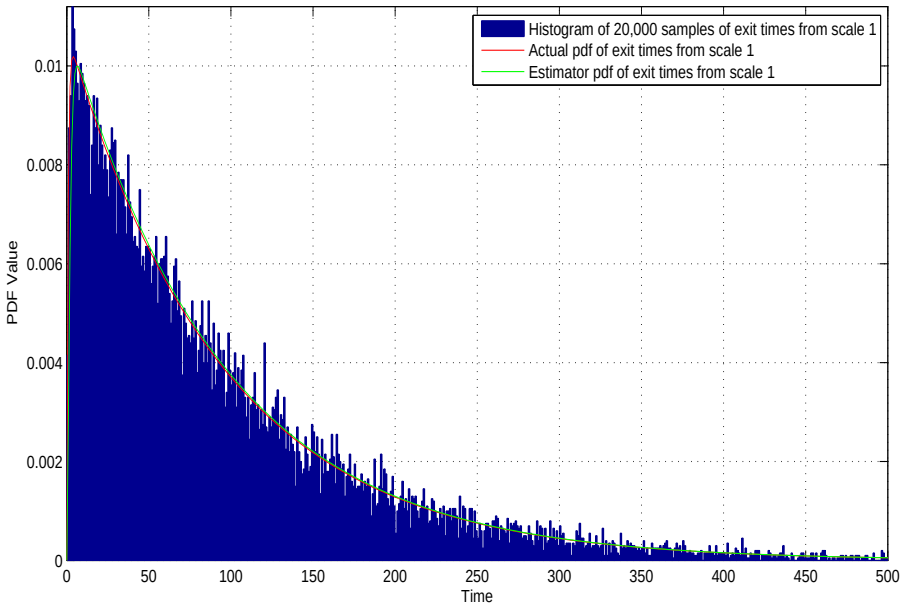
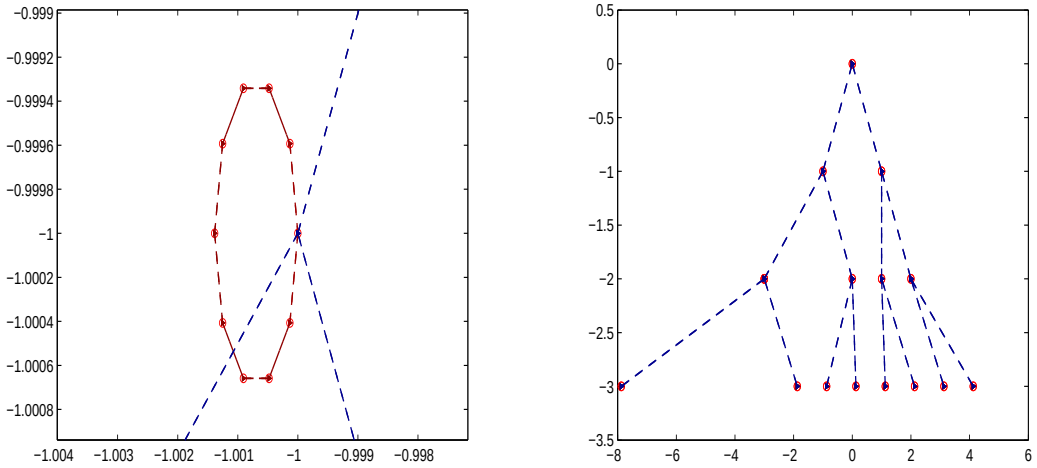
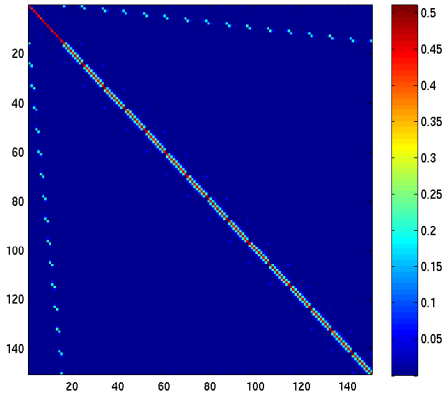


FIGURE 2.2: Exit time comparison from scale 1 in the 3-scale example. The histogram are actual samples of 20,000 exit times from scale 1. The red curve is the actual probability density function from proposition 2, and the green curve is our estimator of this density from proposition 3. For illustration purposes, we've chosen $\epsilon = 0.01$. Bin sizes were chosen to be 1 since hold times for vertices in scale 1 have a mean of 1.

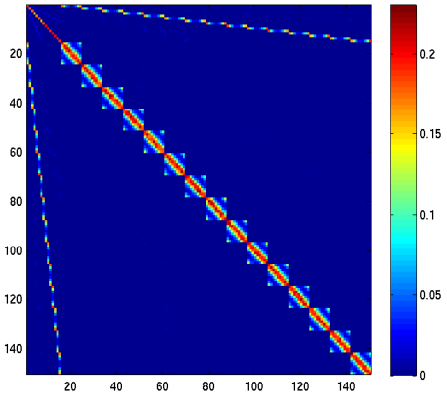


(a) (b)

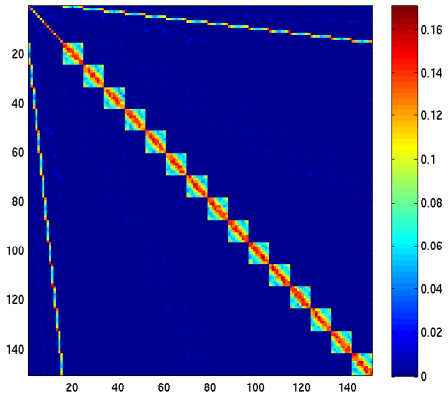
FIGURE 2.3: Vertices of the homogenized graph at scale 1. (a) Vertices of the decagon at scale 2. (b) Vertices of the tree at scale 3



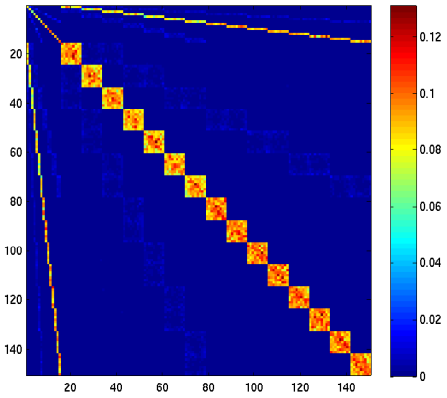
(a) $t = 10^2$.



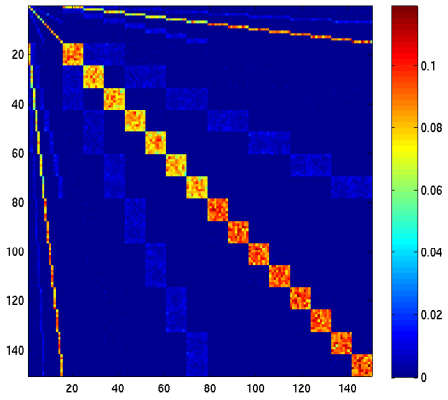
(b) $t = 4 \times 10^2$.



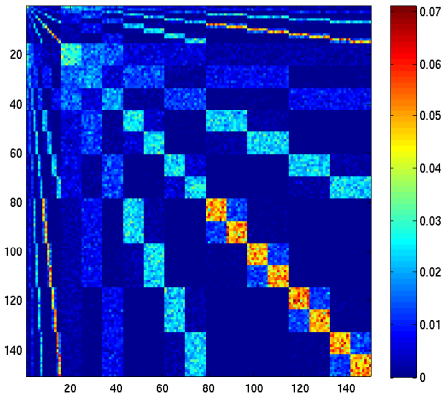
(c) $t = 8 \times 10^2$.



(d) $t = 2 \times 10^3$.



(e) $t = 4 \times 10^3$.



(f) $t = 4 \times 10^4$.

FIGURE 2.4: $p_t^{(1,\epsilon)}$ at various times t using 1,000 sample paths and $\epsilon = 10^{-6}$.

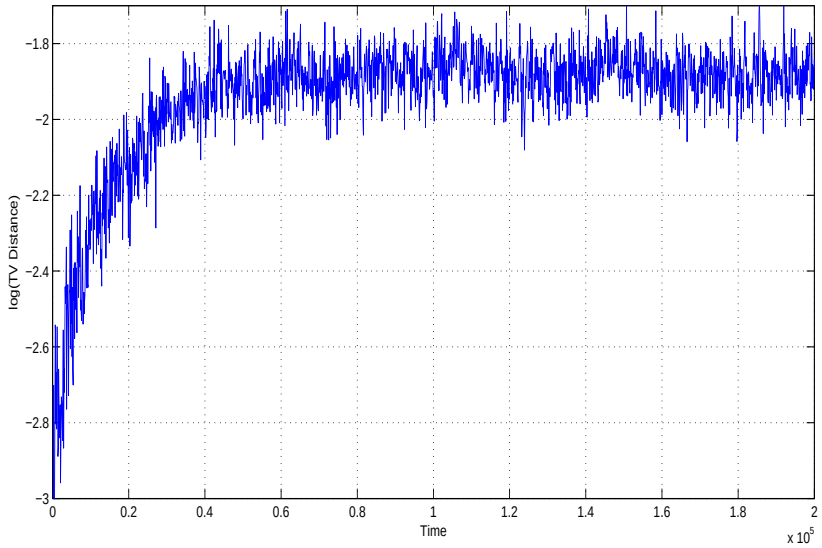


FIGURE 2.5: Approximation of the projection of the heat kernel at scale 0, $(P_t A)$, by $p_t^{(1,\epsilon)}$ using 1,000 sample paths and $\epsilon = 10^{-6}$. y-axis: $\log \max_{x \in \Omega_1} ||(P_t A)(x, \cdot) - p_t^{(1,\epsilon)}(x, \cdot)||_{TV}$. x-axis: time in units of 10^5.

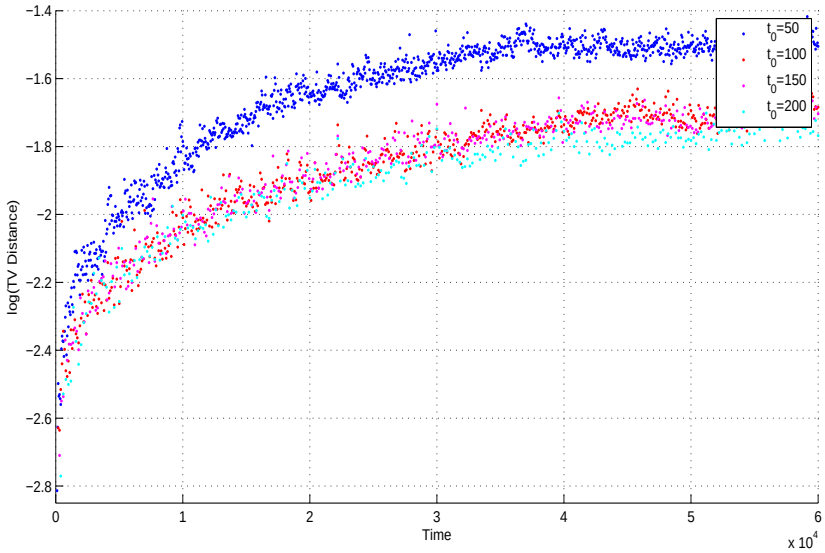


FIGURE 2.6: Plot of the non-Markov property of $p_t^{(1,\epsilon)}$ with $\epsilon = 10^{-6}$. It plots $\log \max_{x \in \Omega_1} \left|\left| \left(p_{t_0}^{(1,\epsilon)}\right)^t (x, \cdot) - p_{t \cdot t_0}^{(1,\epsilon)}(x, \cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using 1,000 sample paths.

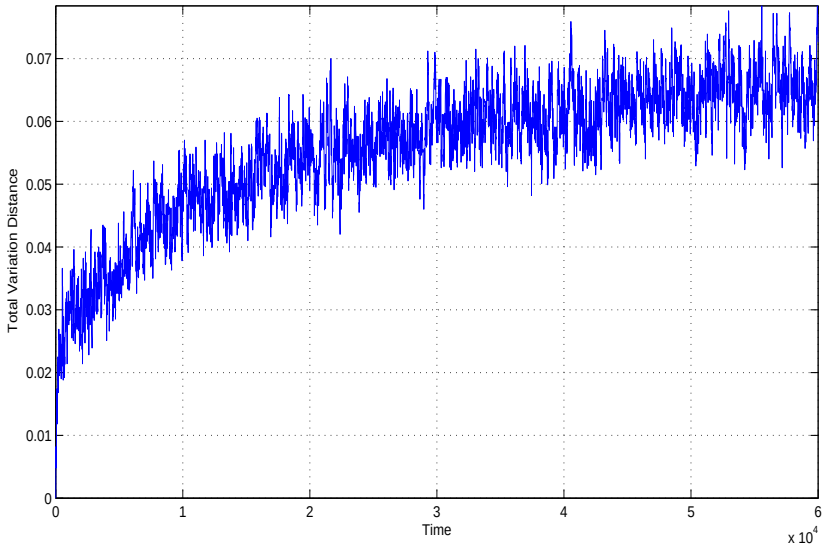


FIGURE 2.7: A plot of $\left|\left|p_t^{(1)}(x,\cdot) - p_t^{(1,\epsilon)}(x,\cdot)\right|\right|_{TV}$ with $\epsilon = 10^{-6}$ and bin sizes $\Delta t = 10$. $10,000$ sample paths starting from the same vertex of each process between times 0 and $60,000$ were used for this plot.

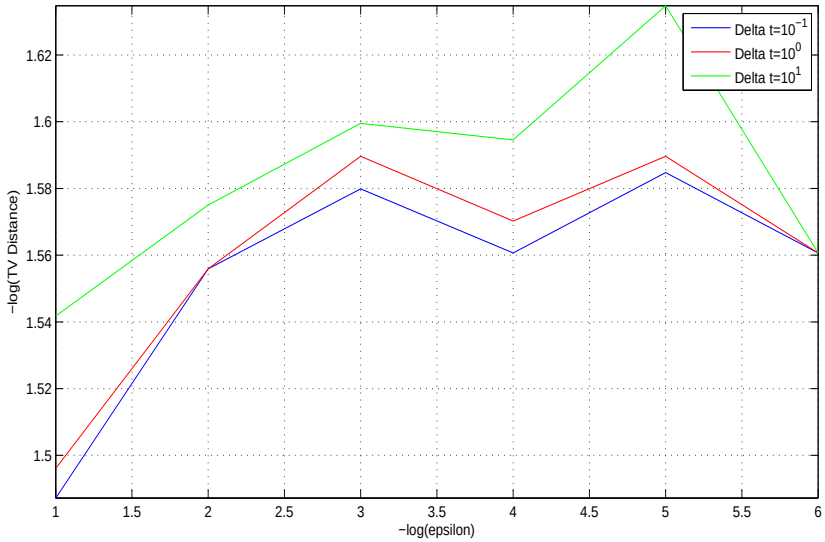


FIGURE 2.8: $-\log \sup_{t\leq 20,000} \max_x ||X_t - X_t^{(1,\epsilon)}||_{TV} = -\log \delta(\epsilon, \Delta t)$ as a function of $-\log \epsilon$ for various choices of Δt where ϵ is the uniform precision in matching the densities of hold times, and Δt is the bin size.

techniques with a Gaussian kernel to obtain an estimator of the exit time density from these clusters.

In the discussion that follows, $X_t^{(2)}$ is the actual homogenized process at scale 2 with heat kernel $p_t^{(2)}$, and $X_t^{(2,N)}$ is the approximate one with hold times matched using kernel density estimation on N samples of exit times from $\widetilde{C}_{2,k}$ *using* $X_t^{(1,\epsilon)}$ with (in the discussion below) $\epsilon = 10^{-6}$. The heat kernel for this latter process, we denote by $p_t^{(2,N)}$ or $p_t^{(2,N,\epsilon)}$ if we want to emphasize the dependence on ϵ.

In figure 2.9, we give a comparison of $10,000$ samples of the exit time distribution from $\widetilde{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ with $\epsilon = 10^{-6}$ to the actual distribution of exit times. Figure 2.10 presents the vertices upon homoginization of scale 2.

Figure 2.11 shows $p_t^{(2,N)}$ with $N = 10,000$ at various times for the homogenized process at scale 2. We notice that at times on the scale of leaving $\mathcal{C}_{2,k}$, the process begins to diffuse along the tree at scale 3.

In figure 2.12, we show how well $p_t^{(2,N)}$ with $N = 10,000$ approximates the projection of the heat kernel of the original process at scale 0 by computing

$$\log \max_{x \in \Omega_2} \left|\left| (P_t A)(x,\cdot) - p_t^{(2,N)}(x,\cdot) \right|\right|_{TV}$$

as a function of t, in steps of size $10,000$, where $P_t = e^{(P-I)t}$ is the heat kernel at scale 0 and A is the $|\Omega_0| \times |\Omega_2|$ matrix for which

$$A(i,j) = \begin{cases} 1 & \text{if } i \in \mathcal{C}_{2,j} \\ 0 & \text{otherwise.} \end{cases}$$

The transition matrix $(P_t A)(x,y)$ for $x \in \Omega_1$ represents the probability starting at a boundary vertex that the random walker, *at scale 0*, is in $\mathcal{C}_{2,y}$ at time t.

Figure 2.13 illustrates the non-Markov property of $X_t^{(2,N)}$ with $N = 10,000$ by computing

$$\log \max_{x \in \Omega_2} \left|\left| \left(p_{t_0}^{(2,N)}\right)^t (x,\cdot) - p_{t \cdot t_0}^{(2,N)}(x,\cdot) \right|\right|_{TV}$$

as a function of t for various choices of t_0. Again, we note that at $t_0 = 20000$, the time scale of transitions between $\mathcal{C}_{2,k}$ the process is maximally Markovian.

Having had tremendous success in this elementary example, we now proceed to the general case. Section 2.2 serves as background material for readers unfamiliar with Markov processes on graphs. Section 2.3 describes the general homogenization procedure for examples more complicated than this introductory one, and section 2.3.3 proves some guarantees of our homogenized processes. Section 2.4 presents further examples of this theory in more complicated examples.

2.2 Diffusions on Graphs

In the introductory example in section 2.1, we started out with a discrete time Markov chain defined naturally by the edge and vertex data of the network. When compressing out regions of the network that take a much smaller time scale to explore than to explore the entire network, we embedded the Markov chain into a continuous time, $exp(1)$, Markov process to analyze the dynamics of the random walker at the next scale without the burden of keeping track of how these unit steps transform at the next scale.

Regardless of whether one uses $exp(1)$ hold times, we'll always assume that the data comes to us as a continuous time Markov chain that we wish to homogenize. Therefore, in this section, we briefly review the theory of random processes on networks that possess a "memory forgetting property" which will naturally captures the geometry of the underlying graph. This exposition is taken from Norris (1997).

2.2.1 Discrete time Markov Chains

We first review the basic theory of Markov chains which is intuitively easier to understand.

A *Markov Chain* on the finite state space Ω is a random walk $\{X_n\}_{n=0}^{\infty}$ on Ω

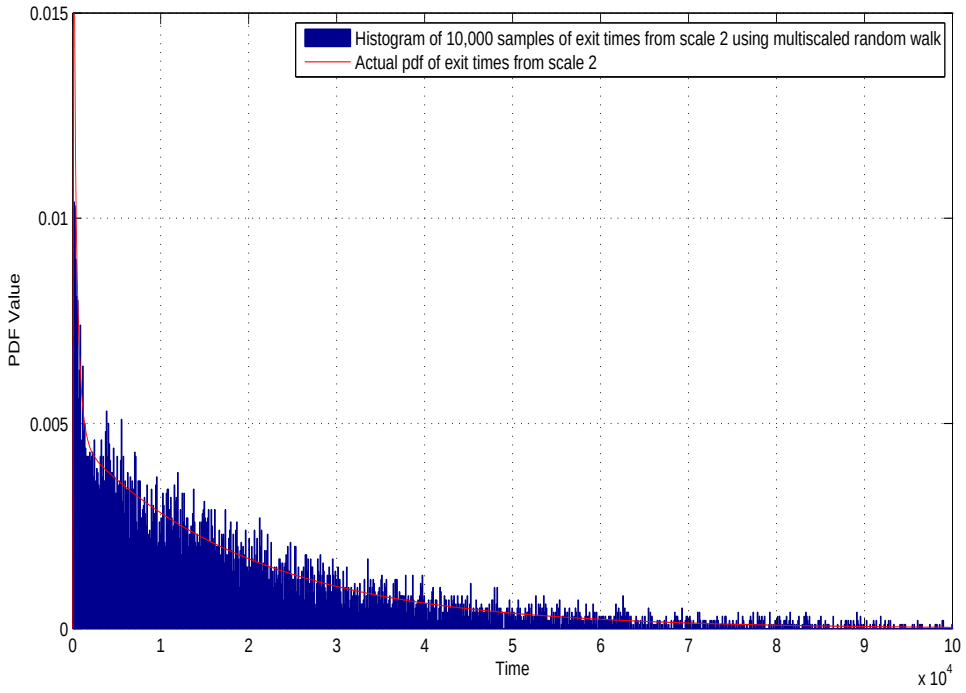


FIGURE 2.9: Exit time comparison from scale 2 in the 3-scale example. The histogram contain samples of 10,000 exit times from $\widetilde{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and $\epsilon = 10^{-6}$. The red curve is the actual probability density function from proposition 2. We've chosen bin sizes of 100 since the average amount of time leaving scale 1 (and hence transitioning between vertices at scale 1) has time 100.

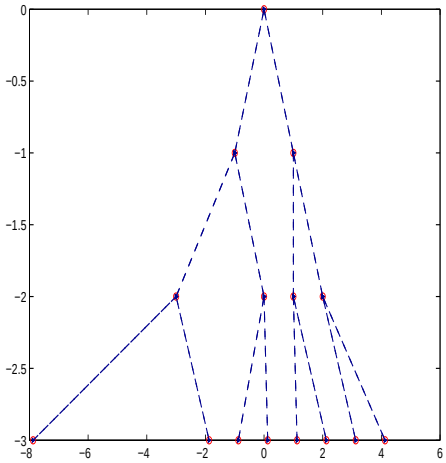
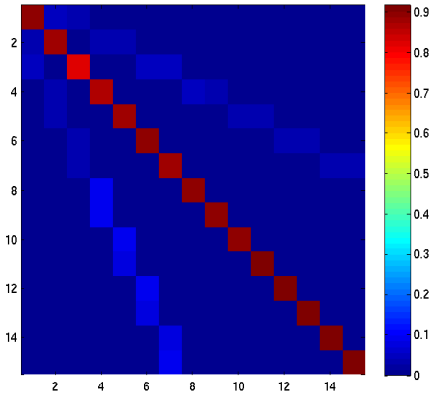
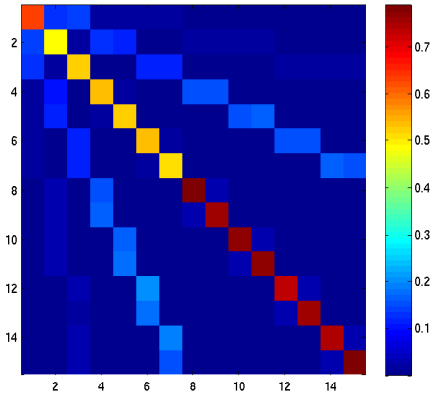


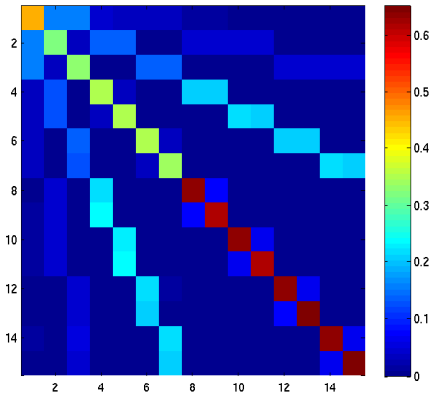
FIGURE 2.10: Vertices of the homogenized graph at scale 2.



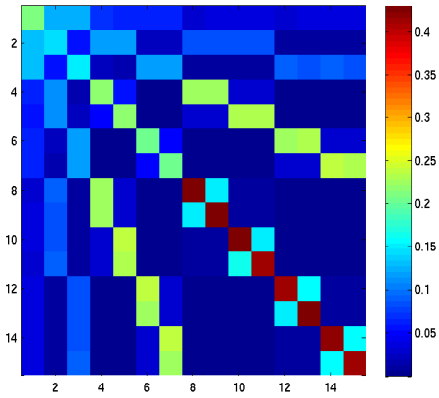
(a) $t = 1 \times 10^3$.



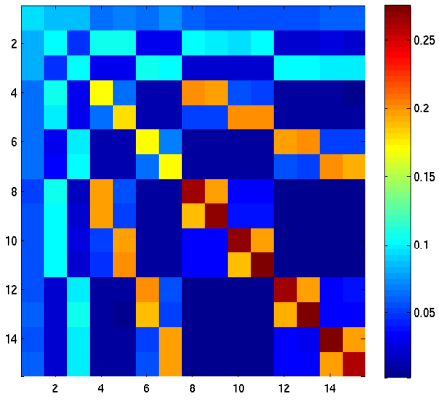
(b) $t = 1 \times 10^4$.



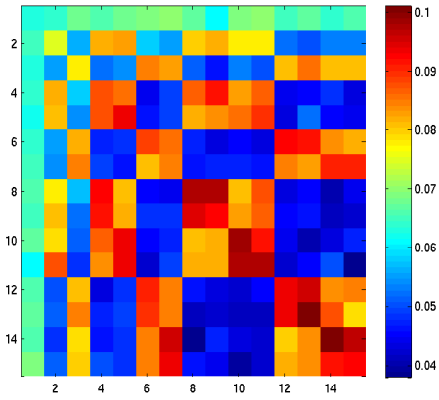
(c) $t = 2 \times 10^4$.



(d) $t = 5 \times 10^4$.



(e) $t = 1 \times 10^5$.



(f) $t = 5 \times 10^5$.

FIGURE 2.11: $p_t^{(2,N)}$ for $X_t^{(2,N)}$ with $N = 10,000$ at various times t.

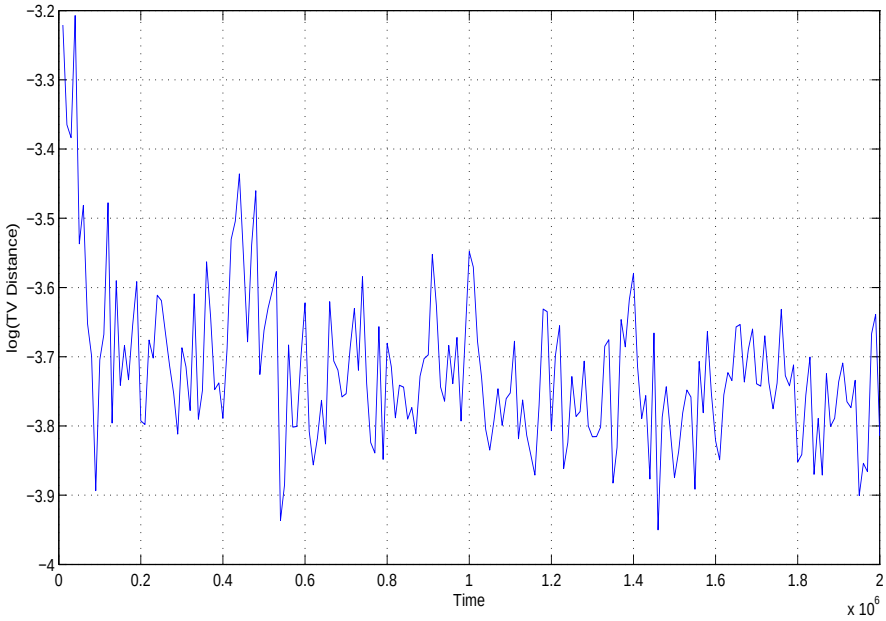


FIGURE 2.12: Approximation of the projection of the heat kernel at scale 0, $(P_t A)$, by $p_t^{(2,N)}$ using $N = 10,000$. y-axis: $\log \max_{x \in \Omega_2} ||(P_t A)(x,\cdot) - p_t^{(2,N)}(x,\cdot)||_{TV}$. x-axis: time in units of 10^6.

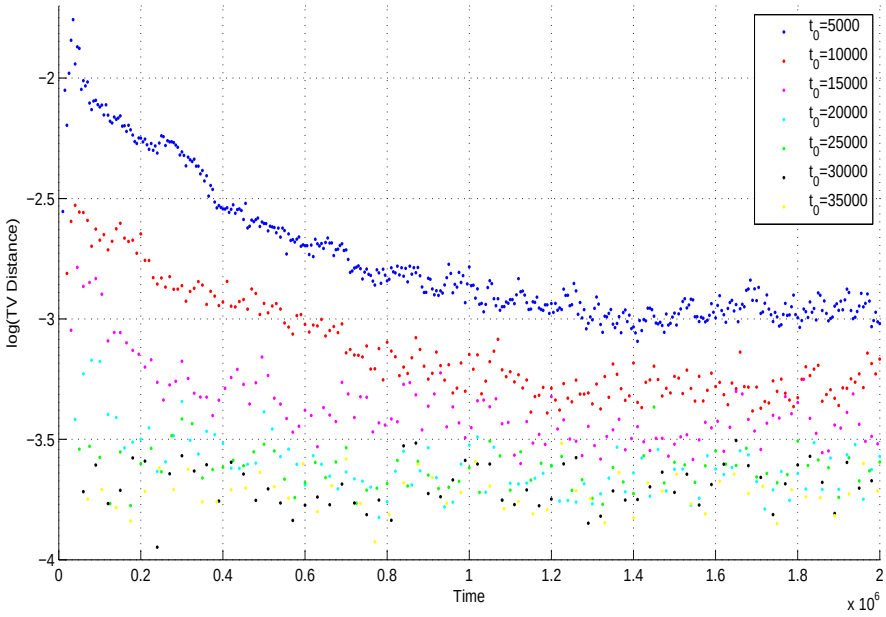


FIGURE 2.13: Plot of the non-Markov property of $p_t^{(2,N)}$ with $N = 10,000$. It plots $\log \max_{x \in \Omega_2} \left|\left| \left(p_{t_0}^{(2,N)}\right)^t (x,\cdot) - p_{t \cdot t_0}^{(2,N)}(x,\cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using $10,000$ sample paths.

satisfying the "memory forgetting" property, or *Markov property*,

$$\mathbb{P}\left[X_{n+1} = y | X_n = x, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1\right]$$
$$= \mathbb{P}\left[X_{n+1} = y | X_n = x\right] = P_n(x, y) \quad \forall n \in \mathbb{N}, \quad x, y, x_j \in \Omega. \quad (2.1)$$

The distributions $P_n(x, \cdot)$ are called the *transition probabilities* at time n. When $P_n = P$ for all n, the random walk is *time-homogeneous*, and throughout this paper, all Markov chains are assumed to be time-homogeneous unless otherwise specified.

Given an initial distribution $\pi(x) = \mathbb{P}\left[X_0 = x\right]$, interpreted to be a non-negative row vector indexed by Ω summing to one, we have that for a time-homogeneous chain:

$$\mathbb{P}\left[X_n = x\right] = (\pi P^n)(x). \quad (2.2)$$

The *stationary distribution* of P is an initial distribution π such that $\pi P = \pi$; that is, it's a non-negative left eigenvector of P with eigenvalue 1. Its interpretation is that if we start in this distribution, we stay there:

$$(\pi P^n)(x) = \pi(x) = (\pi P^m)(x) \Leftrightarrow \mathbb{P}\left[X_n = x\right] = \mathbb{P}\left[X_m = x\right] \quad \forall n, m \in \mathbb{N}. \quad (2.3)$$

Conditions for the existence of a unique stationary distribution are that P must be:

1. *Irreducible*: Given $x, y \in \Omega$ there exists a path of length t connecting them. (read: $\forall x, y \in \Omega$, there exists $n \in \mathbb{N}$ such that $P^n(x, y) > 0$).

2. *Aperiodic*: The random walk cannot oscillate between distributions supported on different subsets of Ω (read: $gcd\left\{n : P^n(x, x) > 0\right\} = 1$).

By the *ergodic theorem*, an irreducible, aperiodic chain has $\mu P^t \to \pi$ as $t \to \infty$ for any choice of initial distribution μ.

A Markov Chain P is *reversible* with respect to the stationary distribution π if

$$\pi(x)P(x, y) = \pi(y)P(y, x) \Leftrightarrow \mathbb{P}\left[X_{n-1} = x, X_n = y\right] = \mathbb{P}\left[X_{n-1} = y, X_n = x\right]. \quad (2.4)$$

Intuitively, this property says that if we start in a state distributed as π, the dynamics of running the chain forwards for a length of time is the same as running the chain backwards for the same length of time.

Finally, we note that if f is a column vector indexed by Ω, interpreted to be a function defined on Ω, then

$$(Pf)(x) = \sum_{y \in \Omega} P(x,y) f(y) = \sum_{y \in \Omega} \mathbb{P}\left[X_1 = y | X_0 = x\right] f(y) = \mathbb{E}_x\left[f(X_1)\right]. \tag{2.5}$$

2.2.2 Continuous Time Markov Processes

When discussing a (time homogenous) Markov chains, a transition matrix was sufficient to prescribe the dynamics since transitions between states were only allowed at unit time steps. In a continuous time Markov process, however, since time is now continuous, transitions between two steps can occur in infinitesimally small time intervals.

Instead of a transition matrix then, we must specify transition *rates* at which the continuous time process transitions between states in a small amount of time. This can be done by introducing a *Q-matrix*, $Q = (q_{i,j})$ on the finite state space Ω which satisfies three properties:

1. $0 \leq -q_{i,i} < \infty$,
2. $q_{i,j} \geq 0$, $i \neq j$,
3. $\sum_j q_{i,j} = 0$ for all i.

The idea is that the entries $q_{i,j}$ for $i \neq j$ specify the infinitesimal transition rates that the process takes between two states i and j. The diagonal entry is set so that $q(i) := q_{i,i} = \sum_j q_{i,j}$ and the rows of Q sum to 0.

The reason for the latter fact is that we'll be using the matrix $P(t) = \exp(Qt)$ which has its rows summing to 1 because the rows of Q sum to zero. By definition

of the Q matrix, the entries of $P(t)$ will also be nonnegative, so that its rows define probability distributions.

We summarize a few properties of this matrix $P(t)$:

Proposition 4. *Let Q be a Q-matrix on a finite state space Ω. If $P(t) = \exp(Qt)$ then,*

1. $P(s+t) = P(s)P(t)$,
2. $\frac{d}{dt}P(t) = QP(t) = P(t)Q$ *with* $P(0) = I$,

A *continuous time Markov process* $(X_t)_{t\geq 0}$ with Q matrix Q satisfies,

$$\mathbb{P}\left[X_{t_n+1}|X_{t_0} = i_0, \ldots, X_{t_n} = i_n\right] = p_{i_n,i_{n+1}}(t_{n+1} - t_n)$$

for all $n = 0, \ldots$ and times $0 \leq t_0 \leq \cdots \leq t_{n+1}$ and all states $i_0, \ldots, i_{n+1}$ where $p_{i,j}(t) = \left(e^{tQ}\right)(i,j)$.

Any random process satisfying this definition also has a (perhaps more intuitive) definition via *sample paths*. Given a Q matrix, define the jump matrix Π of Q by,

$$\pi_{i,j} = \begin{cases} \frac{q_{i,j}}{q_{i,i}} & \text{if } j \neq i \text{ and } q_{i,i} \neq 0 \\ 0 & \text{if } j \neq i \text{ and } q_{i,i} = 0 \end{cases}$$

$$\pi_{i,i} = \begin{cases} 0 & \text{if } q_i \neq 0 \\ 1 & \text{if } q_i = 0 \end{cases}$$

Now, let $\{Y_n\}_{n=0}^{\infty}$ be the Markov chain evolving via Π. Then, X_t is a continuous time Markov process with Q matrix Q if for each $n \geq 1$, conditional on $Y_0, \ldots, Y_{n-1}$, its hold times $S_1, \ldots, S_n$ are independent exponential random variables with parameters $q(Y_0), \ldots, q(Y_{n-1})$.

Another way of saying this is that when the process X_t jumps, where it jumps to is determined by Π. If T_n is the time of the n^{th} jump, then $Y_n = X_{T_n}$ is a Markov chain with transition matrix Π. The hold times of X_t conditional on $Y_0, \ldots, Y_{n-1}$, its hold times $S_1, \ldots, S_n$ are independent exponential random variables with parameters $q(Y_0), \ldots, q(Y_{n-1})$.

A stationary distribution of a continuous time process is defined in the same way as a discrete time one. If λ is any initial distribution (row vector with nonnegative entries summing to 1),

$$\mathbb{P}[X_t = j] = (\lambda P(t))(j).$$

λ is called a stationary distribution if

$$\lambda P(t) = \lambda$$

for any choice of t. It's easy to see that as long as Π defines an irreducible state space, that λ corresponds to a left eigenvector with eigenvalue 0 of Q. Note: aperiodicity of Π is not necessary for the existence of a stationary distribution λ for X_t. Also, if λ is the stationary distribution of X_t, then $\mu_i = \lambda_i q_i$ is the stationary distribution for Π.

We conclude this section with some results regarding $exp(1)$ continuous time Markov processes we'll use extensively. Part 6 considers the estimation of approximate hold times in the second scale homogenization of the example in section 2.1.

Theorem 1. *Let P be an irreducible, aperiodic, Markov Jump transition matrix for a continuous time process X_t with $exp(1)$ holding times on a finite state space V and stationary measure π. Define $Q = P|_{\mathcal{C},\mathcal{C}}$ and $N = (I - Q)^{-1}$ where $\mathcal{C} \subseteq V$ is an irreducible, proper subset. Assume Q is irreducible and aperiodic, and let i and j be two boundary vertices of $\mathcal{C}$. Then the following statements hold for X_t:*

1. *The probability of starting at i and leaving through j is given by*

$$N_{i,j} \sum_{k \in \mathcal{C}^c} P(j,k). \tag{2.6}$$

2. *If s_j are the number of jumps taken to leave through j,*

$$\mathbb{P}_i \left[s_j = \ell \right] = \frac{Q^{\ell-1}(i,j)}{N_{i,j}}, \quad \ell = 1, \ldots \tag{2.7}$$

3. *If τ_j is the time taken to leave $\mathcal{C}$ through j, then*

$$\mathbb{E}_i \left[\tau_j^k \right] = \frac{k!(I-Q)^{-(k+1)}(i,j)}{N_{i,j}}. \tag{2.8}$$

4. *In what follows, suppose further that P is reversible. For $x \in \mathcal{C}$, define*

$$\sigma(x) = \frac{\pi(x)}{\sum_{y \in \mathcal{C}} \pi(y)} \tag{2.9}$$

to be a probability measure on $\mathcal{C}$. Then there exists an orthonormal basis of Q of eigenvectors with real eigenvalues $\{\lambda_k, f_k\}$ with respect to the inner product

$$\langle f, g \rangle = \sum_{x \in \mathcal{C}} f(x) g(x) \sigma(x). \tag{2.10}$$

If $\{\lambda_k\}$ are sorted descending in absolute value and

$$g_{i,j,n} = \sum_{\ell=0}^{n} Q^{\ell}(i,j) + \sum_{\ell=n+1}^{\infty} f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell} = \sum_{\ell=0}^{n} Q^{\ell}(i,j) + f_1(i) f_1(j) \sigma(j) \frac{\lambda_1^{n+1}}{1-\lambda_1} \tag{2.11}$$

then, $0 < \lambda_1 < 1$ *has exactly one eigenvector,* $f_1(x) > 0$ *for all* $x \in \mathcal{C}$, $|\lambda_z| < \lambda_1$ *for* $z \neq 1$ *and*

$$|N_{i,j} - g_{i,j,n}| \leq \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^{n+1}}{1 - |\lambda_2|}. \tag{2.12}$$

5. *Finally, let* $f_{i,j}$ *be the density of* τ_j *starting at* i*. Then,*

$$f_{i,j}(x) = \sum_{\ell=1}^{\infty} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right]. \tag{2.13}$$

where $\gamma_\ell(x) \sim \Gamma(\ell, 1)$.

Set

$$f_{i,j,n}(x) = \rho_{i,j,n} \left[\frac{1}{\rho_{i,j,n}} \sum_{\ell=1}^{n} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] \right] + (1 - \rho_{i,j,n}) \sum_{\ell=0}^{\infty} \gamma_{n+1+\ell}(x) \lambda_1^\ell (1 - \lambda_1) \tag{2.14}$$

where $\rho_{i,j,n} = \sum_{\ell=1}^{n} \mathbb{P}_i \left[s_j = \ell \right]$ *. Then,*

$$||f_{i,j} - f_{i,j,n}||_{TV} \leq \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1 - |\lambda_2|)} \tag{2.15}$$

where $n \geq \min\{\ell : \mathbb{P}_i \left[s_j = \ell \right] \neq 0\}$.

6. *(Multiscaling) Let* $\phi_n = *_{i=1}^{n} \phi$ *be the n-fold convolution of* ϕ *and set*

$$f_{i,j}(x) = \sum_{\ell=1}^{\infty} \phi_\ell(x) \mathbb{P}_i \left[s_j = \ell \right],$$

$$f_{i,j,n}(x) = \rho_{i,j,n} \left[\frac{1}{\rho_{i,j,n}} \sum_{\ell=1}^{n} \phi_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] \right] + (1 - \rho_{i,j,n}) \sum_{\ell=0}^{\infty} \phi_{n+1+\ell}(x) \lambda_1^\ell (1 - \lambda_1)$$

Then,

$$||f_{i,j} - f_{i,j,n}||_{TV} \leq \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)}.$$

If

$$\tilde{f}_{i,j,n} = \rho_{i,j,n} \left[\frac{1}{\rho_{i,j,n}} \sum_{\ell=1}^{n} \tilde{\phi}_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] \right] + (1 - \rho_{i,j,n}) \sum_{\ell=0}^{\infty} \tilde{\phi}_{n+1+\ell}(x) \lambda_1^\ell (1-\lambda_1)$$

where $\tilde{\phi} = *_{i=1}^{n} \tilde{\phi}$, *and* $\tilde{\phi}$ *has*

$$\left|\left| \phi - \tilde{\phi} \right|\right|_{TV} < \epsilon$$

then,

$$\left|\left| f_{i,j} - \tilde{f}_{i,j,n} \right|\right|_{TV} \leq n\epsilon + \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)} + \frac{(1+\rho_{i,j,n})\epsilon(1-\epsilon)^n}{1-(1-\epsilon)\lambda_1}.$$

7. *Let* $f_{i,j}(x)$ *be the density of* τ_j *starting at* i *as in (A.8) for two vertices* $i, j \in \mathcal{C}$. *Let* $\{\lambda_z, f_z\}$ *be the eigenvalues and orthonormal basis of eigenvectors for* Q. *Then,*

$$\begin{aligned} f_{i,j}(x) = \sum_{\ell=1}^{\infty} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] &= \frac{1}{N_{i,j}} \sum_z f_z(i) f_z(j) \sigma(j) e^{-x(1-\lambda_z)} \\ &= \frac{1}{N_{i,j}} \sum_z \frac{f_z(i) f_z(j) \sigma(j)}{1-\lambda_z} (1-\lambda_z) e^{-x(1-\lambda_z)}. \quad (2.16) \end{aligned}$$

8. *Let* $\mathcal{Q}_t$ *be the process* X_t *constrained to* $\mathcal{C}$. *Then, for* $x, y \in \mathcal{C}$,

$$q_t(x,y) = \mathbb{P}_x \left[\mathcal{Q}_t = y \right] = \sum_{n=0}^{\infty} Q^n(x,y) \frac{e^{-t} t^n}{n!}.$$

2.3 Homogenization of Markov Processes

It's worth point out some of the key features that enabled our analysis in the example in section 2.1.1.

1. Scale k was tangent to scale $k+1$ in that there was only one attaching vertex.

2. All graphs at scale k were the same.

In real-world data, these assumptions won't hold, and we need a way to adapt our model to them. Along the way, we'll see that while we'll give up some of the strong analytic results as in section 2.1.1 regarding approximation of the hold times at the first scale, we'll be able to extend our techniques to a variety of data sets.

In the general setup then, we'll assume we have an undirected graph with a continuous time Markov process X_t defined on it with $exp(1)$ hold times with a jump transition matrix Π.

We still, however, assume that the network comes with a multiscale partition of the form $\{\mathcal{C}_{j,k}\}_{k\in\mathcal{K}_j}$ where j is a scale variables, and $k \in \mathcal{K}_j$ are indices at that scale indexing partition members. In our convention, larger j values correspond to a coarser scale, so that partition members become larger. Furthermore, we assume that

$$\mathcal{C}_{j+1,k} = \coprod_{k'} \mathcal{C}_{j,k'}$$

and that for $j = 0$, the partition consists of the vertices as they exist in the graph.

2.3.1 Scale 1

At scale 1, define

$$\partial^- \mathcal{C}_{1,k} \subset \mathcal{C}_{1,k}$$

such that $i \in \partial^- \mathcal{C}_{1,k}$ iff $\Pi(i,j) \neq 0$ for some $j \in \mathcal{C}_{1,k'}$ $k' \neq k$ and

$$\partial^+ \mathcal{C}_{1,k} \subset \mathcal{C}_{1,k}$$

such that $i \in \partial^+ \mathcal{C}_{1,k}$ iff $\Pi(j,i) \neq 0$ for some $j \in \mathcal{C}_{1,k'}$ $k' \neq k$.

We call $\partial^- \mathcal{C}_{1,k}$ *out boundary* vertices of $\mathcal{C}_{1,k}$ and $\partial^+ \mathcal{C}_{1,k}$ *in boundary* vertices of $\mathcal{C}_{1,k}$. While in an undirected network, these sets are the same, transitions between

$$\partial^- \mathcal{C}_{j,k} \to \partial^+ \mathcal{C}_{j',k'}$$

will be different than transitions between

$$\partial^+ \mathcal{C}_{j,k} \to \partial^- \mathcal{C}_{j',k'}$$

so for now, we make the distinction between these sets.

The homogenization of scale 1 will have the state space consisting of

$$\Omega_1 = \coprod_k \left\{ \partial^- \mathcal{C}_{1,k} \cup \partial^+ \mathcal{C}_{1,k} \right\}.$$

where an edge will connect $\partial^+ \mathcal{C}_{1,k}$ to $\partial^- \mathcal{C}_{1,k}$ and $\partial^- \mathcal{C}_{1,k}$ to $\partial^+ \mathcal{C}_{1,k'}$ for $k \neq k'$.

While the state space is (slightly) more complicated than the example in section 2.1.1, the rules governing the random process on this state space will be similar: we need to prescribe hold times and transition probabilities as in 1-4 in section 2.1. It turns out, this type of process has a name.

Definition 1. *A continuous-time stochastic process is called a semi-Markov process or "Markov renewal process" if the embedded jump chain (the discrete process registering what values the process takes) is a Markov chain, and where the holding times (time between jumps) are random variables with any distribution, whose distribution function may depend on the two states between which the move is made.*

First, we have to prescribe transitions from $i \in \partial^+ \mathcal{C}_{1,k}$ to $j \in \partial^- \mathcal{C}_{1,k}$. It turns out, however, that this is given by a variant theorem 1.1, since the proof of this statement never involves the hold times. Transitions between such vertices i and j are given

by,

$$\frac{1}{Z_i} N_{i,j} \sum_{k \in \mathcal{C}_{1,k}^c} \Pi(j,k)$$

where now, $N = Q(I-Q)^{-1}$ (since we force the random walker to take a step into $\mathcal{C}_{1,k}$ before he leaves), and Z_i is the renormalization constant. The hold times are estimated via kernel density estimation over samples of N paths starting at i and ending at j.

Transitions between $i \in \partial^- \mathcal{C}_{1,k}$ and $j \in \partial^+ \mathcal{C}_{1,k'}$ for $k \neq k'$ are given by,

$$\frac{1}{Z_i} \Pi(i,j)$$

where Z_i is the renormalization constant, since the random walker transitions from i to j, *and* enters the set $\mathcal{C}_{1,k'}$. The hold time for these transitions as such i are those from X_t, namely $exp(1)$.

We can store these transition probabilities in a $|\Omega_1| \times |\Omega_1|$ transition matrix, Π_1. This transition matrix will be *off-block diagonal* since we forbid transitions within the same type of boundary vertex. With this and the holds times, we have prescribed our semi-Markov process at scale 1.

2.3.2 Multiple Scales

Now that we have our semi-Markov process at scale 1 with transitions given by Π_1 and approximate hold times, how do we iterate the process?

The partition $\{\mathcal{C}_{2,k}\}$ induces a partition on Ω_1 in a natural way. Since

$$\mathcal{C}_{2,k} = \coprod_{k'} \mathcal{C}_{1,k'}$$

the induced partition on Ω_1 will consist of sets,

$$\mathcal{C}'_{2,k} = \coprod_{k} \left\{ \partial^- \mathcal{C}_{1,k} \cup \partial^+ \mathcal{C}_{1,k} \right\}.$$

We can thus repeat the procedure in section 2.3.1. We may continue this procedure in j until we exhaust the multiscale partition $\{\mathcal{C}_{j,k}\}_{k\in\mathcal{K}_j}$.

2.3.3 Guarantees

While certainly valid models for how the random processes move between scales, one may ask for theoretical guarantees as to how well these processes are approximating the original ones. We quantify these errors in the following theorems.

First, we present a simplified version of theorem 6 in Chapter 3 which helps in estimating the heat kernel of the homogenized processes at time t. We omit the proof of it, since it follows immediately from the proof of theorem 6 in section A.1.

Proposition 5. *Let $p_{n,t}$ be the empirical heat kernel of the homogenized process obtained from n sample paths, and let p_t denote the true heat kernel. Define the total variation distance between probability mass functions p and q by*

$$||p-q||_{TV} = \frac{1}{2}\sum_{i=1}^{N}|p(i)-q(i)|.$$

Then,

$$\mathbb{P}\left[\max_x ||p_{n,t}(x,\cdot)-p_t(x\cdot)||_{TV} > \epsilon\right] \leq 2e^{-\frac{n\epsilon^2}{2N}}$$

where N is size of the homogenized state space.

The next theorem regards reconstruction of the heat kernel at scale 0 from the heat kernel of the homogenized processes at scale 1.

Theorem 2. *Let $x \in \mathcal{C}_1$ and $y \in \mathcal{C}_2$ be in two clusters and $p_t(x,y) = \mathbb{P}_x[Z_t = y]$ where Z_t is a Markov process with exp(1) hold times and jump matrix P on a finite state space V with boundary vertices ∂V, and $Q_i = P_{\mathcal{C}_i,\mathcal{C}_i}$.*

Let $q_t^{(i)}$ be $|\mathcal{C}_i| \times |\mathcal{C}_i|$ matrices defined by

$$q_t^{(i)}(i,j) = \sum_{n=0}^{\infty} Q_i^n(i,j)\frac{e^{-t}t^n}{n!},$$

r_t be a $|\mathcal{C}_1| \times |\partial V|$ matrix with entries

$$r_t(x,a) = \sum_{n=0}^{\infty}(Q_1^n P)(x,a)\frac{e^{-t}t^n}{n!}$$

and s_t be a $|\partial V| \times |\partial V|$ matrix with entries:

$$s_t(\omega_0,\omega_n) = \sum_{n=1}^{\infty}\sum_{|\omega|=n} \mathbb{P}_{\omega_0}\left[Z_{T_1} = \omega_1\right]\cdots\mathbb{P}_{\omega_{n-1}}\left[Z_{T_n} = \omega_n\right] dT_n(t)(\omega)$$

where $dT_n(x)(\omega)$ is the density of T_n, the time of the n^{th} jump on the cluster-cluster process defined pathwise, and

$$\mathbb{P}_{\omega_{i-1}}\left[Z_{T_i} = \omega_i\right] = \sum_{j\in\partial\mathcal{C}_{\omega_{i-1}}} N_{\omega_{i-1},j}P(j,\omega_i)$$

where $N = (I-Q)^{-1}$ and $Q = P|_{\mathcal{C},\mathcal{C}}$ for a cluster $\mathcal{C}$.

Then,

$$p_t(x,y) = q_t^{(1)}(x,y) + (rsq^{(2)})(t,x,y) \tag{2.17}$$

where matrix products are convolutions.

Unfortunately, theorem 2 doesn't multiscale, in that we can reconstruct the heat kernel at scale j by using the process at scale $j+1$ for $j > 0$, since we would have to use the strong Markov property for the continuous time process at scale j.

Finally, we give a theorem bounding the error we incur by approximating a semi-Markov process by another whose hold times agree in total variation within ϵ.

Theorem 3. *Let X_t and $\tilde{X}_t$ be two semi-Markov processes on a finite state space V with the same transition matrix P that is irreducible and aperiodic. Let $p_t(x,\cdot)$ and $\tilde{p}_t(x,\cdot)$ be the laws of X_t and $\tilde{X}_t$, respectively, starting at $x \in V$. Suppose that the hold times of X_t and $\tilde{X}_t$ agree in total variation distance within ϵ. Then, there exists constants $C > 0$ and $\alpha \in (0,1)$ computable completely in terms of P such that*

$$\max_x ||p_t(x,\cdot) - \tilde{p}_t(x,\cdot)||_{TV} \leq \epsilon \left(\frac{1}{2} + C \frac{1+\alpha}{(1-\alpha)(1-(1-\epsilon)\alpha)} \right).$$

One can ask if this theorem localizes when we restrict the laws of p_t and $\tilde{p}_t$ to a cluster $\mathcal{C} \subseteq V$. Intuitively, this makes sense if we don't want to ask questions about the homogenized process beyond some fixed time.

Theorem 4. *Let X_t and $\tilde{X}_t$ be two semi-Markov processes on a finite state space V with the same transition matrix P that is irreducible and aperiodic. Let $\mathcal{C} \subseteq V$ be a subset of the state space on which the jump chain of X_t and $\tilde{X}_t$ are irreducible and aperiodic.*

Define $q_t(x,y)$ and $\tilde{q}_t(x,y)$ to be the laws of the process X_t and $\tilde{X}_t$ restricted to $x, y \in \mathcal{C}$. We can turn q_t and $\tilde{q}_t$ into probability measures, p_t and $\tilde{p}_t$, by closing the system $\mathcal{C}$ by a coffin state ∂ representing when the processes X_t and $\tilde{X}_t$ leaving $\mathcal{C}$. Suppose that the hold times of X_t and $\tilde{X}_t$ agree in total variation distance within ϵ. Then, there exists $C > 0$ and $\alpha \in (0,1)$ computable complete in terms of $Q = P|_{\mathcal{C},\mathcal{C}}$ such that

$$\max_x ||p_t(x,\cdot) - \tilde{p}_t(x,\cdot)||_{TV} \leq \epsilon \left(\frac{1}{2} + C \frac{1+\alpha}{(1-\alpha)(1-(1-\epsilon)\alpha)} \right).$$

As an added result:

Proposition 6. *Let $p_{n,t}$ be the empirical heat kernel of the homogenized process obtained from n sample paths, and let p_t denote the true heat kernel defined on a state space $\tilde{V} = \mathcal{C} \cup \{\partial\}$ as in Theorem 4. Then,*

$$\mathbb{P}\left[\max_x ||p_{n,t}(x,\cdot) - p_t(x\cdot)||_{TV} > \epsilon\right] \leq 2e^{-\frac{n\epsilon^2}{2(N+1)}}$$

where $N = |\mathcal{C}|$.

All proofs are given in section A.1.

2.4 Further Applications

In this section, we present applications of our theory to a variety of synthetic data sets that naturally generalize example 1 from the introduction.

We maintain the same notation as in that example. Ω_j denotes the state space of the homogenized process at scale j on which we have our semi-Markov process $X_t^{(j)}$, taking $X_t^{(0)} = X_t$ to be the continuous time, $exp(1)$, Markov process at scale 0. These are equipped with their heat kernels $p_t^{(j)}(a,b) = \mathbb{P}_a\left[X_t^{(j)} = b\right]$.

We approximate $X_t^{(1)}$ by $X_t^{(1,\epsilon)}$ where we have matched hold times at each $x \in \Omega_1$ in total variation within ϵ. The process $X_t^{(2,N)}$ denotes the multiscale approximation to $X_t^{(2)}$ by homogenizing $X_t^{(1,\epsilon)}$ on Ω_1 and using kernel density estimation on N samples of exit times to infer the exit time distribution.

In these processes, we take $\epsilon = 10^{-6}$ and $N = 10,000$ unless otherwise specified.

We recall here for convenience that in example 1, we chose a complete graph on 5 vertices at scale 1 with weights 180 which had a mean exit time of 95 for $\mathcal{C}_{1,k}$, and a decagon at scale 2 with weights 19 which had a mean exit time from $\mathcal{C}_{2,k}$=(decagon)x(complete graph on 5 vertices) between $1.8239 \times 10^4 - 3.63 \times 10^4$.

We note that we reproduce many of the figures in example 1, but omit the fitting of hold times at scale 1, since these were already proven to be in total variation to within ϵ of what they should be.

2.4.1 Example 2: Change of Weights

Here, we present example 1, except now we randomly change the weights at each scale to allow for cases whose separation of scales might vary across clusters. Scale 1 can now have weights 90 or 180 and scale 2 can have weights 10 or 19. This causes the mean exit time from scale 1 to vary among 48, 91, 95 and 180 depending on the values of the weights adjoining scale 1 to scale 2. The mean exit time from $\mathcal{C}_{2,k}$ will now vary around 10^4.

Figure 2.14 shows a comparison between the histogram of $10,000$ samples of the exit times from $\mathcal{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and the true distribution. By inspection we note that these distributions are rather close.

Next, in figure 2.15 we show values of the heat kernel $p_t^{(1,\epsilon)}$ for various values of t. We note that at $t = 400$ one can see mass flowing at different rates within each decagon (the block diagonals) which continue until time $40,000$ when the process begins to explore the tree.

In figure 2.16, we show how well $p_t^{(1,\epsilon)}$ approximates the projection of the heat kernel of the original process at scale 0 by computing

$$\log \left|\left|(P_t A)(x,\cdot) - p_t^{(1,\epsilon)}(x,\cdot)\right|\right|_{TV}$$

at a fixed vertex x as a function of t in steps of size 100 using 1,000 sample paths of $X_t^{(1,\epsilon)}$, where $P_t = e^{(P-I)t}$ is the heat kernel at scale 0 and A is the $|\Omega_0| \times |\Omega_1|$ matrix for which

$$A(i,j) = \begin{cases} 1 & \text{if } i \in \mathcal{C}_{1,j} \\ 0 & \text{otherwise.} \end{cases}$$

The transition matrix $(P_t A)(x,y)$ for $x \in \Omega_1$ represents the probability starting at a boundary vertex that the random walker, *at scale 0*, is in $\mathcal{C}_{1,y}$ at time t.

Figure 2.17 illustrates the non-Markov property of $X_t^{(1,\epsilon)}$ by computing

$$\log \max_{x \in \Omega_1} \left|\left| \left(p_{t_0}^{(1,\epsilon)}\right)^t (x,\cdot) - p_{t \cdot t_0}^{(1,\epsilon)}(x,\cdot) \right|\right|_{TV}$$

as a function of t for various choices of t_0. We note that at $t_0 \approx 150$ the curves settle, and the process is maximally Markovian.

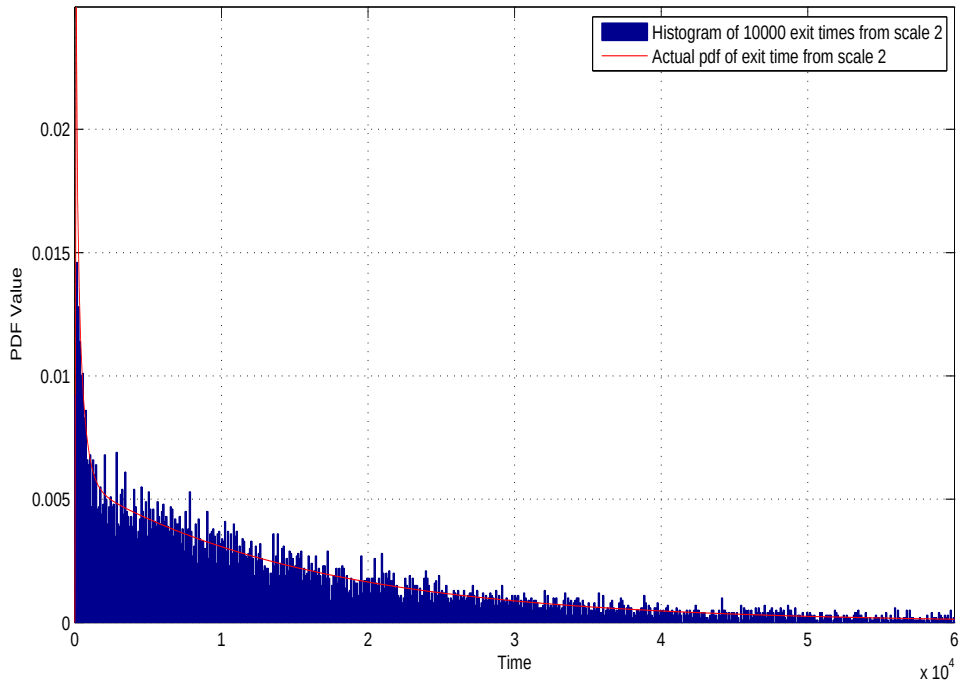
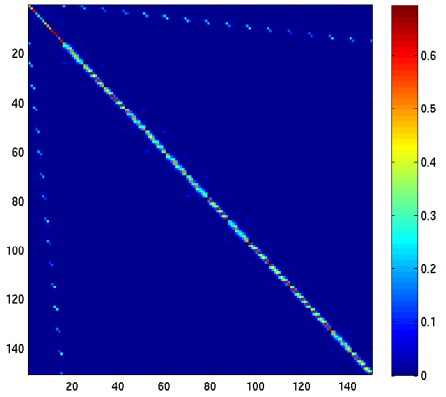
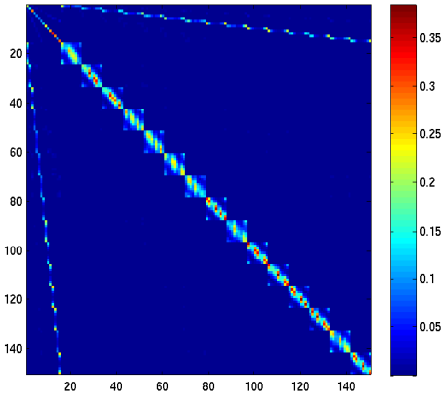


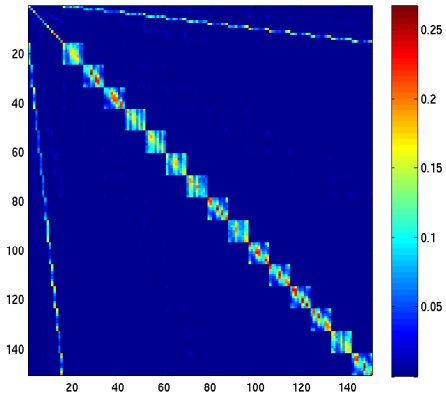
FIGURE 2.14: Exit time comparison from scale 2 in example 2. The histogram contain samples of 10,000 exit times from $\widetilde{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and $\epsilon = 10^{-6}$. The red curve is the actual probability density function from proposition 2. We've chosen bin sizes of 100 since the average amount of time leaving scale 1 (and hence transitioning between vertices at scale 1) has time approximately 100.



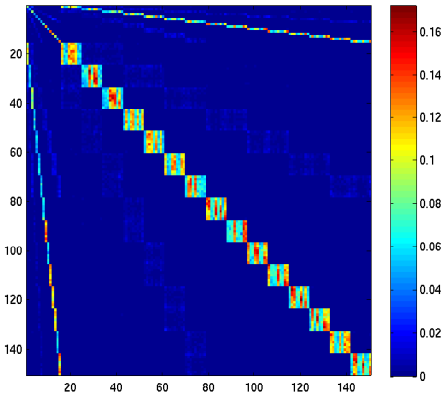
(a) $t = 10^2$.



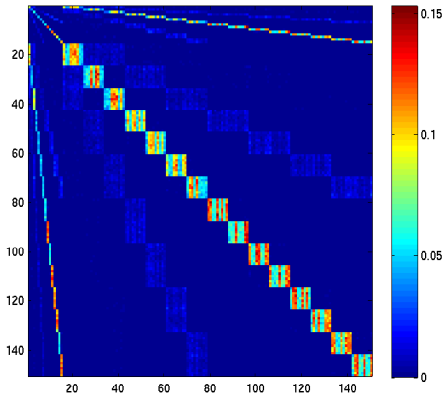
(b) $t = 4 \times 10^2$.



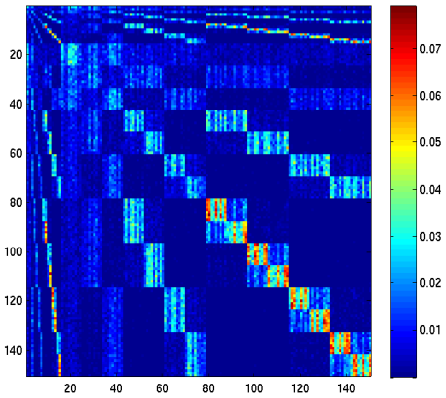
(c) $t = 8 \times 10^2$.



(d) $t = 2 \times 10^3$.



(e) $t = 4 \times 10^3$.



(f) $t = 4 \times 10^4$.

FIGURE 2.15: $p_t^{(1,\epsilon)}$ for $X_t^{(1,\epsilon)}$ for example 2 at various times t using 1,000 sample paths and $\epsilon = 10^{-6}$.

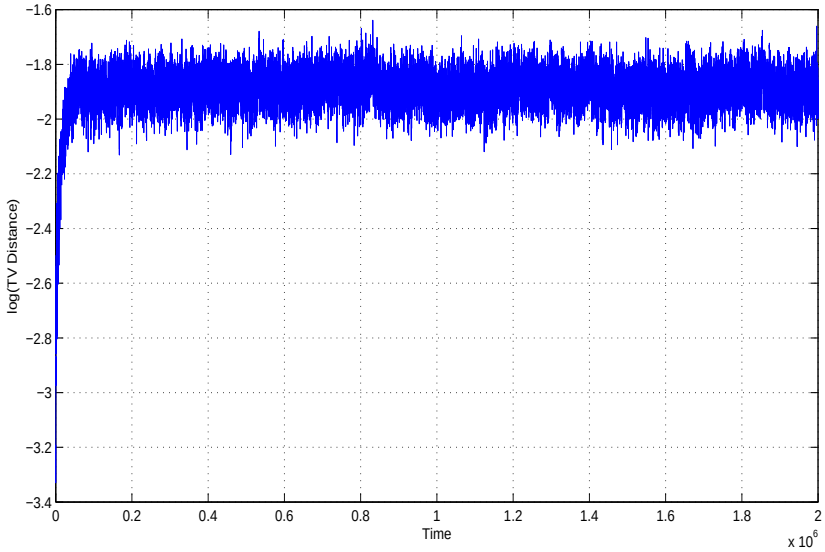


FIGURE 2.16: Approximation of the projection of the heat kernel at scale 0, $(P_t A)$, by $p_t^{(1,\epsilon)}$ for example 2 using 1,000 sample paths and $\epsilon = 10^{-6}$. y-axis: $\log ||(P_t A)(x,\cdot) - p_t^{(1,\epsilon)}(x,\cdot)||_{TV}$ for a fixed x. x-axis: time in units of 100.

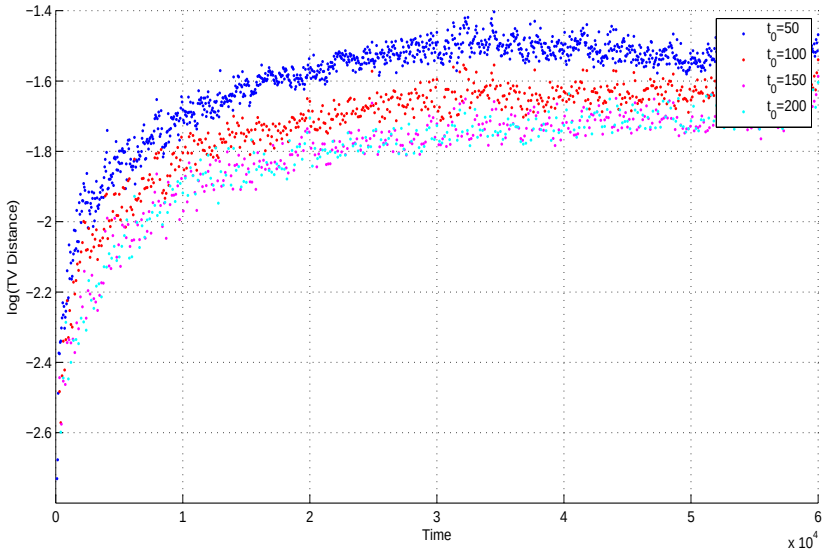


FIGURE 2.17: Plot of the non-Markov property of $p_t^{(1,\epsilon)}$ for example 2 with $\epsilon = 10^{-6}$. It plots $\log \max_{x\in\Omega_1} \left|\left| \left(p_{t_0}^{(1,\epsilon)}\right)^t (x,\cdot) - p_{t\cdot t_0}^{(1,\epsilon)}(x,\cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using 1,000 sample paths.

At scale 2, figure 2.18 shows $p_t^{(2,N)}$ at various times for the homogenized process at scale 2. We notice that due to varying separation of scales, mass flow at $t = 1,000$ is no longer symmetric as it was in example 1, and continues through time.

In figure 2.19, we show how well $p_t^{(2,N)}$ approximates the projection of the heat kernel of the original process at scale 0 by computing

$$\log \left|\left|(P_t A)(x,\cdot) - p_t^{(2,N)}(x,\cdot)\right|\right|_{TV}$$

from a fixed point x as a function of t in steps of size $10,000$, where $P_t = e^{(P-I)t}$ is the heat kernel at scale 0 and A is the $|\Omega_0| \times |\Omega_2|$ matrix for which

$$A(i,j) = \begin{cases} 1 & \text{if } i \in \mathcal{C}_{2,j} \\ 0 & \text{otherwise.} \end{cases}$$

The transition matrix $(P_t A)(x,y)$ for $x \in \Omega_1$ represents the probability starting at a boundary vertex that the random walker, *at scale 0*, is in $\mathcal{C}_{2,y}$ at time t.

Finally, figure 2.20 illustrates the non-Markov property of $X_t^{(2,N)}$ by computing

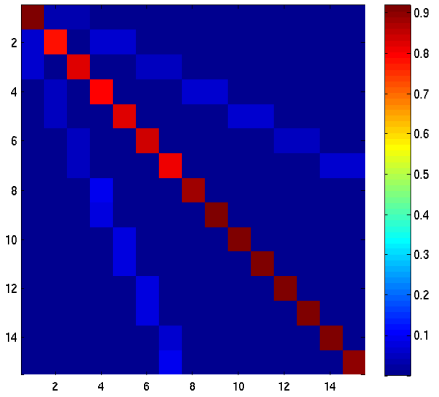
$$\log \max_{x \in \Omega_2} \left|\left|\left(p_{t_0}^{(2,N)}\right)^t (x,\cdot) - p_{t \cdot t_0}^{(2,N)}(x,\cdot)\right|\right|_{TV}$$

as a function of t for various choices of t_0.

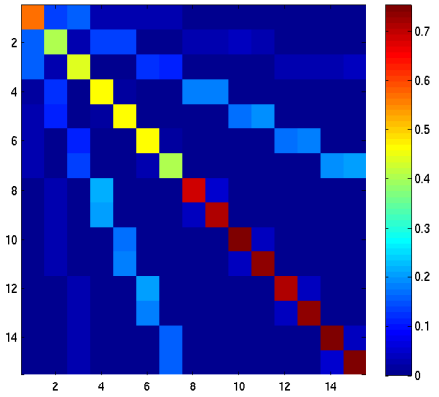
2.4.2 Example 3: Change of Geometries

Next, we allow for different geometries at each scale, but leave the weight values at each scale unchanged (scale 1: 180, scale 2: 19).

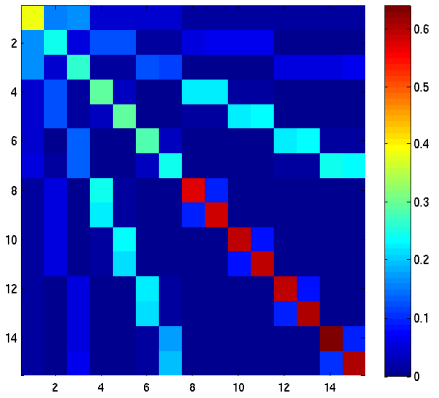
Scale 1 can now have either a complete graph on 5 vertices or a pentagon. Scale 2 can not have either a complete graph on 10 vertices or a decagon. The mean exit time from scale 1 can now vary from 21, 22, 48 and 95, while the mean exit time from $\mathcal{C}_{2,k}$ will vary around 10^4.



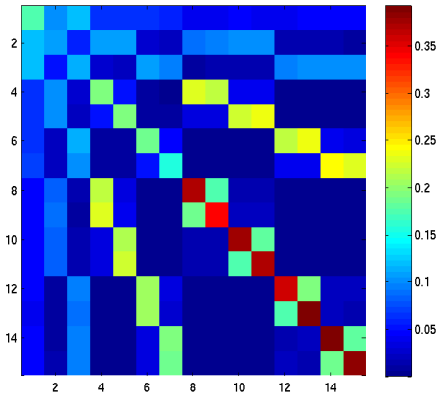
(a) $t = 1 \times 10^3$.



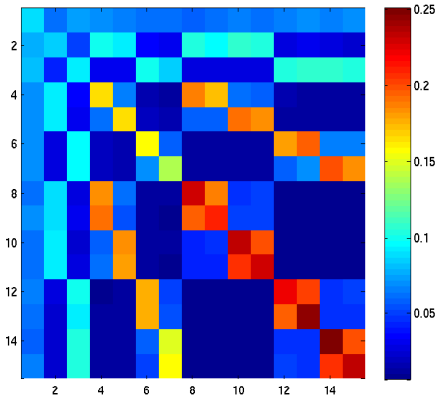
(b) $t = 1 \times 10^4$.



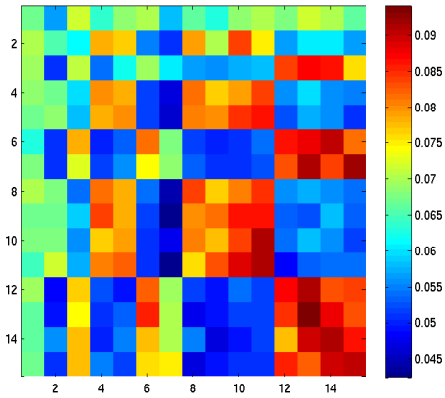
(c) $t = 2 \times 10^4$.



(d) $t = 5 \times 10^4$.



(e) $t = 1 \times 10^5$.



(f) $t = 5 \times 10^5$.

FIGURE 2.18: $p_t^{(2,N)}$ for $X_t^{(2,N)}$ in example 2 at various times t.

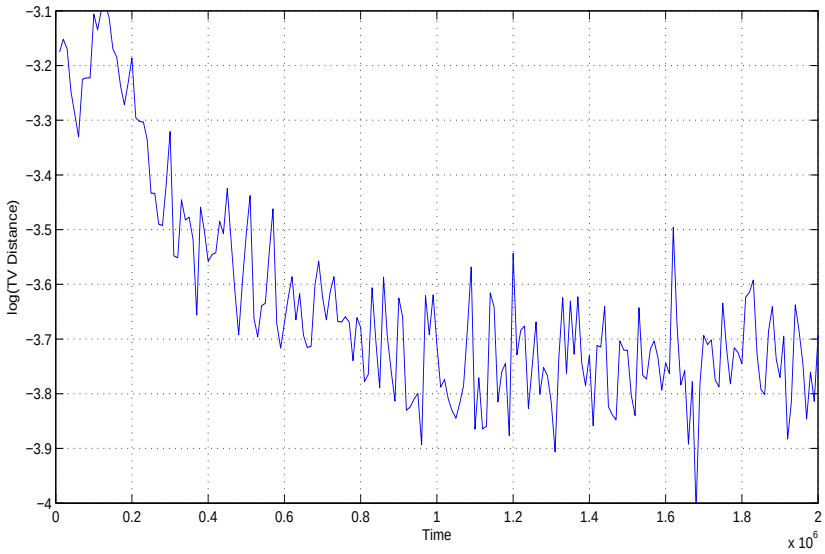


FIGURE 2.19: Approximation of the projection of the heat kernel at scale 0, $(P_t A)$, by $p_t^{(2,N)}$ in example 2 using $N = 10,000$. y-axis: $\log ||(P_t A)(x,\cdot) - p_t^{(2,N)}(x,\cdot)||_{TV}$ for a fixed x. x-axis: time in units of 10^6.

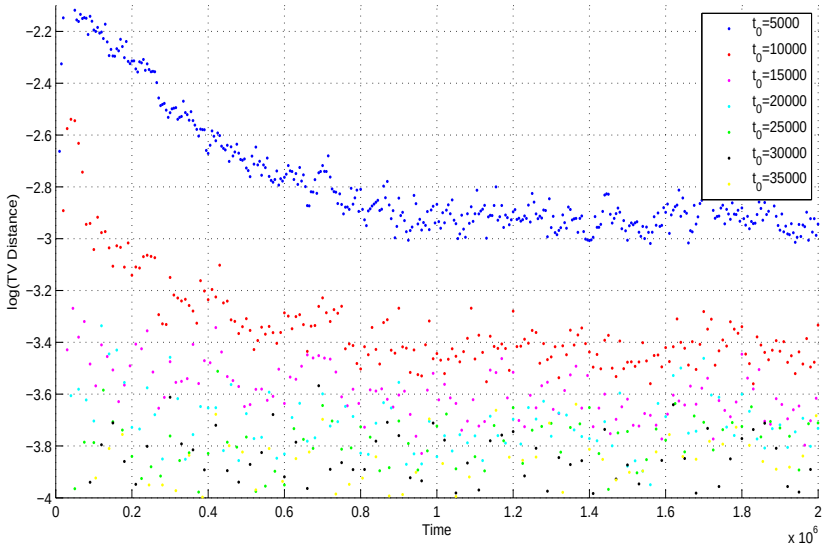


FIGURE 2.20: Plot of the non-Markov property of $p_t^{(2,N)}$ in example 2 with $N = 10,000$. It plots $\log \max_{x\in\Omega_2} \left|\left| \left(p_{t_0}^{(2,N)}\right)^t (x,\cdot) - p_{t\cdot t_0}^{(2,N)}(x,\cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using $10,000$ sample paths.

Figure 2.21 shows a comparison between the histogram of $10,000$ samples of the exit times from $\mathcal{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and the true distribution. Again, we see by inspection that these distributions are rather close.

Next, in figure 2.22 we show values of the heat kernel $p_t^{(1,\epsilon)}$ for various values of t. We note that at $t = 100$ the complete graph on 10 vertices have a lot of mass spread within them, and this continues until time 800 and 2000 when these scales have mixed. Starting at time $40,000$, the process begins to explore the tree.

In figure 2.23, we show how well $p_t^{(1,\epsilon)}$ approximates the projection of the heat kernel of the original process at scale 0 by computing

$$\log \left|\left|(P_t A)(x,\cdot) - p_t^{(1,\epsilon)}(x,\cdot)\right|\right|_{TV}$$

at a fixed vertex x as a function of t in steps of size 100 using 1,000 sample paths of $X_t^{(1,\epsilon)}$, where $P_t = e^{(P-I)t}$ is the heat kernel at scale 0 and A is the $|\Omega_0| \times |\Omega_1|$ matrix for which

$$A(i,j) = \begin{cases} 1 & \text{if } i \in \mathcal{C}_{1,j} \\ 0 & \text{otherwise.} \end{cases}$$

The transition matrix $(P_t A)(x,y)$ for $x \in \Omega_1$ represents the probability starting at a boundary vertex that the random walker, *at scale 0*, is in $\mathcal{C}_{1,y}$ at time t.

Figure 2.24 illustrates the non-Markov property of $X_t^{(1,\epsilon)}$ by computing

$$\log \max_{x \in \Omega_1} \left|\left|\left(p_{t_0}^{(1,\epsilon)}\right)^t (x,\cdot) - p_{t\cdot t_0}^{(1,\epsilon)}(x,\cdot)\right|\right|_{TV}$$

as a function of t for various choices of t_0.

At scale 2, figure 2.25 shows $p_t^{(2,N)}$ at various times for the homogenized process at scale 2.

In figure 2.26, we show how well $p_t^{(2,N)}$ approximates the projection of the heat

kernel of the original process at scale 0 by computing

$$\log \left|\left|(P_t A)(x,\cdot) - p_t^{(2,N)}(x,\cdot)\right|\right|_{TV}$$

from a fixed point x as a function of t in steps of size $10,000$, where $P_t = e^{(P-I)t}$ is the heat kernel at scale 0 and A is the $|\Omega_0| \times |\Omega_2|$ matrix for which

$$A(i,j) = \begin{cases} 1 & \text{if } i \in \mathcal{C}_{2,j} \\ 0 & \text{otherwise.} \end{cases}$$

The transition matrix $(P_t A)(x,y)$ for $x \in \Omega_1$ represents the probability starting at a boundary vertex that the random walker, *at scale 0*, is in $\mathcal{C}_{2,y}$ at time t.

Finally, figure 2.27 illustrates the non-Markov property of $X_t^{(2,N)}$ by computing

$$\log \max_{x \in \Omega_2} \left|\left|\left(p_{t_0}^{(2,N)}\right)^t (x,\cdot) - p_{t\cdot t_0}^{(2,N)}(x,\cdot)\right|\right|_{TV}$$

as a function of t for various choices of t_0.

2.4.3 Example 4: Change of Boundaries

Here, we keep the weights at each scale the same, except remove the constraint of scale j is tangent to scale $j+1$. This changes the geometry considerably at scale 0, and changes the homogenizes state space as discussed in section 2.3. For simplicity, we add multiple boundaries only to scale 1, with weights equal to 19 (that of scale 2).

We present the graph in figure 2.28.

When we homogenize the first scale, we obtain a graph whose geometry is given by figure 2.29. Note that with the addition of an edge connecting two clusters at scale 1, we add two new vertices to Ω_1.

When we homogenize the second time, we obtain a graph whose geometry is presented in figure 2.30.

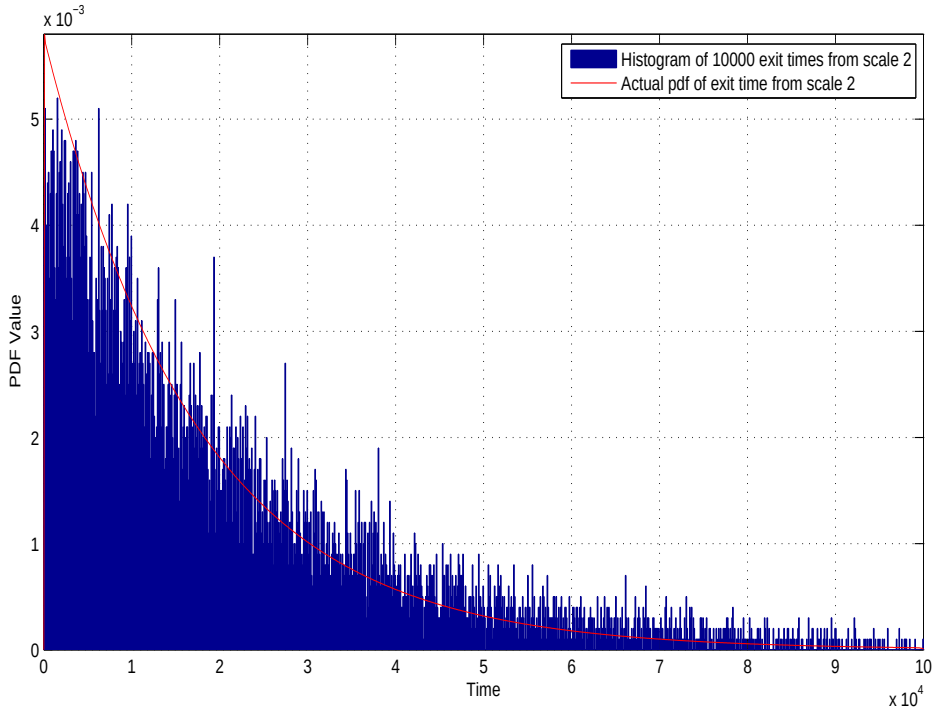
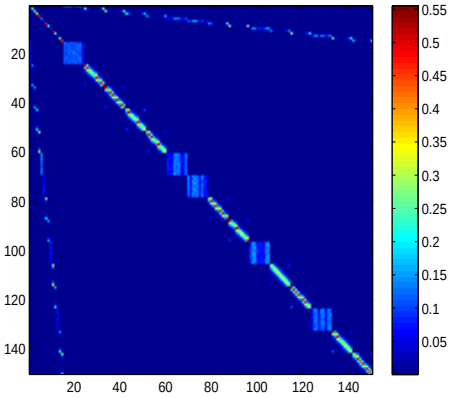


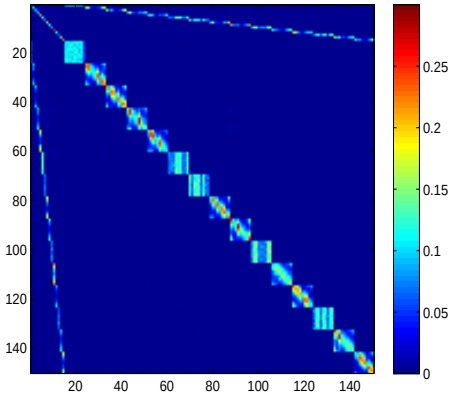
FIGURE 2.21: Exit time comparison from scale 2 in example 3. The histogram contain samples of 10,000 exit times from $\widetilde{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and $\epsilon = 10^{-6}$. The red curve is the actual probability density function from proposition 2. We've chosen bin sizes of 100 since the average amount of time leaving scale 1 (and hence transitioning between vertices at scale 1) has time approximately 100.

In this example, a cluster at scale 1 can have multiple out boundary vertices. Figure 2.31 shows a comparison between the histogram of $10,000$ samples of the exit times from $\mathcal{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and the true distribution for entry and exit through the same point. Figure 2.32 shows a comparison between the histogram of $10,000$ samples of the exit times from $\mathcal{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and the true distribution for entry and exit through different points.

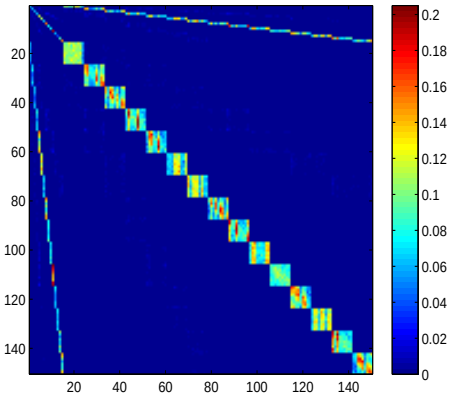
At scale 2, figure 2.33 shows $p_t^{(1,\epsilon)}$ at various times for the homogenized process at scale 1. We note that at $t = 100$, we see mass flow immediately through the new vertices added as a result of adding multiple boundaries. At larger times, we



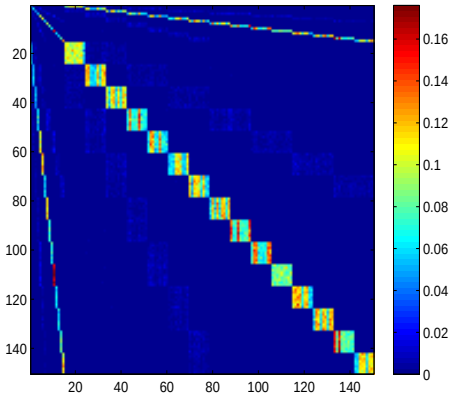
(a) $t = 10^2$.



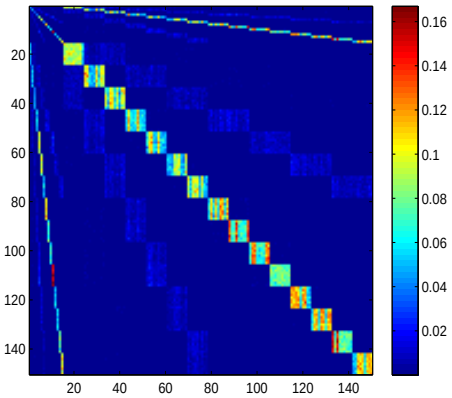
(b) $t = 4 \times 10^2$.



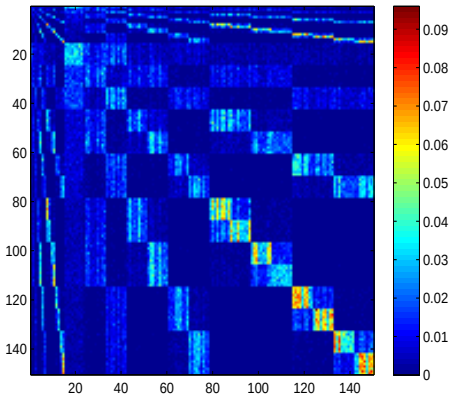
(c) $t = 8 \times 10^2$.



(d) $t = 2 \times 10^3$.



(e) $t = 4 \times 10^3$.



(f) $t = 4 \times 10^4$.

FIGURE 2.22: $p_t^{(1,\epsilon)}$ for $X_t^{(1,\epsilon)}$ in example 3 at various times t using 1,000 sample paths and $\epsilon = 10^{-6}$.

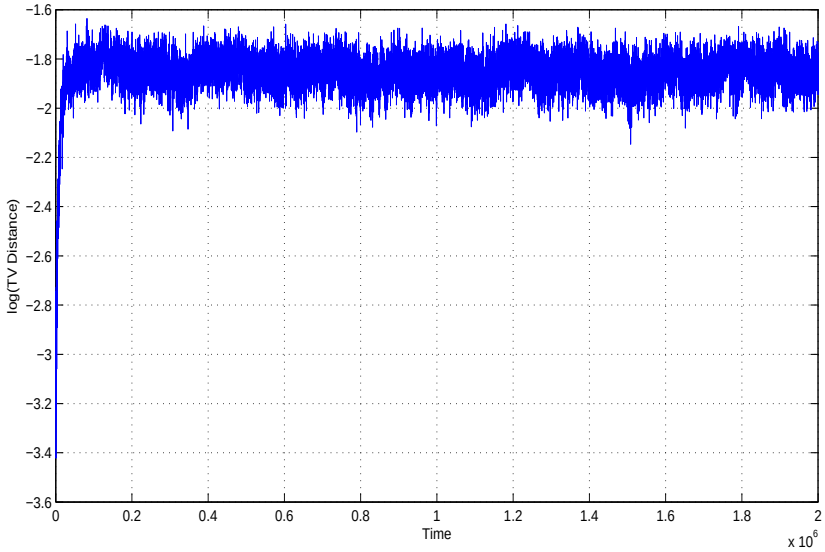


FIGURE 2.23: Approximation of the projection of the heat kernel at scale 0, $(P_t A)$, by $p_t^{(1,\epsilon)}$ in example 3 using 1,000 sample paths and $\epsilon = 10^{-6}$. y-axis: $\log ||(P_t A)(x,\cdot) - p_t^{(1,\epsilon)}(x,\cdot)||_{TV}$ from a fixed x. x-axis: time in units of 100.

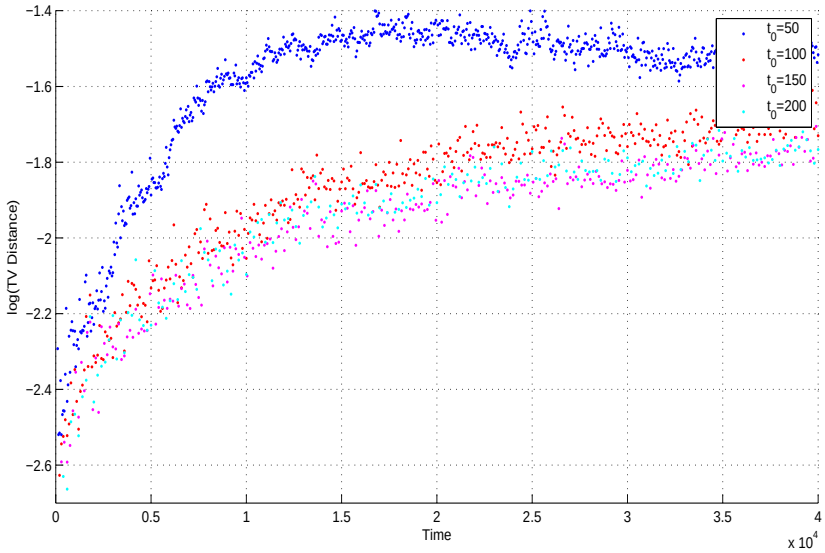
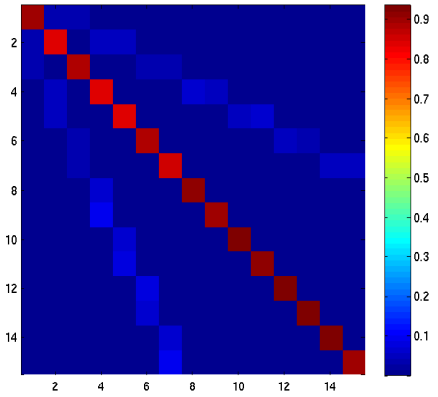
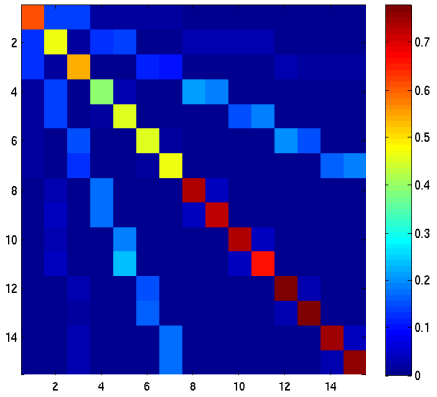


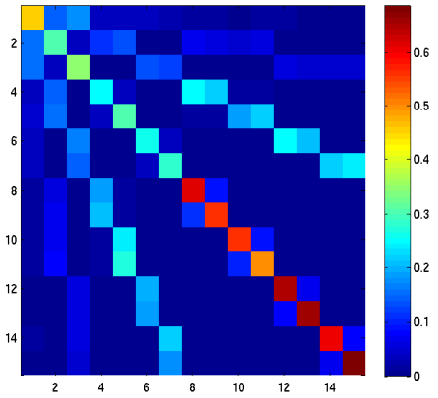
FIGURE 2.24: Plot of the non-Markov property of $p_t^{(1,\epsilon)}$ in example 3 with $\epsilon = 10^{-6}$. It plots $\log \max_{x\in\Omega_1} \left|\left| \left(p_{t_0}^{(1,\epsilon)}\right)^t (x,\cdot) - p_{t\cdot t_0}^{(1,\epsilon)}(x,\cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using 1,000 sample paths.



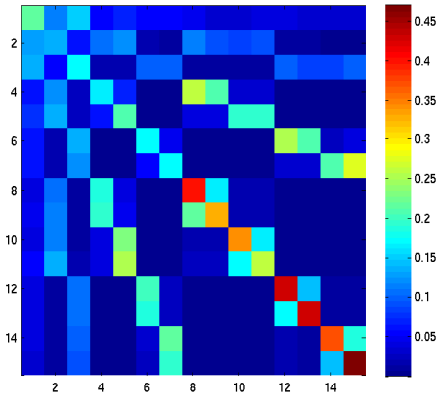
(a) $t = 1 \times 10^3$.



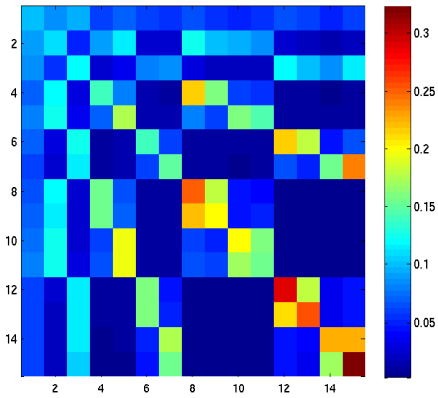
(b) $t = 1 \times 10^4$.



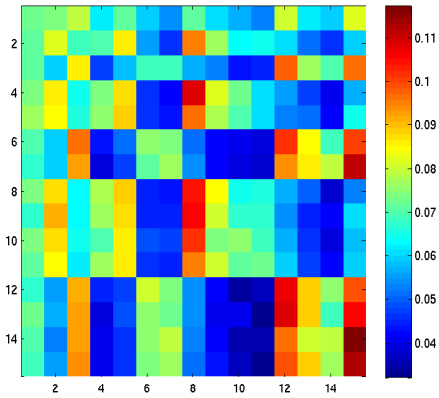
(c) $t = 2 \times 10^4$.



(d) $t = 5 \times 10^4$.



(e) $t = 1 \times 10^5$.



(f) $t = 5 \times 10^5$.

FIGURE 2.25: $p_t^{(2,N)}$ for $X_t^{(2,N)}$ in example 3 with $N = 10,000$ at various times t.

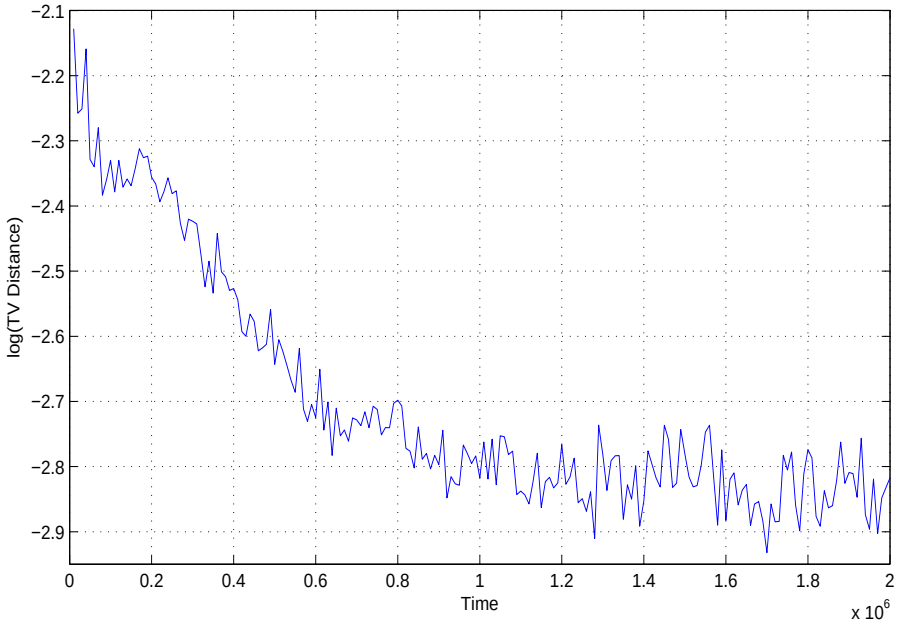


FIGURE 2.26: Approximation of the projection of the heat kernel at scale 0, $(P_t A)$, by $p_t^{(2,N)}$ in example 3 using $N = 10,000$. y-axis: $\log ||(P_t A)(x,\cdot) - p_t^{(2,N)}(x,\cdot)||_{TV}$ from a fixed x. x-axis: time in units of $10,000$.

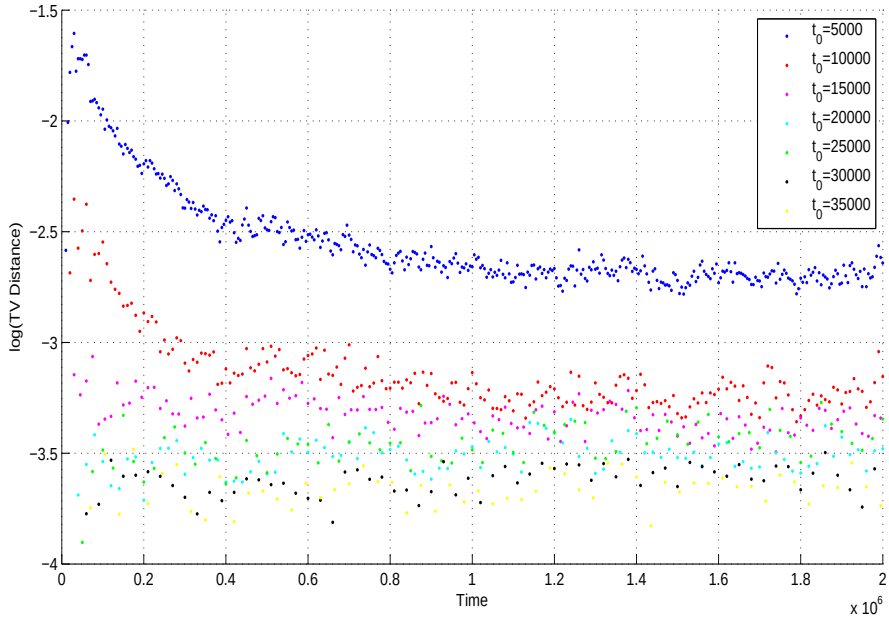
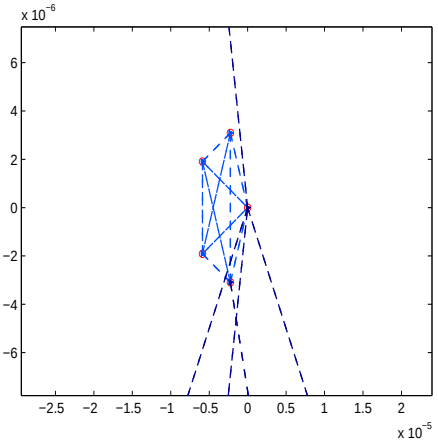
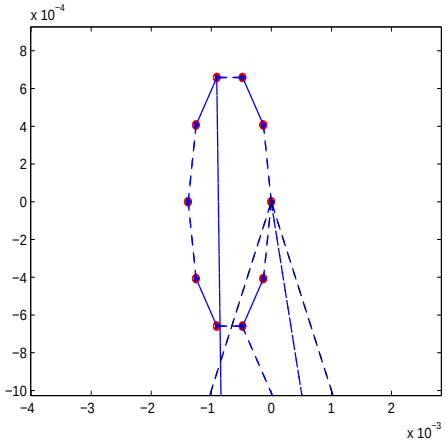


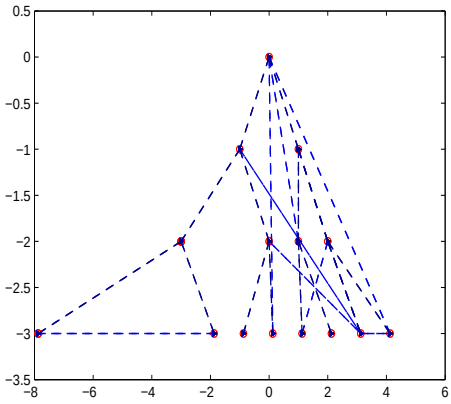
FIGURE 2.27: Plot of the non-Markov property of $p_t^{(2,N)}$ in example 3 with $N = 10,000$. It plots $\log \max_{x\in\Omega_2} \left|\left| \left(p_{t_0}^{(2,N)}\right)^t (x,\cdot) - p_{t\cdot t_0}^{(2,N)}(x,\cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using $10,000$ sample paths.



(a) Multiscale graph with multiple boundaries at scale 1.



(b) Multiscale graph with multiple boundaries at scale 2.



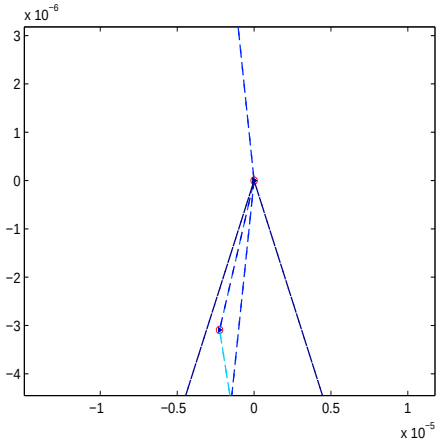
(c) Multiscale graph with multiple boundaries at scale 3.

FIGURE 2.28: The 3-scale multiscale graph with multiple boundaries. (a), (b) and (c) present visualizations of the graph at scales 1, 2 and 3, respectively

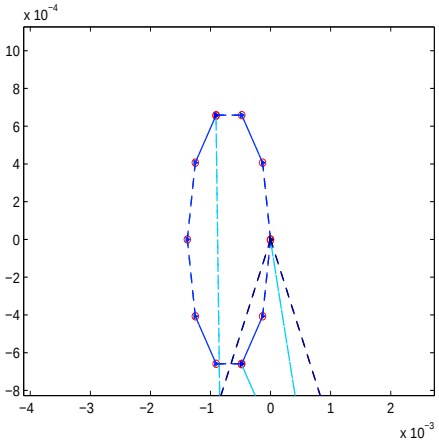
still see the mixing of each decagon, but now see "echos" of the decagons on the off block-diagonal due to mass flow through the new vertices. At $t = 2000$, we begin to see two decagons communicating with each other due to an edge connecting two of their clusters at scale 1.

Finally, figure 2.34 illustrates the non-Markov property of $X_t^{(1,\epsilon)}$ by computing

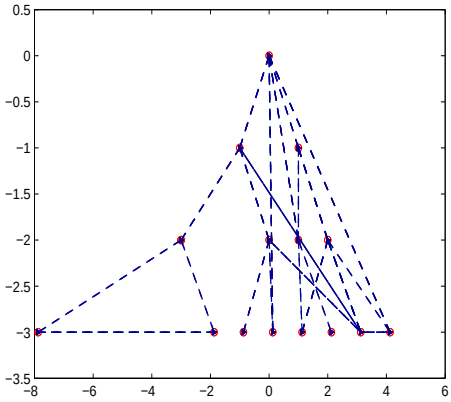
$$\log \max_{x \in \Omega_1} \left|\left| \left(p_{t_0}^{(1,\epsilon)}\right)^t (x, \cdot) - p_{t \cdot t_0}^{(1,\epsilon)}(x, \cdot) \right|\right|_{TV}$$



(a) Homogenized multiscale geometry at scale 1.



(b) Homogenized multiscale geometry at scale 1.

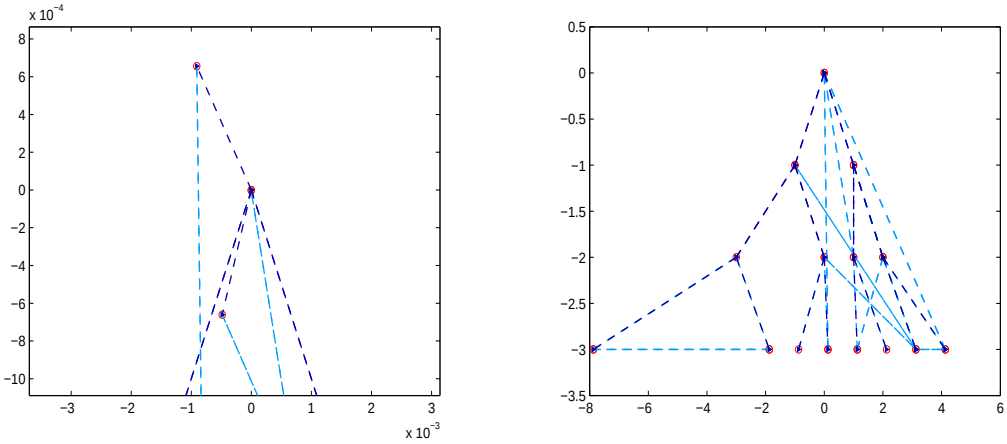


(c) Homogenized multiscale geometry at scale 3.

FIGURE 2.29: Homogenized geometry at scale 1 in example 4. (a), (b) and (c) present visualizations of the graph at scales 1, 2 and 3, respectively

as a function of t for various choices of t_0.

At scale 2, figure 2.35 shows $p_t^{(2,N)}$ at various times for the homogenized process at scale 2. The most interesting artifact here is that for large times, the treeline is transient and all the mass is absorbed into the vertices induced via multiple boundaries.



(a) Homogenized multiscale geometry at scale 2.

(b) Homogenized multiscale geometry at scale 3.

FIGURE 2.30: Homogenized geometry at scale 2 in example 4. (a) and (b) present visualizations of the graph at scales 1 and 2 respectively

Finally, figure 2.36 illustrates the non-Markov property of $X_t^{(2,N)}$ by computing

$$\log \max_{x \in \Omega_2} \left|\left| \left(p_{t_0}^{(2,N)}\right)^t (x, \cdot) - p_{t \cdot t_0}^{(2,N)}(x, \cdot) \right|\right|_{TV}$$

as a function of t for various choices of t_0.

2.4.4 Example 5: A General Example

Finally, we give a general example of a graph with different geometries *and* multiple boundaries.

Figures 2.37 and 2.38 give the comparison of exit times from scale 2 using $X_t^{(1,\epsilon)}$ with the the true density.

Figures 2.39 and 2.41 plot $p_t^{(1,\epsilon)}$ and $p_t^{(2,N)}$ for various choices of t, respectively, and figures 2.40 and 2.42 the non-Markov property for these homogenizations.

2.5 Conclusions and Future Work

When an undirected network admits a multiscale partition $\{\mathcal{C}_{j,k}\}_{k \in \mathcal{K}_j}$ whose members at scale j represent regions in which the time scale a Markov process takes to

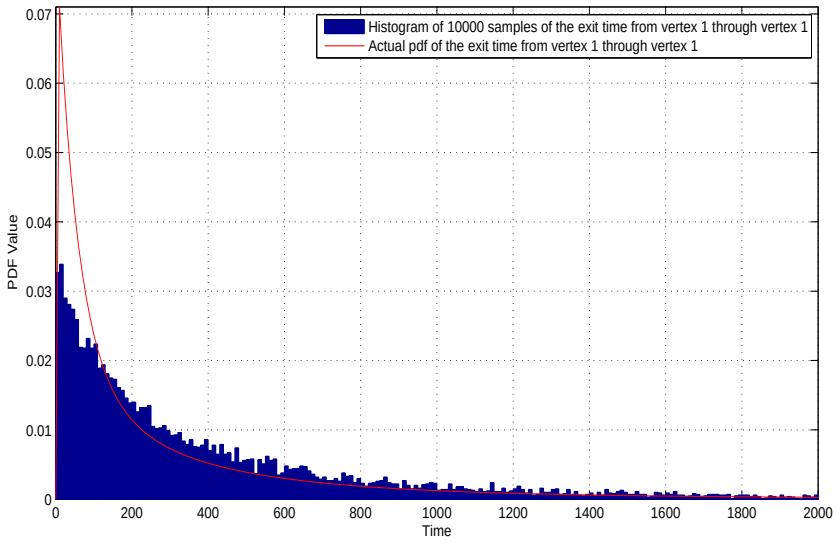


FIGURE 2.31: Exit time comparison starting and leaving through the same vertex at scale 2 in example 4. The histogram contain samples of 10,000 exit times from $\widetilde{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and $\epsilon = 10^{-6}$. The red curve is the actual probability density function from proposition 2. We've chosen bin sizes of 10 for visualization.

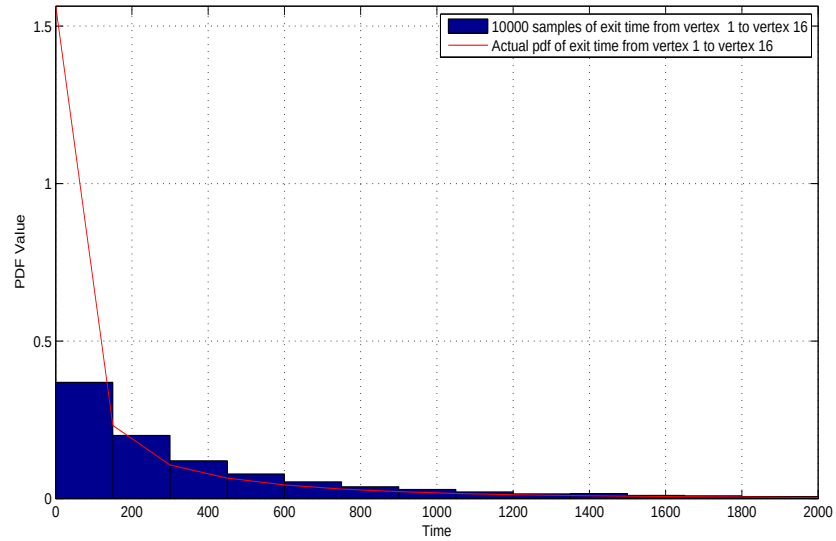
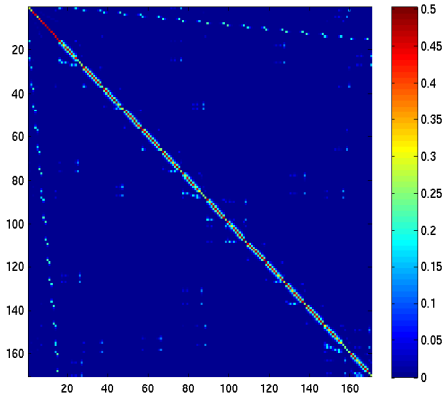
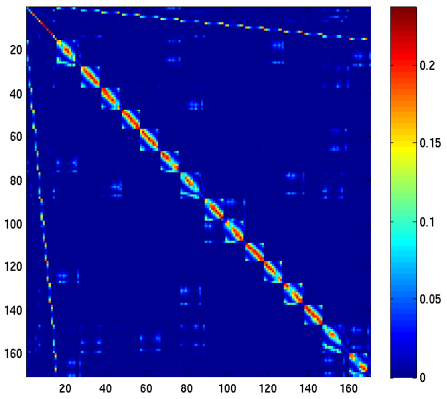


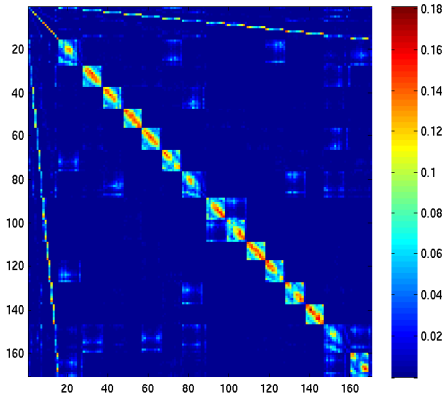
FIGURE 2.32: Exit time comparison starting and leaving through different vertices at scale 2 in example 4. The histogram contain samples of 10,000 exit times from $\widetilde{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and $\epsilon = 10^{-6}$. The red curve is the actual probability density function from proposition 2. We've chosen bin sizes of 150 for clarity.



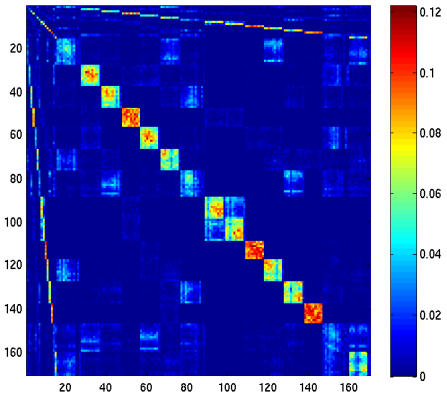
(a) $t = 10^2$.



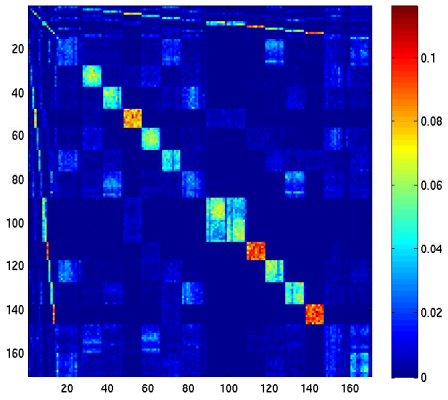
(b) $t = 4 \times 10^2$.



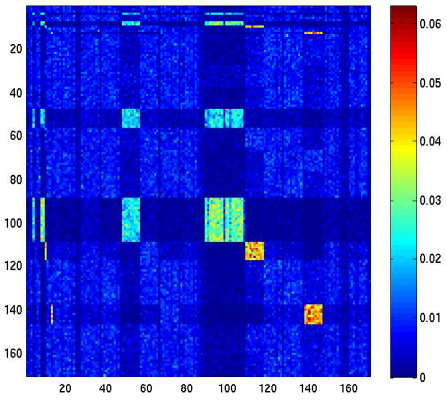
(c) $t = 8 \times 10^2$.



(d) $t = 2 \times 10^3$.



(e) $t = 4 \times 10^3$.



(f) $t = 4 \times 10^4$.

FIGURE 2.33: $p_t^{(1,\epsilon)}$ for $X_t^{(1,\epsilon)}$ in example 4 at various times t using 1,000 sample paths and $\epsilon = 10^{-6}$.

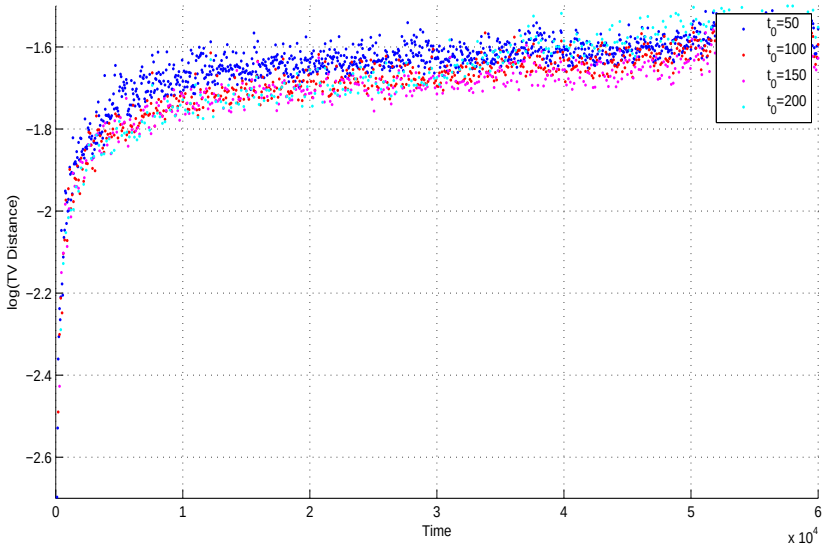
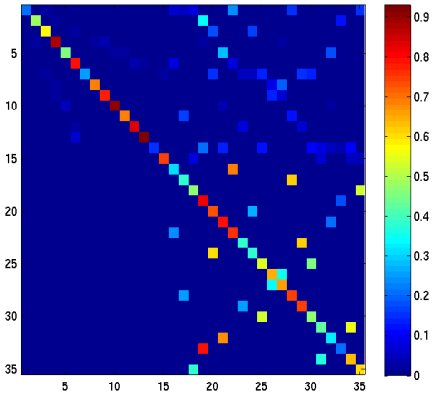


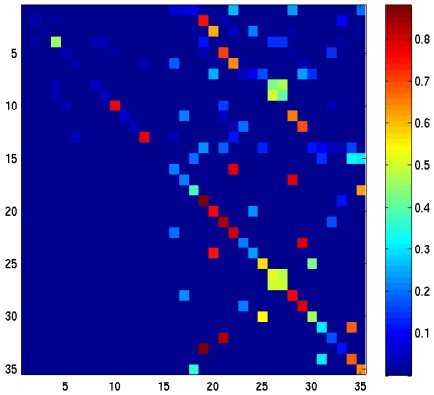
FIGURE 2.34: Plot of the non-Markov property of $p_t^{(1,\epsilon)}$ in example 4 with $\epsilon = 10^{-6}$. It plots $\log \max_{x\in\Omega_1} \left|\left|\left(p_{t_0}^{(1,\epsilon)}\right)^t (x,\cdot) - p_{t\cdot t_0}^{(1,\epsilon)}(x,\cdot)\right|\right|_{TV}$ as a function of t for various choices of t_0 using 1,000 sample paths.

explore them are much smaller than the time scale of exploring the entire network, we've delivered a stochastic (non-Markov) process representing the original process' transitions at these scales. By using kernel density estimation for the corresponding hold times, we also prove error bounds in total variation distance of such approximate processes to their real ones, and deliver concentration type inequalities determining how many sample paths are necessary to simulate to approximate the law of the process within a given precision, and evaluate the performance of our algorithms on several synthetic data sets.

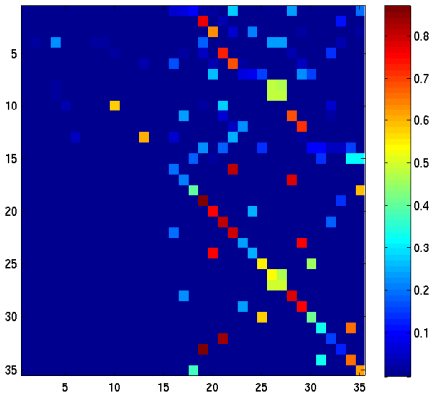
In real world data, however, we will have no a priori knowledge of this partition, and it will have to be learned in an adapted way. One way to do this, at least for undirected networks, is to use local spectral algorithms [Spielman and Teng (2004), Anderson and Chung (2006), Mahoney (2010)] to partition a network, and iterate the procedure on each partition and their members successively. This is indeed the



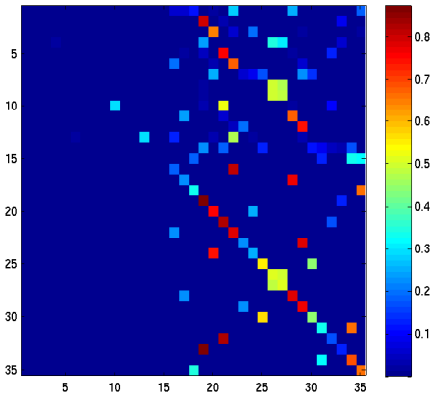
(a) $t = 1 \times 10^3$.



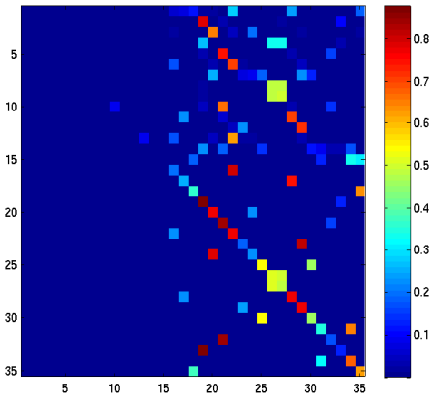
(b) $t = 1 \times 10^4$.



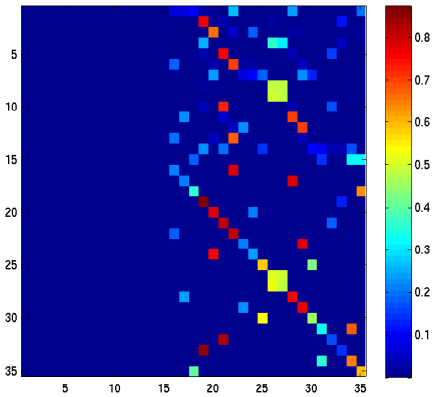
(c) $t = 2 \times 10^4$.



(d) $t = 5 \times 10^4$.



(e) $t = 1 \times 10^5$.



(f) $t = 5 \times 10^5$.

FIGURE 2.35: $p_t^{(2,N)}$ for $X_t^{(2,N)}$ in example 4 with $N = 10,000$ at various times t.

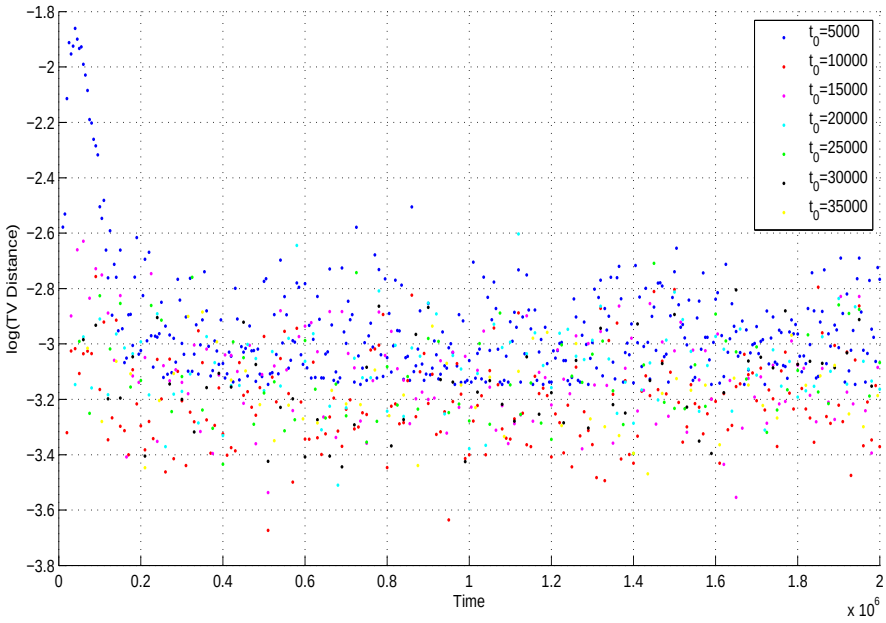


FIGURE 2.36: Plot of the non-Markov property of $p_t^{(2,N)}$ in example 4 with $N = 10,000$. It plots $\log \max_{x \in \Omega_2} \left|\left| \left(p_{t_0}^{(2,N)}\right)^t (x,\cdot) - p_{t \cdot t_0}^{(2,N)}(x,\cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using $10,000$ sample paths.

algorithm we have used in our undirected, synthetic, examples.

One setback from this is the stability of local partitions as a function of the seed nodes and input conductance. If any scales in the network truly exist, then we should be able to detect them regardless of the choice of seed nodes. In our toy graphs, this is definitely case since we've enforced a rather strong separation of scale through the choice of weights, however in real data, we'll have no idea how to choose the maximum conductance in these algorithms. Robustness of local spectral methods to the input seed nodes and conductance indeed is an open problem.

The extension of these methods to continuous time Markov processes without $exp(1)$ hold times is straightforward, and the authors have done so. What remains, at least in the undirected case, is to use this multiscale decomposition to perform *graph*

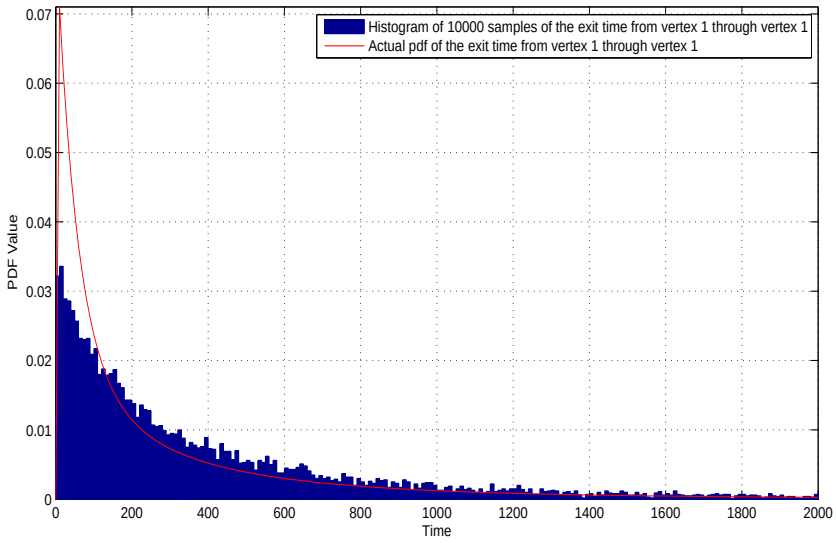


FIGURE 2.37: Exit time comparison starting and leaving through the same vertex at scale 2 in example 5. The histogram contain samples of 10,000 exit times from $\widetilde{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and $\epsilon = 10^{-6}$. The red curve is the actual probability density function from proposition 2. We've chosen bin sizes of 10 for visualization.

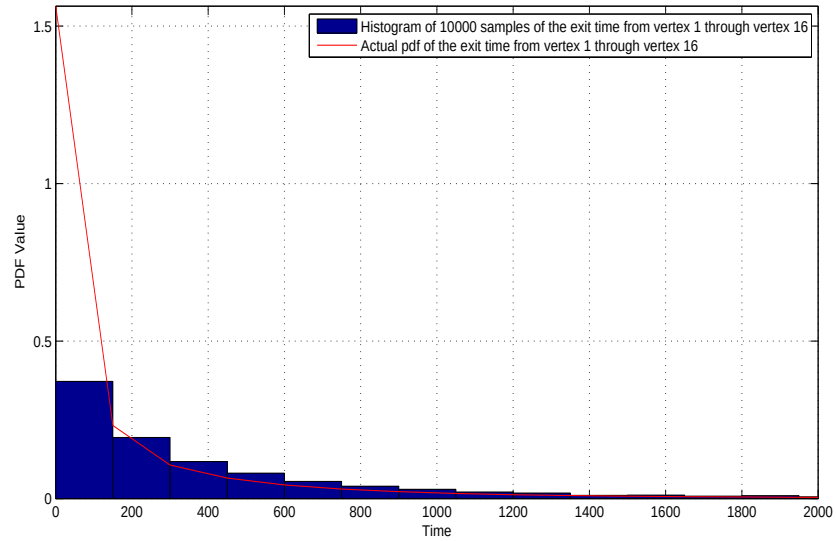
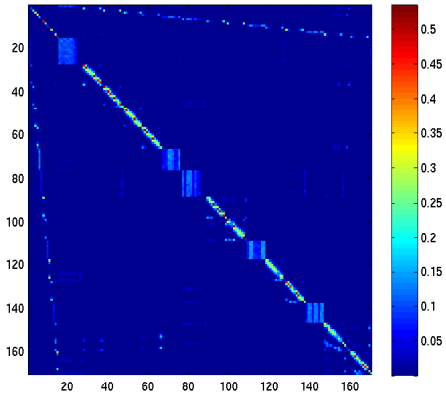
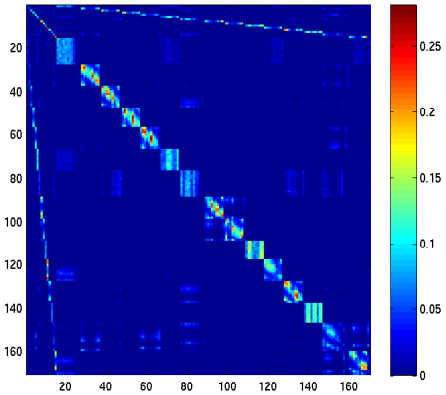


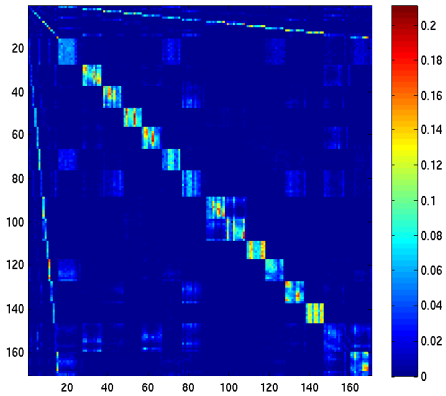
FIGURE 2.38: Exit time comparison starting and leaving through different vertices at scale 2 in example 5. The histogram contain samples of 10,000 exit times from $\widetilde{C}_{2,k}$ using $X_t^{(1,\epsilon)}$ and $\epsilon = 10^{-6}$. The red curve is the actual probability density function from proposition 2. We've chosen bin sizes of 150 for visualization.



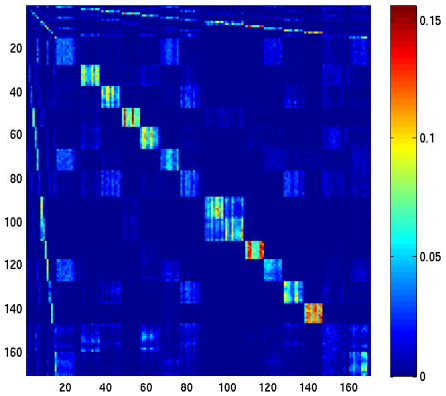
(a) $t = 10^2$.



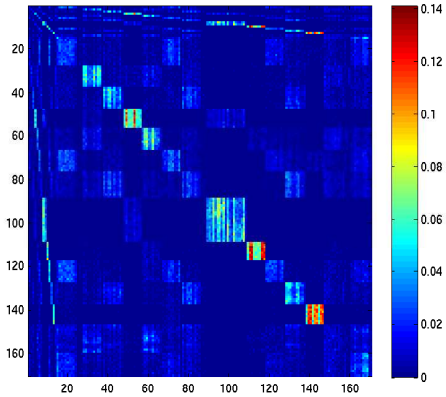
(b) $t = 4 \times 10^2$.



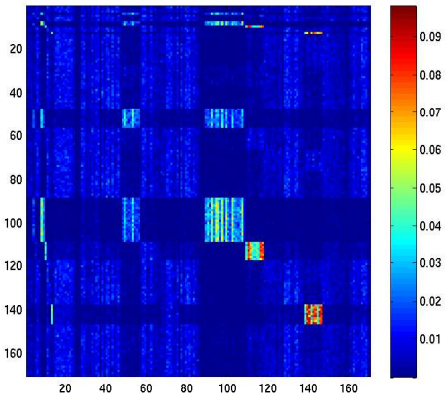
(c) $t = 8 \times 10^2$.



(d) $t = 2 \times 10^3$.



(e) $t = 4 \times 10^3$.



(f) $t = 4 \times 10^4$.

FIGURE 2.39: $p_t^{(1,\epsilon)}$ for $X_t^{(1,\epsilon)}$ in example 5 at various times t using 1,000 sample paths and $\epsilon = 10^{-6}$.

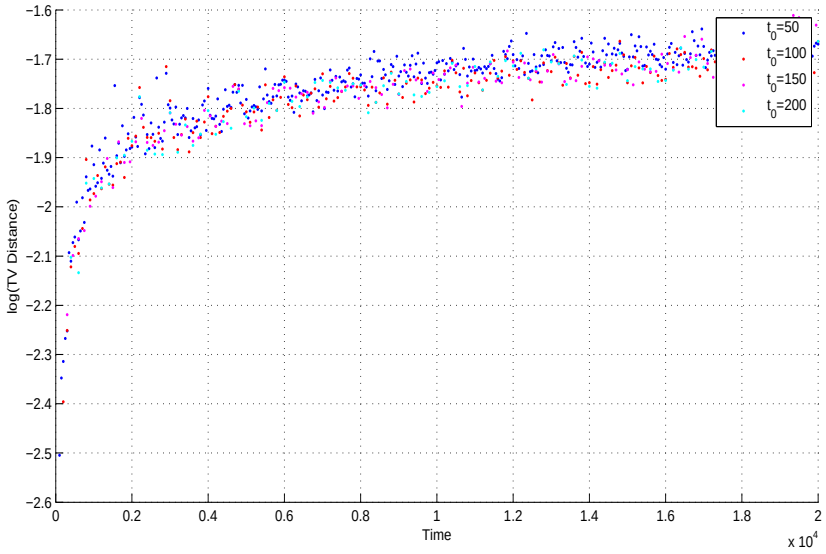
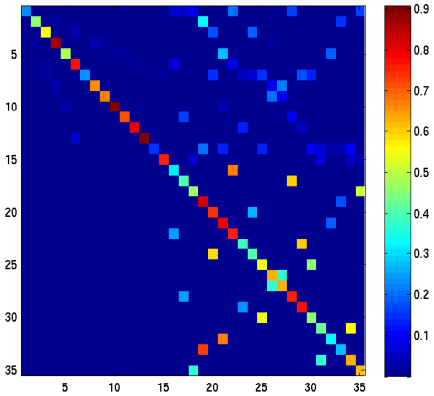


FIGURE 2.40: Plot of the non-Markov property of $p_t^{(1,\epsilon)}$ in example 5 with $\epsilon = 10^{-6}$. It plots $\log \max_{x\in\Omega_1} \left|\left| \left(p_{t_0}^{(1,\epsilon)}\right)^t (x,\cdot) - p_{t\cdot t_0}^{(1,\epsilon)}(x,\cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using 1,000 sample paths.

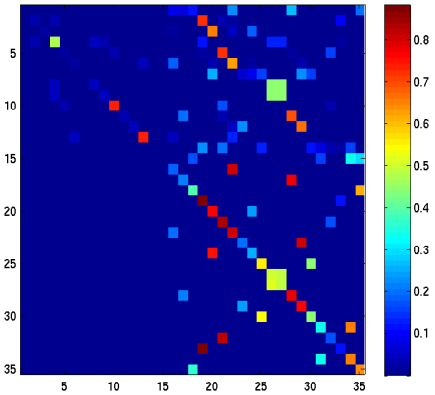
comparison. We've seen great success at this theory in ascertaining the geometry of a network at different scales, and this is the ultimate goal the authors have in mind.

It might be noticed that these methods immediately carry over to the regime of directed networks. Indeed, much of what we have done does extend straightforward, and interestingly since the in and out boundary vertices could be very different. We have held off on these applications, however, because developing a multiscale partition that clusters directed networks remains an (important) open problem.

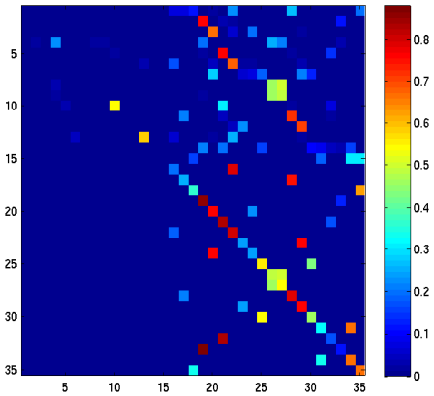
Anderson and Chung (2007) has a made a first attempt is made in this direction by generalizing conductance to a directed network and developing a local spectral algorithm akin to those for undirected networks. One major setback of this theory, however, is the requirement of strong connectness to ensure that the stationary distribution is supported everywhere. That is, if the network has transient states, then the algorithm fails. Indeed, during the second sweep of partitioning a directed



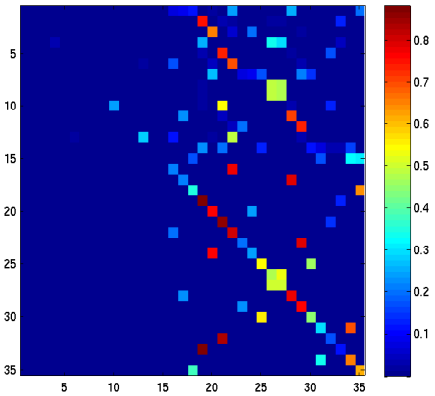
(a) $t = 1 \times 10^3$.



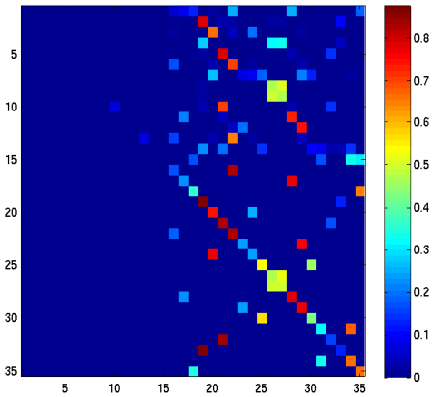
(b) $t = 1 \times 10^4$.



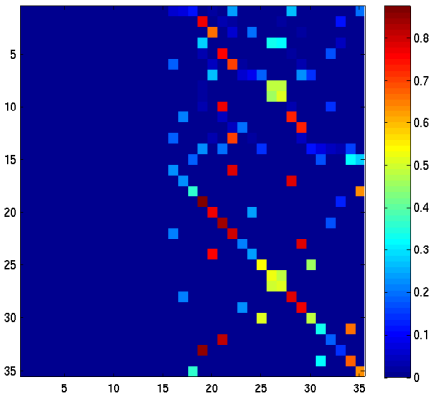
(c) $t = 2 \times 10^4$.



(d) $t = 5 \times 10^4$.



(e) $t = 1 \times 10^5$.



(f) $t = 5 \times 10^5$.

FIGURE 2.41: $p_t^{(2,N)}$ for $X_t^{(2,N)}$ in example 5 with $N = 10,000$ at various times t.

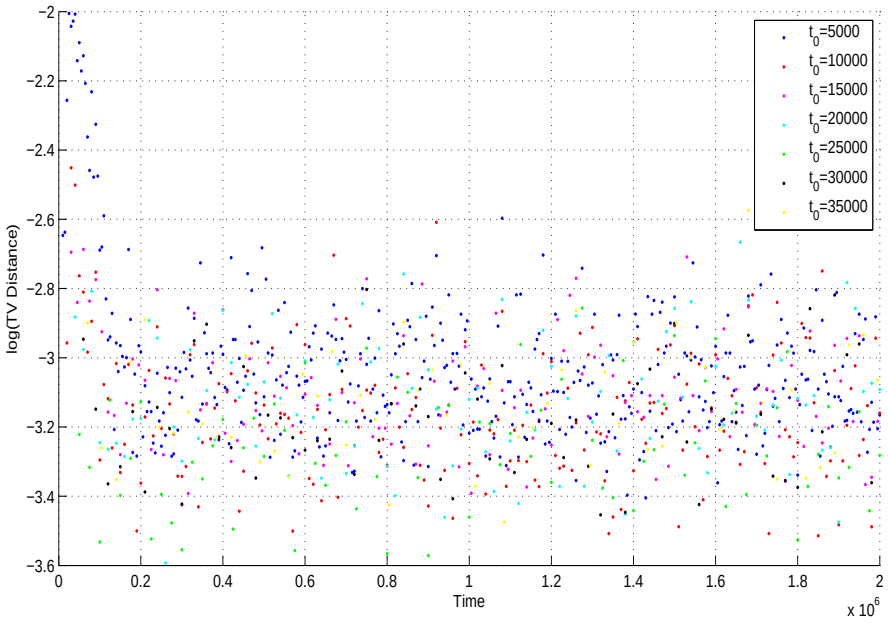


FIGURE 2.42: Plot of the non-Markov property of $p_t^{(2,N)}$ in example 5 with $N = 10,000$. It plots $\log \max_{x \in \Omega_2} \left|\left| \left(p_{t_0}^{(2,N)}\right)^t (x,\cdot) - p_{t \cdot t_0}^{(2,N)}(x,\cdot) \right|\right|_{TV}$ as a function of t for various choices of t_0 using $10,000$ sample paths.

network via this algorithm, one could have removed a set in the previous step that introduced transient states. The authors circumvent this situation by adding a backgound adjacency matrix that breaks transience in order to repeat the algorithm.

While certainly a first step, one way around this is to use a *dynamic* measure of clustering that does not reference the stationary distribution, and hence, does not require strong connectedness. This is certainly an avenue of future research the authors will pursue.

Finally is the issue of kernel density estimation. The nonparametric density estimation techniques we have used in our algorithm use a Gaussian kernel. While this theory is certainly well understood and useful in many circumstances, it does not seem appropriate for density estimation of nonnegatively supported distributions

with potential discontinuities such as exit times of Markov processes we've seen in our examples. Development of nonparametric density estimation techniques that can deal with such examples is certainly an interesting open question, and leaves developing nonasymptotic rates of convergence of the estimator to the true density in total variation an open and challenging problem.

3

Higher Order Semi-Supervised Learning

3.1 Introduction

In semi-supervised learning, one develops techniques that make use of both labeled and unlabeled data for training. Typically, there is a small amount of labeled data and a large amount of unlabeled data. A classical problem is the so-called classificaton problem (also referred to as transductive inference) was described in the introduction: let $G = (V, E, W)$ be a weighted graph, where vertices are the data elements, and edges have weights along them defining some sort of symmetric similarity or affinity between data points. Let $S \subset V$ be a subset of V, and $y : S \to \{-1, 1\}$ be a labeling of the points in S. Given this local labeling of points in S, can we infer a global labeling of points in V? How well does it perform in the case where we know the global labels?

This problem has received considerable attention in recent years. One approach, diffusive transductive inference Szlam (2008), involves constructing a reversible Markov process P on the graph G in which probability transitions between vertices are proportional to the weight (similarity) connecting them. One then runs the heat kernel

t steps into the future on step functions corresponding to the initial labeling, and by labeling a point by that label with the most amount of heat on it, one obtains a global classifier. Using this process to cross validate the initial data using leave-K-out cross validation to prevent overfitting, one obtains a value of t in which the above procedure is repeated on all initial class labels, obtaining a global classifier.

Let's consider applying this technology to a focused gene expression microarray of tumors from 255 men with prostate cancer using RNA from archival FFPE tissue Nakagawa (2008). Here, 502 paraffin-embedded blocks from eligible men were identified and each block was surveyed for the tissue present, so the data is given by a 502×255 real-valued matrix, $D(i, j)$. Each patient is labeled as either systemic, PSA, or NED, representing the state of cancer in each patient, and we take the knowledge of 24 randomly chosen individuals as our training set, keeping at least three members from each class.

We can treat each gene as a continuous or discrete random variable. In the continuous case, in order to compare the expression of different genes, we reformat the data so that $D(i, j)$ is now the z-score [appropriately rescaled expression] for the i^{th} gene. Then, we can interpret each patient as a continuous random variable: given a patient, we randomly select a gene, and report its z-score. Pairwise similarity here is given by the one minus the (absolute value) correlation coefficient between two patients.

In the discrete case, we can treat the genes as either on or off, taking the values 1 and 0, respectively. We compute the mean and standard deviation of each row (paraffin-embedded block), and reformatted the rows to take values in $\{0, 1\}$: 1 if the block produced within 1 standard deviation of its mean, and 0 otherwise. We then obtain the interpretation of each patient as a Bernoulli random variable: we randomly choose a gene, and measure whether or not it is activated (0 or 1). Similarity here is given by one minus the normalized mutual information between

two patients.

Since the labels concern the state of cancer in each patient, either choice of similarity is reasonable, since the state of cancer in two patients should be more similar the more correlated randomly selected genes from each individual are.

In the experiments that follow, for any approach, we compute the exponential of the pairwise similarity, and keep the 20 nearest neighbors similarity, and use leave-23-out cross validation to determine the value of t. We use 12 splits of 24 different patients to obtain cross-validation errors and test errors, and report the average of these errors.

Using the continuous approach, we obtain a cross-validation error of 61.81% with a test error of 67.06%. In the discrete approach, we obtain a cross-validation error of 55.9% and a test error of 66.49%. Errors are given in table 3.2 under the P-column (for any similarity, we call this chain the P chain).

Now, some of these patients (remember, they're random variables) might be pairwise independent, so that they have zero pairwise similarity. However, there might be a higher order dependence among three of them that doesn't show up in pairwise dependence. Can we take into account this higher order dependency and possibly improve the predication error and its cross-validation error? Our approach attempts this.

The idea is as follows: interpret the existence of each such three-wise similarity as a face on its incident vertices defined appropriately to capture the three-wise dependence. Then, we turn the set of undirected edges on the graph itself into a graph, with the undirected edges as the vertices, and faces as the edges. We construct a reversible chain on this new graph in the usual way, and pull back the random walk to a random walk on the original vertex set in two different ways: one by using its stationary distribution as weights on edges (the π chain), and another by transitioning on the vertices through single iteration of the edge-walk (the $E1$-chain). With

P_1 as the P-chain and P_2 as one of the $\pi, E1$-chains, we perform a Markov mixture, $P_c = cP_1 + (1-c)P_2$, and proceed as usual with diffusive transductive inference described above to diffuse class labels, using leave-K-out cross validation to estimate the two parameters c and t, the idea being that by interpolating between clusters reflecting pairwise correlations (P_1) on the one hand, and threewise correlations on the other (P_2), we can diffuse information more accurately and obtain better classifiers.

We use the same 12 splits and report the average cross-validation error and test error over these splits. Errors and test errors reported in table 3.1. We also include percentage cross-validation and test errors in using just the π and $E1$ chains in table 3.2.

Table 3.1: Summary of mixing chain percentage errors in the prostate cancer data set for different chains.

	P Chain		π Chain		$E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error	CV Error	Test Error
Correlations	61.81%	67.06%	55.90%	65.33%	57.64%	64.03%
Mutual Information	55.90%	66.49%	57.64%	66.38%	60.07%	66.52%

Table 3.2: Summary of percentage errors in the prostate cancer data set using just pairwise similarities.

	$P-\pi$ Chain		$P-E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error
Correlations	52.43%	65.15%	52.43%	64.97%
Mutual Information	53.12%	67.06%	54.17%	64.79%

A few observations on our results. First, in using correlations, the π and $E1$ chain always gives a better test and CV error than using just the P chain. Using mutual information, the π chain gives a better test error than using P, but using $E1$ gives us a worse test error, but not by much in both cases. The fact that π and $E1$ perform better than P in some cases is in itself is surprising, at least to the authors. Second, the mixed chains consistently give better CV errors, and better if not much worse test errors except in the case of the $P-\pi$ mixing, where it is worse than

both the P and π test errors. Regardless of the similarity one chooses, our method always delivers a better test error: using mutual information, we get the best test error of 64.79% with $P - E1$, and the best test error of 64.03% with $E1$ when using correlations.

We also include the cross-validation landscape of errors in (c, t) in the correlation case. Figure 3.1 gives the landscape using the $P - E1$ mixing and figure 3.2 gives the error landscape in using the $P - \pi$ mixing. Columns denote time steps and range from 0 to 10, while rows denote mixing probabilities c incrementing in steps of 0.1 from top to bottom starting at 0. Notice how the fluctuations of the cross-validation error in a neighborhood of its minimum value is approximately the same in all cases compared to the case of using just P (the last row in each of the figures).

These results suggest by formalizing the approach and applying it on other data sets, we might achieve substantially better results. Indeed, this is the case, and the purpose of this paper is to flesh out the details of this approach in regards to data sets that can be interpreted as random variables, and present the results of experiments performed on other data sets. The structure is as follows. Section 3.2.1 briefly reviews the theory of Markov chains relevant for our construction. In section 3.2.2, we present the construction of the pairwise walk, and in sections 3.2.3 and 3.2.4 construct the edge walk and its pullback to the vertex set, respectively. Section 3.2.5 assembles the pullback with the pairwise one and describes how we perform our diffusive inference.

Section 3.3 reviews the notions of Shannon entropy and correlations relevant in the applications of our theory, and section 3.4 discusses concentration phenomena when using either of these similarities. Finally, section 3.5 contains applications to a variety of data sets.

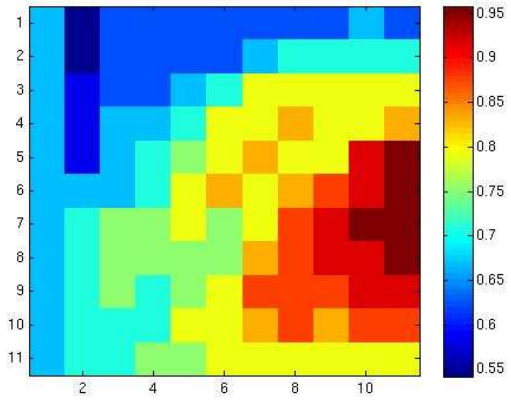


FIGURE 3.1: Cross-validation errors as a function of (c, t) in the prostate cancer data set using correlations and $P_c = cP_P + (1-c)P_{E1}$. Columns denote time starting from 0 in steps of 1 from left to right. Rows denote mixing probabilities c in increments of 0.1 from top to bottom.

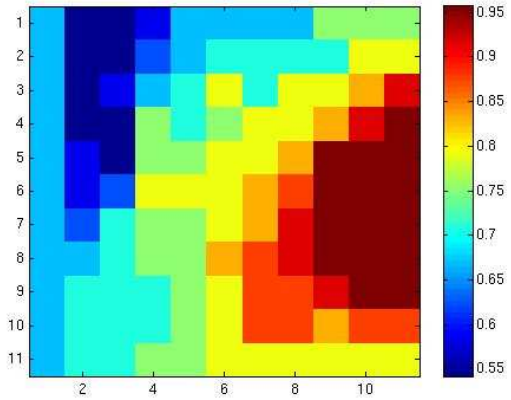


FIGURE 3.2: Cross-validation errors as a function of (c, t) in the prostate cancer data set using correlations and $P_c = cP_P + (1-c)P_{\pi}$. Columns denote time starting from 0 in steps of 1 from left to right. Rows denote mixing probabilities c in increments of 0.1 from top to bottom.

3.2 Diffusions on Graphs

There are two ways in mapping data onto graphs: as a point cloud in $\mathbb{R}^n$ whose (undirected) edges represent pairwise similarities, or a system of random variables with (undirected) edges denoting dependencies. In practice, both could be true for a given data set. Indeed, with the prostate cancer data set in section 1 we could interpret each patient as a point in $\mathbb{R}^{502}$ using the raw data, or as a continuous (resp. discrete) random variable after possibly reformatting the data. For all data sets presented in this paper, the perspective we take is the latter.

So we've mapped our data set onto a graph. Now what? In general, we can obtain an intrinsic analysis of a graph by the use of a random walk or diffusion. The purpose of sections 3.2.1 and 3.2.2 are to review these constructions. In sections 3.2.3–3.2.5 we describe how we obtain higher order corrections to these diffusion processes.

3.2.1 Markov Chains

We first recall some concepts and terminology from the theory of Markov Chains.

A *Markov Chain* on the finite state space Ω is a random walk $\{X_n\}_{n=0}^{\infty}$ on Ω satisfying the "memory forgetting" property, or *Markov property*,

$$\begin{aligned}\mathbb{P}\left[X_{n+1} = y | X_n = x, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1\right] \\ = \mathbb{P}\left[X_{n+1} = y | X_n = x\right] = P_n(x, y) \quad \forall n \in \mathbb{N}, \quad x, y, x_j \in \Omega. \quad (3.1)\end{aligned}$$

The distributions $P_n(x, \cdot)$ are called the *transition probabilities* at time n. When $P_n = P$ for all n, the random walk is *time-homogeneous*, and throughout this paper, all Markov chains are assumed to be time-homogeneous unless otherwise specified.

Given an initial distribution $\pi(x) = \mathbb{P}\left[X_0 = x\right]$, interpreted to be a non-negative row vector indexed by Ω summing to one, we have that for a time-homogeneous chain:

$$\mathbb{P}\left[X_n = x\right] = (\pi P^n)(x). \quad (3.2)$$

The *stationary distribution* of P is an initial distribution π such that $\pi P = \pi$; that is, it's a non-negative left eigenvector of P with eigenvalue 1. Its interpretation is that if we start in this distribution, we stay there:

$$(\pi P^n)(x) = \pi(x) = (\pi P^m)(x) \Leftrightarrow \mathbb{P}[X_n = x] = \mathbb{P}[X_m = x] \quad \forall n, m \in \mathbb{N}. \tag{3.3}$$

Conditions for the existence of a unique stationary distribution are that P must be:

1. *Irreducible*: Given $x, y \in \Omega$ there exists a path of length t connecting them. (read: $\forall x, y \in \Omega$, there exists $n \in \mathbb{N}$ such that $P^n(x, y) > 0$).

2. *Aperiodic*: The random walk cannot oscillate between distributions supported on different subsets of Ω (read: $gcd\{n : P^n(x, x) > 0\} = 1$).

By the *ergodic theorem*, an irreducible, aperiodic chain has $\mu P^t \to \pi$ as $t \to \infty$ for any choice of initial distribution μ.

A Markov Chain P is *reversible* with respect to the stationary distribution π if

$$\pi(x)P(x, y) = \pi(y)P(y, x). \tag{3.4}$$

Intuitively, this property says that if we start in π, the dynamics of running the chain forwards for a length of time is the same as running the chain backwards for the same length of time.

Next, if P_1 and P_2 are time homogeneous Markov transition kernels, then for any $c \in [0, 1]$, so is $P_c = cP_1 + (1 - c)P_2$, as can easily be verified since the entries are obviously nonnegative and the rows sum to 1. We call P the *mixture of the chains* P_1 *and* P_2 with mixing probability c.

Finally, we note that if f is a column vector indexed by Ω, interpreted to be a function defined on Ω, then

$$(Pf)(x) = \sum_{y \in \Omega} P(x, y)f(y) = \sum_{y \in \Omega} \mathbb{P}[X_1 = y | X_0 = x] f(y) = \mathbb{E}_x[f(X_1)]. \tag{3.5}$$

3.2.2 The Vertex Walk

Consider a finite weighted graph $G = (V, E, W)$ consisting of vertices V, edges E, and a weight function $W : E \to \mathbb{R}^+$. The *weight* along an edge connecting x and y is given by $W(x, y)$, and we write $x \sim y$ if and only if $W(x, y) > 0$. The weight $W(x, y)$ is a measure of pairwise similarity between the vertices x and y. If W is symmetric, i.e. when pairwise similarities are symmetric, we say that the graph is *undirected*. It is *directed* otherwise.

Question: How shall we construct a Markov Chain on the state space V that respect pairwise relations $W(x, y)$?

The idea is to construct transition probabilities $P(x, y)$ that are directly proportional to $W(x, y)$. Define the *degree at* x to be:

$$w(x) = \sum_{y \in V} W(x, y), \tag{3.6}$$

and

$$P(x, y) = \frac{W(x, y)}{w(x)}. \tag{3.7}$$

It's easy to check that this P satisfies the criterion described in 3.2.1 for a time-homogeneous Markov Chain on V.

It turns out in this setting, the criterion for reversibility is easy to characterize. Define $VolG = \sum_{x \in V} w(x)$, and $\pi_V(x) = \frac{w(x)}{VolG}$. Then, P is reversible with respect to π_V if and only if W is symmetric - that is, if and only if G is undirected. We refer a random walk constructed in this manner with respect to a notion of pairwise similarity, to be *the P (pairwise) chain* for that similarity.

3.2.3 The Edge Walk

For a data set modeled as a graph, the walk in 3.2.2 respects *pairwise* similarities in the data set. There might exist trinary similarities in the data set that do not depend on pairwise similarities defined by W. So, we can ask the

Question: How shall we construct a Markov Chain on the state space V that respect trinary relations?

The idea here is that if weighted edges encode pairwise similarities, then weighted faces should encode trinary relations. But, geometrically, faces should only communicate to with their induced boundary edges. So, we can rephrase our previous question by asking

Question: How shall we construct a Markov Chain on the state space of undirected edges that respect trinary relations?

We address how this random walk can be realized on the vertex set V in 3.2.4.

To answer our question, we turn the set of all undirected edges in G, $V \otimes_{sym} V = V^2$, itself into a weighted graph, so that these edges are the new vertices, and faces are the new edges connecting two new vertices, with the condition that two edges communicate with each other if and only if they are adjacent to a face.

More precisely, consider the weighted graph $H = (V^2, F, K)$ consisting of vertices V^2, edges F, and a weight function $K : F \to \mathbb{R}^+$ symmetric in the first two entries (the edges are undirected, and there are no self-faces of any kind). With a slight abuse of notation, we refer to K as a $|V^2| \times |V^2|$ matrix such that

$$K(e_{x,y}, e_{z,w}) = \begin{cases} K(x,y,z) & \text{if w = x or y} \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

which requires the use of symmetry in the first two entries of K.

Thus, we have a dual interpretation of K: the weight $K(x, y, z)$ is a measure of the three-wise similarity between the vertices x, y, z, and is the pairwise similarity between each $e_{x,y}$, $e_{x,z}$, $e_{y,z}$. If K is symmetric in entries 1 and 3, then it is fully symmetric, and we say the graph H is *face-undirected*. It is *face-directed* otherwise. In this paper, all such graphs are face face-undirected.

Now, if we're sitting at $e_{x,y}$ and have a face on $\{x, y, z\}$ we should be able to transition to $e_{y,z}$ and $e_{x,z}$ only through this face. Since this transition should be proportional to $K(x, y, z)$, this suggests that we define

$$k(x, y) = \sum_{z \in V} K(x, y, z) \tag{3.9}$$

so that

$$Q(e_{x,y}, e_{y,z}) = \frac{K(x, y, z)}{2k(x, y)} \quad \text{and} \quad Q(e_{x,y}, e_{x,z}) = \frac{K(y, x, z)}{2k(x, y)} \tag{3.10}$$

are the corresponding transition probabilities. It's easy to check that Q satisfies the criterion described in 3.2.1 for a time-homogeneous Markov Chain on V^2.

To discuss reversibility, define $\pi_E(e_{x,y}) = \frac{2k(x,y)}{VolH}$, where $VolH = 2\sum_{e \in V^2} k(e)$. Then, Q is reversible with respect to π_E if and only if H is face-undirected.

3.2.4 Pulling back the Edge Walk

Given the Markov chain on undirected edges constructed section 3.2.3,

Question: How shall we reqlize the edge walk on H as a vertex walk on G?

We answer this question in two suggestive ways.

The π Chain

The first is to simply run Q to its stationary distribution $\pi(e_{x,y})$, and to use these numbers as weights in a $V \times V$ similarity weight matrix. Roughly speaking: two

points will be more similar the greater $\pi\left(e_{x,y}\right)$ is. Doing so yields a time-homogenous chain, as constructed in section 3.2.2, with transition kernel

$$P_\pi(x,y) = \frac{\pi_E(e_{x,y})}{\sum_z \pi_E(e_{x,z})}.$$

It follows from section 3.2.2 that the stationary distribution of this chain is exactly the set of marginals,

$$\pi(x) = \sum_z \pi\left(e_{x,z}\right).$$

We refer to this vertex walk as *the* π *chain* obtained from Q.

The $E1$ *Chain*

The second: starting at $x \in V$, choose an edge according to π_E conditioned on being adjacent to x; starting from this edge, take a step according to Q, landing on an undirected edge, say $e_{z,w}$. Then, with probability $\frac{1}{2}$, proceed to z or w.

To make this more precise, let

$$\pi_{x,E}(e) = \begin{cases} \frac{\pi_E(e_{x,y})}{\sum_z \pi_E(e_{x,z})} & \text{if } e = e_{x,y} \\ 0 & \text{otherwise} \end{cases}$$

be the probability distribution of transitioning to an edge e adjacent to the vertex x. Then, define

$$P_{E1}(x,y) = \frac{1}{2}\sum_w (\pi_{x,E}Q)(e_{w,y}). \tag{3.11}$$

It's easy to show that P defines a time homogeneous transition kernel representing the aforementioned walk.

The stationary distribution and reversibility of P_{E1} is given by the following theorem.

Theorem 5. *1. Define $\pi_{VE}(x) = \frac{1}{2}\sum_z \pi_E(e_{x,z})$. Then, π_{VE} is a stationary measure for P_{E1} and $\mu P_{E1} \to \pi_{VE}$ as $t \to \infty$ for any probability measure μ on V.*

2. $\pi_{VE}(x)P_{E1}(x,y) = \pi_{VE}(y)P_{E1}(y,x)$.

For the proof, see section A.2.1. We call this chain *the $E1$ (edge iterated) chain* obtained from Q.

3.2.5 Putting it all Together

Consider two positive, symmetric, weight matrices on V, W_1 and W_2. How do we put them together to perform diffusive transductive inference?

To answer this question, we introduce three operators, the *combinatorial Laplacian*, *normalized Laplacian*, and *random walk Laplacian*:

$$L_{comb} = D - W, \quad L_{norm} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}, \quad L_{rw} = I - D^{-1}W. \tag{3.12}$$

where W is a symmetric nonnegative weight matrix, and $D = diag(w_i)$, $w_i = \sum_j W(i,j)$. For a given geometry (weight matrix), these operators encode different, but related, information.

We could add together their combinatorial Laplacians:

$$L_{comb} = cL_{1,comb} + (1-c)L_{2,comb}.$$

This would be equivalent to constructing a new weight matrix $W = W_1 + W_2$ and proceeding with diffusive transductive inference using a reversible Markov chain corresponding to W. Then, $W(x,y)$ would treat the information contained in W_1 and W_2 on the same footing, appropriately weighted. This approach has already been done in Zhou (2005).

One could also add the corresponding normalized Laplacians,

$$L_{norm} = cL_{1,norm} + (1-c)L_{2,norm}$$

but a convex linear combination of these operators does not again yield a normalized Laplacian (it could destroy the existence of a kernel).

Finally, we could add the random walk Laplacians

$$L_{rw} = cL_{1,rw} + (1-c)L_{2,rw}.$$

This is the same as taking a convex linear combination of the infinitesimal generators of the process P_1 and P_2. We immediately have from this that

$$I - L_{rw} = P_c = cP_1 + (1-c)P_2. \tag{3.13}$$

In other words, adding the random walk Laplacians induces a mixture of P_1 and P_2 with mixing probability c.

This is an approach which has not been used to date, and is the one we take. We now use P_c to perform diffusive transductive inference, using leave-K-out cross validation to determine the *two* values c and t, choosing P_1 and P_2 among the P, π and $E1$ chains constructed. Assuming we start at some state x, P_c transitions to y by flipping a coin, and with probability c chooses to transition to y through an edge with probability $P_1(x, y)$, and with probability $1 - c$ chooses to transition to y through an edge with probability $P_2(x, y)$. Thus, this method has the advantage allowing information to diffuse in directions which respect the geometry of pairwise and threewise relations, respectively: it separates the scales of information flow, and diffuses information across these scales according to the choice of c. However, there is a drawback (which coincidentally, also makes the theory interesting), P is evidentally *not* reversible.

3.3 Similarities

3.3.1 Shannon Entropy

We wish to perform transductive inference on data sets with higher order relations, using the random walks constructed in section 3.2. But first, we need to address the

following question:

What notions of similarity exhibit threewise similarities that are independent of pairwise similarities?

An answer to this important question is motivated by some elementary probability theory. Recall that two events A and B are *independent* if

$$\mathbb{P}\left[A \text{ and } B\right] = \mathbb{P}\left[A\right]\mathbb{P}\left[B\right].$$

In terms of conditional probability $\mathbb{P}\left[A|B\right] = \frac{\mathbb{P}[A \text{ and } B]}{\mathbb{P}[B]}$, this can be rephrased as

$$\mathbb{P}\left[A|B\right] = \mathbb{P}\left[A\right], \qquad \mathbb{P}\left[B|A\right] = \mathbb{P}\left[B\right].$$

Of course the notion of pairwise independence is symmetric: A is independent from B if and only if B is independent from A.

More generally, a collection of events $\{A_i\}_{i=1}^n$ are independent (or n-wise independent) if

$$\mathbb{P}\left[A_1 \text{ and } A_2 \text{ and} \cdots A_n\right] = \mathbb{P}\left[A_1\right] \cdots \mathbb{P}\left[A_n\right]$$

and of course, this definition is symmetric in $A_1, \ldots, A_n$.

An important property of independence is that if a collection of events are n-wise independent, then every $n-1$ subcollection is $(n-1)$-wise independent. However, the converse is *not* true.

To see this, consider the random variables constructed as follows. X_1 is the toss of a fair two-sided coin, and X_2 is an (independent) toss of the same coin. Let X_3 be a random variable that takes the values heads if $(X_1, X_2) = (H, H)$ or (T, T) and tails otherwise. One can easily check that while the random variables X_1, X_2, X_3 are pairwise independent, they are not three-wise independent.

So, we can use independence to encode our notions of pairwise and three-wise similarity. By doing this, we interpret the data set as a collection of random variables having been sampled a suitable number of times.

Pairwise independence can be captured by using mutual entropy, or Shannon entropy. Given two discrete random variables X and Y with distributions $p_{X,i} = \mathbb{P}[X = i]$, $p_{Y,i} = \mathbb{P}[Y = i]$, and $p_{i,j} = \mathbb{P}[X = i, Y = j]$, define the *pairwise Shannon entropy* to be

$$H(X,Y) = \sum_{i,j} p_{i,j} \log\left(\frac{p_{i,j}}{p_{X,i}p_{Y,j}}\right). \tag{3.14}$$

From this definition follows that $H(X,Y)$ is zero if and only if X and Y are independent, that $H(X,Y) \leq H(X), H(Y)$, and that $H(X,Y)$ is symmetric in X and Y.

For three random variables X, Y, Z with distributions $p_{X,i}$, $p_{Y,i}$, $p_{Z,i}$, $p_{i,j,k} = \mathbb{P}[X = i, Y = j, Z = k]$ define the *three-wise Shannon entropy* to be

$$H(X,Y,Z) = \sum_{i,j,k} p_{i,j,k} \log\left(\frac{p_{i,j,k}}{p_{X,i}p_{Y,j}p_{Z,k}}\right). \tag{3.15}$$

From this definition follows that $H(X,Y,Z)$ is zero if and only if X, Y, and Z are independent, that $H(X,Y,Z) \leq H(X), H(Y), H(Z)$, and that $H(X,Y,Z)$ is symmetric in X, Y, and Z.

Using these concepts, we define the notion of similarity between two random variables to be

$$W_0(x,y) = \exp\left[\left(\frac{1 - \frac{H(X,Y)}{\max\{H(X),H(Y),H(Z)\}}}{\epsilon}\right)^2\right] \tag{3.16}$$

and the similarity of three random variables to be

$$W_1(x,y,z) = \exp\left[\left(\frac{1 - \frac{H(X,Y,Z)}{\max\{H(X),H(Y),H(Z)\}}}{\epsilon}\right)^2\right] \tag{3.17}$$

where ϵ is a thresholding parameter. These definitions have the advantage of making independent random variables far apart, while being localizing and maintaining

symmetry. Also, and very important, it's clear from the definitions that random variables are k-wise independent if and only if the corresponding k-wise Shannon entropy vanishes. We refer to the normalized entropies appearing in the arguments in (3.16) and (3.17) as *normalized* k-wise Shannon entropies.

For the proof of the above facts and more, we refer the reader to Gray (1990).

The use of Shannon entropy stems from the case when the random variables in question take on values which aren't numbers. For random variables taking values in an alphabet, we can't necessarily compute means and covariances, but we can compute joint distributions, in which case Shannon entropy is useful. It's also useful for random variables taking on values on a scale, where the value of each scale might not have any meaning, but the distribution of points at each scale does. It's also a stronger way of measuring the correlation of random variables compared to correlations, since entropy is purely positive, and measures dependence directly by keeping track of the joint distributions.

It should be noted, however, that it is computationally more expensive, than, say computing correlations or tricorrelations (section 3.3.2), since it requires computing joint distributions of random variables.

3.3.2 Correlations

Recall from probability that the *covariance of two random variables* X *and* Y is defined to be

$$C(X,Y) = \mathbb{E}\left[(X-\mu_X)(Y-\mu_Y)\right] = \mathbb{E}\left[XY\right] - \mathbb{E}\left[X\right]\mathbb{E}\left[Y\right].$$

The *correlation coefficient of* X *and* Y is then

$$c(X,Y) = \frac{C(X,Y)}{\sigma_X \sigma_Y}$$

where σ_X and σ_Y are the standard deviations of X and Y, respectively. It can be shown using the Cauchy-Schwartz inequality that $|c(X,Y)| \leq 1$ for all random

variables.

Now, suppose that three random variables are pairwise independent. We've shown in section 3.3.1 that they might not be threewise independent. How can we capture such a dependency?

Define the *trivariance* of the random variables X, Y and Z to be

$$\begin{aligned} T(X,Y,Z) &= \mathbb{E}\left[(X-\mu_X)(Y-\mu_Y)(Z-\mu_Z)\right] \\ &= \mathbb{E}\left[XYZ\right] - \mu_X\mu_Y\mu_Z - \left(\mu_X C(Y,Z) + \mu_Y C(X,Z) + \mu_Z C(X,Y)\right). \end{aligned} \tag{3.18}$$

and the *tricorrelation coefficient* of X, Y, and Z to be

$$t(X,Y,Z) = \frac{T(X,Y,Z)}{\sigma_X\sigma_Y\sigma_Z}$$

where μ_X, μ_Y, μ_Z and σ_X, σ_Y, σ_Z are the mean and standard deviation of the random variables X, Y, and Z respectively. In applications, we'll only keep the first 20 nearest neighbors of each point using pairwise similarities. This means that the remaining random variables have a zero effective exponential correlates. Thus, the threewise similarity as described above captures the three-wise dependency exactly.

Having established this, we define the notion of similarity between two random variables to be

$$W_0(x,y) = \exp\left[\left(\frac{1-|c(X,Y)|}{\epsilon}\right)^2\right]. \tag{3.19}$$

Since there is no meaningful upper bound on $|t(X,Y,Z)|$ as there is in $|c(X,Y)|$, we define the three-wise notion of similarity to be:

$$W_1(x,y,z) = \exp\left[\left(\frac{\max_W |t(X,Y,W)| - |t(X,Y,Z)|}{\epsilon}\right)^2\right] \tag{3.20}$$

where ϵ is a thresholding parameter. These definitions have the advantage of making independence random variables far apart, while being localized and maintaining symmetry. This notion of similarity is particularly useful when the values of the random variables themselves have meaning. Furthermore, these quantities are all computationally fast. However, the drawback of these exponential kernels is that it treats positive and negative correlations the same since we take absolute values, unlike entropy.

3.4 Concentration

A natural question at this point using any of the notions of similarity we've discussed is how many samples are necessary to ensure what we measure empirically is close to what it actually should be. As discussed in Szlam (2008), when the data resembles a Riemannian manifold, we expect diffusive inference to work better, so one might perform a dimensionality procedure like PCA to achieve this. However, this reduces the number of samples, increasing the error in how close the empirical Shannon entropies/correlations are to their true values. What's the error? In this section, we derive concentration inequalities to answer this question precisely.

First, we prescribe how many samples are necessary to estimate the actual joint distribution of a family of m random variables with high probability.

Theorem 6. *Let $p_n = (p^{(n)}_{i_1,\ldots,i_m})$ be the empirical joint distribution vector for m random variables, $X_1, \ldots, X_m$ obtained from n samples, and let $p = (p_{i_1,\ldots,i_m})$ be the actual joint distribution. More precisely,*

$$p_{i_1,\ldots,i_m} = \mathbb{P}\left[X_1 = i_1, \ldots, X_m = i_m\right]$$

where $i_k = 1, \ldots, N_{X_k}$, and $p_n = \frac{1}{n}\sum_{k=1}^{n} \mathbf{X_k}$ are the empirical probabilities obtained from n samples, where $\mathbf{X_k}$ are independent multivariate Bernoulli random variables

with probabilities $p_{i_1,\ldots,i_m}$. *Furthermore, define the total variation distance between probability mass functions* p *and* q *by*

$$||p-q||_{TV} = \frac{1}{2}\sum_{i=1}^{N}|p(i)-q(i)|.$$

Then,

$$\mathbb{P}\left[||p_n - p||_{TV} > \epsilon\right] \leq 2e^{-\frac{n\epsilon^2}{2\prod_{k=1}^{m} N_{X_k}}}.$$

Next, we prescribe how many samples are necessary to estimate the actual Shannon entropy of a family of m random variables with high probability.

Theorem 7. *Given the same notation in Theorem 6, define*

$$H_n(X_1,\ldots,X_m) = \sum_{i_1,\ldots,i_m} p^{(n)}_{i_1,\ldots,i_m} \log \frac{p^{(n)}_{i_1,\ldots,i_m}}{\prod_{k=1}^{m} p^{(n)}_{X_k}(i_k)}$$

and

$$H(X_1,\ldots,X_m) = \sum_{i_1,\ldots,i_m} p_{i_1,\ldots,i_m} \log \frac{p_{i_1,\ldots,i_m}}{\prod_{k=1}^{m} p_{X_k}(i_k)}$$

to be the m*-wise Shannon entropy of* $X_1,\ldots,X_m$, *where* p_{X_k} *and* $p^{(n)}_{X_k}$ *denote the* k^{th} *marginal distributions of* p *and* p_n, *respectively. Let*

$$Z^{(n)}_{i_1,\ldots,i_m} = \sum_{k=1}^{n} (\mathbf{X_k})_{i_1,\ldots,i_m}$$

be the multinomial distribution with parameters $(n, p_{i_1,\ldots,i_m})$ *giving the number of the* n *samples that equal the* m*-tuple,* $(i_1,\ldots,i_m)$. *Then, if* $p_{i_1,\ldots,i_m} \neq 0$, $e \leq \mathbb{E}\left[Z^{(n)}_{i_1,\ldots,i_m}\right]$ *and* $e^2 \leq n$,

$$\mathbb{P}\left[|H_n(X_1,\ldots,X_m) - H(X_1,\ldots,X_m)| > \epsilon\right] \leq e^{-8\frac{n}{(\log n-1)^2}\cdot\frac{\epsilon^2}{(m+1)^2\prod_{k=1}^{m} N_{X_k}}}.$$

In practice, one will first use theorem 6 to reliable estimate the joint distribution and threshold appropriately those probabilities near zero. On those remaining nonzero probabilities then, one may use theorem 7 to (if necessary) obtain more samples to infer the Shannon entropy.

For trivariances, we present the following concentration inequality for bounded random vectors in $\mathbb{R}^d$.

Theorem 8. *Let X, Y and Z be random vectors in $\mathbb{R}^d$ bounded by M, and (x_i, y_i, z_i) be independent samples of (X, Y, Z). Define*

$$T(X,Y,Z) = \mathbb{E}\left[(X - \mathbb{E}\left[X\right]) \otimes (Y - \mathbb{E}\left[Y\right]) \otimes (Z - \mathbb{E}[Z])\right],$$

$$\begin{aligned} T(X_n, Y_n, Z_n) &= \mathbb{E}_n\left[(X - \mathbb{E}_n[X]) \otimes (Y - \mathbb{E}_n[Y]) \otimes (Z - \mathbb{E}_n[Z])\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}_n[X]) \otimes (y_i - \mathbb{E}_n[Y]) \otimes (z_i - \mathbb{E}_n[Z]), \quad (3.21) \end{aligned}$$

$$C_n(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}[X]) \otimes (y_i - \mathbb{E}[Y]).$$

and the two norms

$$\langle T, T\rangle = \sum_{i,j,k} T(i,j,k)T(i,j,k), \quad \text{and } \langle C, C\rangle = \sum_{i,j} C(i,j)C(i,j).$$

Suppose

$$\mathbb{E}\left[||(X - \mathbb{E}[X]) \otimes (Y - \mathbb{E}[Y]) \otimes (Z - \mathbb{E}[Z]) - T(X,Y,Z)||^p\right] \leq \frac{1}{2}p!b^2L^{p-2}, \;\; p \geq 2,$$

for two positive constants b and L. Then there exists a constant C so that

$$\mathbb{P}\left[||T(X_n, Y_n, Z_n) - T(X,Y,Z) > \epsilon\right] \leq 10\exp\left(-\frac{n\epsilon^2}{CM^6d^3}\right).$$

For general results and a proof of theorems 6, 7 and 8 see sections A.2.2, A.2.3, A.2.4.

3.5 Applications

In the applications that follow, we take the random variable approaches described above.

For pairwise similarities, we keep only the 20 nearest edge neighbors of each vertex. In the construction of the $E \times E$ similarity matrix, we also keep 20 nearest neighbors, but only keep 10 nearest neighbors when we pull it back to the vertex set. This ensures that our correction is, in fact, higher order, and does not compete at the same scale as the 1st order diffusion. We choose ϵ in the exponential kernels in section 3.3 by autotuning: we choose the 5^{th} largest value of the 10 nearest neighbors for each point, and symmetrize the corresponding adjacency matrix.

In all examples, there are 12 splits of training data, and all cross-validation and test errors are reported as averages of those cross-validation and test errors derived from the splits. Finally, whenever there are multiple minimum values for the cross-validation error, we take the median t for which these values occur, and then the median c for that time, if necessary. We increment c in steps of 0.1 from 0 to 1, and t in steps of 1 from 0 to 10.

We should remark that for very large data sets, computation of the tricorrelation tensors can require time to compute and large amount of memory to store. In this case, one may want to threshold the entries.

We summarize this algorithm in table 3.3.

3.5.1 Synthetic Bayesian Networks

First, let's address when we expect our method to work. Figure 3.3(a) shows a gene network modeled as a Bayesian graphical model consisting of five bowties, each vertex which is a Bernoulli$\left(\frac{1}{2}\right)$ random variable (on or off). The network is

Table 3.3: Our algorithm for semi-supervised learning with higher order relations.

Input: two weight matrices W_1 and W_2 via section 3.2.2 and 3.2.4 using any similarities in 3.3, an integer K, class labels $\{c_i\}_{i=1}^n$, labeled, and unlabeled data. Compute $P_1 = D_1^{-1}W_1$. Compute $P_2 = D_2^{-1}W_2$. For each class label c_i, use leave-K-out cross validation to construct indicator functions χ_i on the training set, leaving K out. Use K-out cross-validation on the remaining training data to infer 1) the value of c in $P(c) = cP_1 + (1-c)P_2$, 2) the number of powers to take $P(c)^t$, that minimize the test error on the training set. Use this (c, t) to propagate labels to the unlabeled data. Output: class labels on the unlabeled data.

constructed so that every pair of vertices are pairwise independent, but the three groups of vertices in the left, center, and right of each bowtie are not three-wise independent. The geometry of these structures under the π and E_1 chains translates into the undirected network in figure 3.3(b).

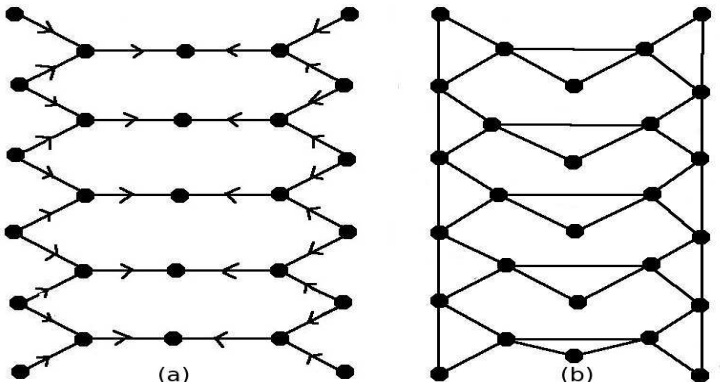


FIGURE 3.3: A Bayesian network of pairwise independent genes. (a) is the Bayesian network modeling the gene regulatory network. Each gene is Bernoulli$\left(\frac{1}{2}\right)$ and the network is designed so that all vertices are pairwise independent, but the left, middle, and right groups of three are not three-wise independent. (b) is the manifestation of this network as an undirected graph whose geometry is determined by the reversible random walks P_π and P_{E1}.

In this simple example, the weights on all the edges in figure 3.3(b) are one. Due to the symmetry of the figure, it's clear that the best cut to separate the graph into two balanced pieces should be to cut along the center of the graph (which also makes sense due to the symmetry in figure 3.3(a)). However, since we do not like vertex cuts, our labeling will place a -1 on the left side of the cut, a 1 on the right side, and a 0 for those vertices in the center.

In the computations that follow, we consider a similar bowtie network with 200 bowties (1002 vertices) having been sampled 500 times. With the labeling as described above, and using 12 splits of 100 vertices as a training set and leave-99-out cross validation, we perform our algorithm using correlations, and report the results in tables 3.4 and 3.5. We note that we threshold pairwise correlations below 0.1 and threewise correlations below 0.2 so that when choosing the autotuning parameter, one does not renormalize noise.

Table 3.4: Summary of mixing chain percentage errors in the synthetic gene network with no pairwise correlations for different chains.

	P Chain		π Chain		$E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error	CV Error	Test Error
Correlations	64.5%	59.52%	57.25%	57.02%	57.25%	56.59%

Table 3.5: Summary of percentage errors in the synthetic gene network with no pairwise correlations using the mixed chains.

	$P-\pi$ Chain		$P-E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error
Correlations	56.92%	56.5%	56.92%	57.08%

It's clear that in this case, any of our algorithms is superior: the random walk using pairwise correlations will produce a degenerate diffusion, and we cannot expect to propagate class labels under such circumstances (there are no clusters for which to propogate labels to). The presence of induced edges via faces, however, clusters the underlying data set to propagate labels to, and results in a substantially lower

CV error, and lower test error. We present verification of this in the landscape of errors in figures 3.4 and 3.5.

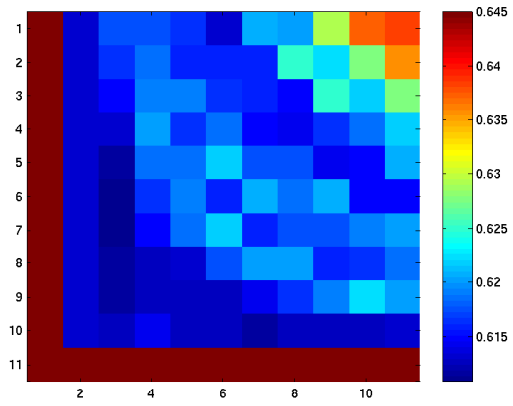


FIGURE 3.4: Cross-validation errors as a function of (c, t) in the bowtie data set without pairwise correlations, using the similarity of correlations and $P_c = cP_P + (1-c)P_{E1}$. Columns denote time starting from 0 in steps of 1 from left to right. Rows denote mixing probabilities c in increments of 0.1 from top to bottom.

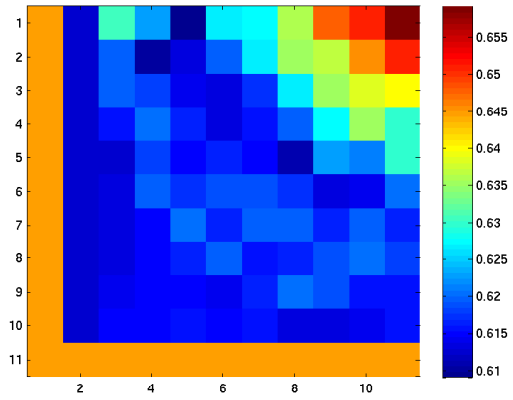


FIGURE 3.5: Cross-validation errors as a function of (c, t) in the bowtie data set without pairwise correlations, using the similarity of correlations and $P_c = cP_P + (1-c)P_{\pi}$. Columns denote time starting from 0 in steps of 1 from left to right. Rows denote mixing probabilities c in increments of 0.1 from top to bottom.

This is in the absence of pairwise correlations (real edges). How does our algorithm perform when there are more faces than edges?

Consider a slight modification of the network in figure 3.3 presented in figure 3.6.

Figure 3.6(a) shows a gene network modeled as a Bayesian graphical model consisting of five bowties, each vertex of which is uniform on the integers $1-5$ describing five conformal states of the gene. The network is constructed so that all vertices except the middle three are pairwise independent, and the left and right groups of vertices are pairwise but not three-wise independent. This can be seen by the geometry of the P_P walk and the P_π/P_{E1} walk in figures 3.6 (b) and (c), respectively.

In the computations that follow, we consider a similar bowtie network with 200 bowties (1002 vertices) having been sampled 500 times. With the same labeling as described above, and using 12 splits of 100 vertices as a training set and leave-99-out cross validation, we perform our algorithm using corelations, and report the results in tables 3.6 and 3.7. Again, we note that we threshold pairwise correlations below 0.1 and threewise correlations below 0.2 so that when choosing the autotuning parameter, one does not renormalize noise.

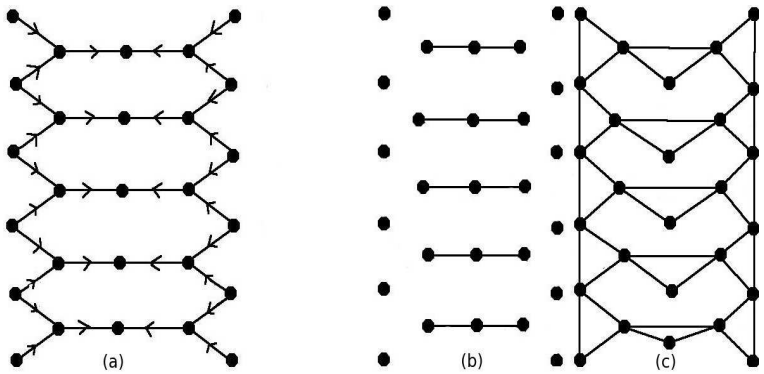


FIGURE 3.6: A Bayesian network of pairwise independent genes. (a) is the Bayesian network modeling the gene regulatory network. Each gene is uniform on the integers $1-5$ describing five conformal states of the gene. The network is is designed so that all vertices except the middle three are pairwise independent, and the left and right groups of vertices are pairwise but not three-wise independent. (b) is the manifestation of this network as an undirected graph whose geometry is determined by the reversible random walks P_P. (c) is the manifestation of this network as an undirected graph whose geometry is determined by the reversible random walks P_π and P_{E1}.

Table 3.6: Summary of mixing chain percentage errors in the synthetic gene network with pairwise correlations correlations for different chains.

	P Chain		π Chain		$E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error	CV Error	Test Error
Correlations	69.92%	63.91%	60.00%	64.68%	60.00%	63.73%

Table 3.7: Summary of percentage errors in the synthetic gene network with pairwise correlations for mixed chains.

	$P-\pi$ Chain		$P-E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error
Correlations	59.58%	65.04%	59.58%	64.33%

In this case, our algorithm always has a substantially lower CV error. In the test error regime, however, it sometimes if at all performs better. The CV error will be substantially lower since P_P can only make the values along the middle three vertices the same (these are the only clusters according to P_P), while the other chains will diffuse information correctly along either side of the bowtie. Our test error will, however, be comparable to the test error for P_P since the edges induced by faces don't cluster the data much better than the true edge connections. We present verification of this in the landscape of errors in figures 3.7 and 3.7.

This example does, however, illustrate when we can expect our algorithm to work in the presence of edges: face clusters must be at least as good as edge clusters, and class labels must respect these clusters.

3.5.2 Handwritten Digits

In this section, we present the results of our technique on a set of handwritten images obtained from the MNIST repository LeCun (1998). Original images are 28×28 images of the handwritten values $0-9$ with pixel values between $0-255$.

We consider 1000 instances of the values $3, 8$ and $1, 2$, respectively, and so obtain a data set which is a 784×1000 matrix consisting of the values $0-255$. Finally,

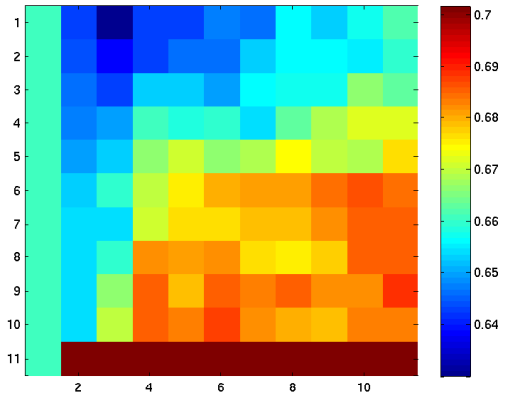


FIGURE 3.7: Cross-validation errors as a function of (c, t) in the bowtie data set with pairwise correlations, using the similarity of correlations and $P_c = cP_P + (1-c)P_{E1}$. Columns denote time starting from 0 in steps of 1 from left to right. Rows denote mixing probabilities c in increments of 0.1 from top to bottom.

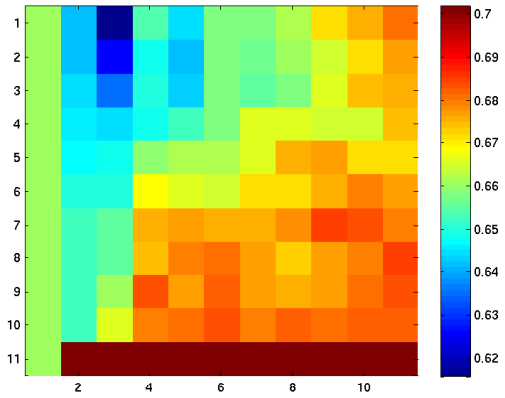


FIGURE 3.8: Cross-validation errors as a function of (c, t) in the bowtie data set with pairwise correlations, using the similarity of correlations and $P_c = cP_P + (1-c)P_\pi$. Columns denote time starting from 0 in steps of 1 from left to right. Rows denote mixing probabilities c in increments of 0.1 from top to bottom.

we use a training set consisting of 100 randomly chosen digits with at least 10 from each class and cross-validate each entry by leave-99-out. Results using correlations for $3, 8$ are presented in tables 3.8 and 3.9 and results for $1, 2$ using correlations are presented in tables 3.10 and 3.11.

Table 3.8: Summary of mixing chain percentage errors in the MNIST38 data set for different chains.

	P Chain		π Chain		$E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error	CV Error	Test Error
Correlations	4.17%	4.19%	10.17%	10.77%	10.5%	10.47%

Table 3.9: Summary of percentage errors in the MNIST38 data set using mixed chains.

	$P - \pi$ Chain		$P - E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error
Correlations	4.25%	5.31%	4.25%	5.16%

Table 3.10: Summary of mixing chain percentage errors in the MNIST12 data set for different chains.

	P Chain		π Chain		$E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error	CV Error	Test Error
Correlations	1.08%	1.77%	18.75%	20.08%	18.42%	19.31%

Table 3.11: Summary of percentage errors in the MNIST12 data set using mixed chains.

	$P - \pi$ Chain		$P - E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error
Correlations	1.83%	2.73%	1.42%	2.4%

3.5.3 Yeast Genes

This data set was derived from Ideker et al. Ideker (2001). Here, four replicate hybridizations were performed for each cDNA array experiment. 205 genes were reproducibly measured and the produced expression patters reflect four functional categories in the Gene Ontology listings [Ashburner (2000)].

There are four total classes, and our training set consists of knowledge of 20 points in total. We keep at least three genes from each class in each of the 12 splits. We use leave-19-out cross validation to determine the values of c and t used in the testing phase. In using correlations, the data set is reformatted so that we use their z-scores instead. When we use Shannon entropy, we declare that an attribute has a value one (on) if it falls within one standard deviation of its mean, and zero otherwise (off). In this fashion, we may interpret each instance as a random variable: we pick an attribute uniformly at random and report whether it was on or off. That way, two instances being correlated is the same as two randomly selected attributes being correlated.

Our results are presented in tables 3.12 and 3.13, and remark that using the $P - E1$ chain yields a lower test error when using correlations.

Table 3.12: Summary of mixing chain percentage errors in the yeast gene data set for different chains.

	P Chain		π Chain		$E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error	CV Error	Test Error
Correlations	33.33%	37.3%	45.42%	43.92%	46.25%	43.11%
Mutual Information	40.42%	40.23%	55%	67.34%	54.58%	65.36%

Table 3.13: Summary of percentage errors in the yeast gene data set using mixed chains.

	$P - \pi$ Chain		$P - E1$ Chain	
Similarity	CV Error	Test Error	CV Error	Test Error
Correlations	30.83%	38.38%	30.42%	36.22%
Mutual Information	45.42%	55.77%	42.92%	50.32%

3.5.4 The Benchmark Data Sets

Next, we consider the so-called benchmark data sets. These data sets were prepared by Chapelle (2010) in testing the performance of various semisupervised learning algorithms. Here, we use only the raw data, and take the 12 splits as given. We

reformat the data set using PCA so that the data points are random vectors in $\mathbb{R}^{50}$. For completeness, we include the performance of our algorithm on these data sets using correlations in tables A.1 and A.2 in section A.2.5. Comparisons to the state-of-the-art methods and regular diffusive inference is given in table A.3 in section A.2.5.

3.6 Conclusions

When we consider networks of random variables, the role of the vertices as random variables is pivotal to any question regarding the graph. In particular, if we're interested in questions about dependency structures of the underlying random variables, pairwise similarities are simply not enough, since as shown in section 3.3, a node might be edge-isolated from the rest of the graph because it's pairwise independent, but might not be face-wise isolated since it could be threewise dependent. As such, there needs to exist algorithms that tackle this phenomena.

Our method does this. In contrast to Zhou (2005), we treat information arising from pairwise and threewise relations as fundamentally different types instead of adding their contributions directly. This gives rise to two different types of clusters, one reflecting pairwise relationships, the other reflecting threewise relations. By interpolating between these clusters by mixing their random walk, we diffuse class labels in order to reflect this interplay, and show that it can work when face clusters are at least as good as edge clusters with class labels respecting these geometries.

However this random variable interpretation of a data set merits caution. In the case of time series data such as the yeast galactose data set, or the prostate cancer patients, the random variable interpretation certainly seems reasonable, and we outperform the pairwise theory. In other cases, however, as demonstrated with some of the benchmark data sets, and MNIST data set, this assumption might not be a good one, since it's not readily understood whether correlations is a suitable

metric to cluster the underlying data - a necessary condition for the success of any semi-supervised learning algorithm. Therefore, the validity of our approach should be guided by the nature of the data set in question.

When the random variable interpretation is appropriate, we prove concentration inequalities to quantify the error of empirical quantities to their true value. This is very important, especially when one employs dimensionality reduction techniques like PCA which reduces the number of samples but makes the data more manifold-like and smoother.

Future work includes other ways to construct the random walk Q constructed in section 3.2.3 other than from threewise similarities. Furthermore, by pulling back Q to a reversible random walk P and its weight matrix W_E we obtain a one parameter family of similarities (metrics/weighted networks) on the data set. We've already seen how some of these metrics perform better than the ad hoc ones, and further investigations as to why this happens is certainly encouraged.

The required condition that face clusters are at least as good as edge clusters with class labels respecting these geometries also brings up the question of how one can perform diffusive transductive inference with a kernel that is not symmetric, since for almost all values of $c \in (0, 1)$ this will be true for $P(c) = cP_0 + (1-c)P_1$. Developing a precise definition of a cluster with respect to these types of nonreversible random walks will allow one to check beforehand if the two types of similarities chosen will in fact yield better results than just using pairwise diffusive inference on either similarity individually.

This issue also gives rise to what can be said in the unsupervised case. In this approach, we obtain a one parameter family of networks that describe the data set, and in the semi-supervised case, we can use cross-validation using training data to determine which network best fits the data. In the unsupervised approach, we have no such validation scheme, and which parameter to choose remains open. Leaving

c as an tuning parameter can give rise to multiscale clustering, or clustering that interpolates between two different similarities in the data, but since the random walks are non-reversible, the meaning of such clusters will need to be examined, as well as if class labeling respects such clusters. An answer could be how to embed the data set using the non-reversible walks into $\mathbb{R}^n$, since then, we can visualize the clusters and get at least an intuitive idea as to what they mean.

Finally, and probably the most important future direction, would be what the replacement for threewise entropy and tricorrelations should be in the case where the random variable interpretation fails. In some data sets, higher order similarities can be prescribed beforehand, but in the absence of such data, we need to discover these features. Any notion of such higher order structures should be derived from the data set in question, and so how to learn if such structures exist, what they are, and how they are incorporated into the geometry of the data set, is important. While we know what these features are for data sets of random variables, the authors do believe that some of the ideas presented here will be valuable in future research in this direction.

Appendix A

Appendix: Proof of Selected Theorems

A.1 Chapter 2

A.1.1 Theorem 1

Theorem 1. *Let P be an irreducible, aperiodic Markov Jump transition matrix for a continuous time process X_t with $exp(1)$ holding times on a finite state space V and stationary measure π. Define $Q = P|_{\mathcal{C},\mathcal{C}}$ and $N = (I - Q)^{-1}$ where $\mathcal{C} \subseteq V$ is an irreducible proper subset.. Assume Q is irreducible and aperiodic, and let i and j be two boundary vertices of $\mathcal{C}$. Then the following statements hold for X_t:*

1. *The probability of starting at i and leaving through j is given by*

$$N_{i,j} \sum_{k \in \mathcal{C}^c} P(j,k). \tag{A.1}$$

2. *If s_j are the number of jumps taken to leave through j,*

$$\mathbb{P}_i [s_j = \ell] = \frac{Q^{\ell-1}(i,j)}{N_{i,j}}, \quad \ell = 1, \dots \tag{A.2}$$

3. *If τ_j is the time taken to leave $\mathcal{C}$ through j, then*

$$\mathbb{E}_i\left[\tau_j^k\right] = \frac{k!(I-Q)^{-(k+1)}(i,j)}{N_{i,j}}. \tag{A.3}$$

4. *In what follows, further assume that P is reversible.*

   For $x \in \mathcal{C}$, define

$$\sigma(x) = \frac{\pi(x)}{\sum_{y\in\mathcal{C}}\pi(y)} \tag{A.4}$$

   to be a probability measure on $\mathcal{C}$. Then, there exists an orthonormal basis of Q of eigenvectors with real eigenvalues $\{\lambda_k, f_k\}$ with respect to the inner product

$$\langle f, g\rangle = \sum_{x\in\mathcal{C}} f(x)g(x)\sigma(x). \tag{A.5}$$

   If $\{\lambda_k\}$ are sorted descending in absolute value and

$$g_{i,j,n} = \sum_{\ell=0}^{n} Q^\ell(i,j) + \sum_{\ell=n+1}^{\infty} f_1(i)f_1(j)\sigma(j)\lambda_1^\ell = \sum_{\ell=0}^{n} Q^\ell(i,j) + f_1(i)f_1(j)\sigma(j)\frac{\lambda_1^{n+1}}{1-\lambda_1} \tag{A.6}$$

   then, $0 < \lambda_1 < 1$ has exactly one eigenvector, $f_1(x) > 0$ for all $x \in \mathcal{C}$, $|\lambda_z| < \lambda_1$ for $z \neq 1$ and

$$|N_{i,j} - g_{i,j,n}| \leq \sqrt{\frac{\sigma(j)}{\sigma(i)}}\frac{|\lambda_2|^{n+1}}{1-|\lambda_2|}. \tag{A.7}$$

5. *Finally, let $f_{i,j}$ be the density of τ_j starting at i. Then,*

$$f_{i,j}(x) = \sum_{\ell=1}^{\infty} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right]. \tag{A.8}$$

where $\gamma_\ell(x) \sim \Gamma(\ell, 1)$.

Set

$$f_{i,j,n}(x) = \rho_{i,j,n} \left[\frac{1}{\rho_{i,j,n}} \sum_{\ell=1}^{n} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] \right] + (1 - \rho_{i,j,n}) \sum_{\ell=0}^{\infty} \gamma_{n+1+\ell}(x) \lambda_1^\ell (1 - \lambda_1) \tag{A.9}$$

where $\rho_{i,j,n} = \sum_{\ell=1}^{n} \mathbb{P}_i \left[s_j = \ell \right]$. *Then,*

$$||f_{i,j} - f_{i,j,n}||_{TV} \leq \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1 - |\lambda_2|)} \tag{A.10}$$

where $n \geq \min\{\ell : \mathbb{P}_i \left[s_j = \ell \right] \neq 0\}$.

6. *(Multiscaling) Let* $\phi_n = *_{i=1}^{n} \phi$ *be the n-fold convolution of* ϕ *and set*

$$f_{i,j}(x) = \sum_{\ell=1}^{\infty} \phi_\ell(x) \mathbb{P}_i \left[s_j = \ell \right],$$

$$f_{i,j,n}(x) = \rho_{i,j,n} \left[\frac{1}{\rho_{i,j,n}} \sum_{\ell=1}^{n} \phi_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] \right] + (1 - \rho_{i,j,n}) \sum_{\ell=0}^{\infty} \phi_{n+1+\ell}(x) \lambda_1^\ell (1 - \lambda_1)$$

Then,

$$||f_{i,j} - f_{i,j,n}||_{TV} \leq \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1 - |\lambda_2|)}.$$

If

$$\tilde{f}_{i,j,n} = \rho_{i,j,n}\left[\frac{1}{\rho_{i,j,n}}\sum_{\ell=1}^{n}\tilde{\phi}_\ell(x)\mathbb{P}_i\left[s_j=\ell\right]\right] + (1-\rho_{i,j,n})\sum_{\ell=0}^{\infty}\tilde{\phi}_{n+1+\ell}(x)\lambda_1^\ell(1-\lambda_1)$$

where $\tilde{\phi} = *_{i=1}^{n}\tilde{\phi}$, *and* $\tilde{\phi}$ *has*

$$\left|\left|\phi - \tilde{\phi}\right|\right|_{TV} < \epsilon$$

then,

$$\left|\left|f_{i,j} - \tilde{f}_{i,j,n}\right|\right|_{TV} \leq n\epsilon + \sqrt{\frac{\sigma(j)}{\sigma(i)}}\frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)} + \frac{(1+\rho_{i,j,n})\epsilon(1-\epsilon)^n}{1-(1-\epsilon)\lambda_1}.$$

7. *Let* $f_{i,j}(x)$ *be the density of* τ_j *starting at* i *as in (A.8) for two vertices* $i, j \in \mathcal{C}$. *Let* $\{\lambda_z, f_z\}$ *be the eigenvalues and orthonormal basis of eigenvectors for* Q. *Then,*

$$f_{i,j}(x) = \sum_{\ell=1}^{\infty}\gamma_\ell(x)\mathbb{P}_i\left[s_j=\ell\right] = \frac{1}{N_{i,j}}\sum_z f_z(i)f_z(j)\sigma(j)e^{-x(1-\lambda_z)}$$

$$= \frac{1}{N_{i,j}}\sum_z \frac{f_z(i)f_z(j)\sigma(j)}{1-\lambda_z}(1-\lambda_z)e^{-x(1-\lambda_z)}. \quad \text{(A.11)}$$

8. *Let* $\mathcal{Q}_t$ *be the process* X_t *constrained to* $\mathcal{C}$. *Then, for* $x, y \in \mathcal{C}$,

$$q_t(x,y) = \mathbb{P}_x\left[\mathcal{Q}_t = y\right] = \sum_{n=0}^{\infty}Q^n(x,y)\frac{e^{-t}t^n}{n!}.$$

Proof of Theorem 1:

1. $Q^\ell(i,j)$ is the probability starting at i of taking ℓ jumps in $\mathcal{C}$ and reaching j. Clearly then,

$$Q^\ell(i,j)\sum_{k\in\mathcal{C}^c}P(j,k)$$

is the probability of starting at i, taking ℓ jumps in $\mathcal{C}$ and then leaving through j. The probability of starting at i and leaving through j is then given by

$$\sum_{\ell=0}^{\infty}Q^\ell(i,j)\sum_{k\in\mathcal{C}^c}P(j,k)=(I-Q)^{-1}(i,j)\sum_{k\in\mathcal{C}^c}P(j,k)=N_{i,j}\sum_{k\in\mathcal{C}^c}P(j,k)$$

which is exactly (A.1), where we've used the well known fact that

$$N_{i,j}=(I-Q)^{-1}_{i,j}=\sum_{\ell=0}^{\infty}Q^\ell(i,j).$$

2. Given (A.1) then, and the fact that for a boundary vertex j, $\sum_{k\in\mathcal{C}^c}P(j,k)\neq 0$ so that (A.1) is nonzero,

$$\mathbb{P}_i\left[s_j=\ell+1\right]=\frac{Q^\ell(i,j)\sum_{k\in\mathcal{C}^c}P(j,k)}{N_{i,j}\sum_{k\in\mathcal{C}^c}P(j,k)}=\frac{Q^\ell(i,j)}{N_{i,j}} \tag{A.12}$$

which is (A.2).

3. For (A.3), in addition to (A.2), we'll also need the following Lemma.

Lemma 1. *Let ℓ and k be a positive integers, and $X\sim\Gamma(\ell,1)$. Then, $\mathbb{E}\left[X^k\right]=k!\binom{\ell+k-1}{\ell-1}$.*

Proof of Lemma 1:

$$\mathbb{E}\left[X^k\right]=\int_0^\infty x^k\frac{x^{\ell-1}\exp(-x)}{\Gamma(\ell)}dx=\frac{1}{\Gamma(\ell)}\int_0^\infty x^{\ell+k-1}\exp(-x)dx$$
$$=\frac{\Gamma(\ell+k)}{\Gamma(\ell)}=k!\binom{\ell+k-1}{\ell-1} \tag{A.13}$$

where we've used the fact that $\Gamma(n) = (n-1)!$ for positive integers n. $\triangle$

To show (A.3) then, let $X_\ell \sim \Gamma(\ell, 1)$ for positive integer ℓ and define s_j to be the number of jumps taken taken to leave $\mathcal{C}$. Then,

$$\mathbb{E}_i\left[\tau_j^k\right] = \mathbb{E}_i\left[\mathbb{E}_i\left[\tau_j^k | s_j\right]\right] = \mathbb{E}_i\left[\sum_{\ell=1}^{\infty} X_\ell^k(\omega)\mathbb{P}_i\left[s_j = \ell\right]\right] = \sum_{\ell=1}^{\infty}\mathbb{E}\left[X_\ell^k\right]\mathbb{P}_i\left[s_j = \ell\right]$$

$$= \frac{k!}{N_{i,j}}\sum_{\ell=1}^{\infty}\binom{\ell+k-1}{\ell-1}Q^{\ell-1}(i,j) = \frac{k!}{N_{i,j}}\sum_{\ell=1}^{\infty}\binom{\ell+k-1}{k}Q^{\ell-1}(i,j)$$

$$= \frac{k!}{N_{i,j}}\sum_{\ell=0}^{\infty}\binom{\ell+k}{k}Q^{\ell}(i,j) \quad \text{(A.14)}$$

$$= \frac{k!(I-Q)^{-(k+1)}(i,j)}{N_{i,j}} \quad \text{(A.15)}$$

where we've used the formula,

$$\frac{1}{(1-x)^{k+1}} = \sum_{\ell=k}^{\infty}\binom{\ell}{k}x^{\ell-k} = \sum_{\ell=0}^{\infty}\binom{\ell+k}{k}x^{\ell}.$$

4. For two functions f and g on $\mathcal{C}$,

$$\langle Qf, g\rangle = \sum_{z\in\mathcal{C}}(Qf)(z)g(z)\sigma(z) = \sum_{x,z\in\mathcal{C}}Q(z,x)f(x)g(z)\frac{\pi(z)}{\sum_{y\in\mathcal{C}}\pi(y)}$$

$$= \sum_{x,z\in\mathcal{C}}Q(x,z)f(x)g(z)\frac{\pi(x)}{\sum_{y\in\mathcal{C}}\pi(y)} = \sum_{x\in\mathcal{C}}f(x)(Qg)(x)\sigma(x) = \langle f, Qg\rangle \quad \text{(A.16)}$$

where in the third inequality, we've used reversibility of $Q = P_{\mathcal{C},\mathcal{C}}$ with respect to π.

So, with respect to this inner product, Q, is symmetric, so that by the spectral theorem, there exists an orthonormal basis of functions $\{f_k\}$ defined on $\mathcal{C}$ with real eigenvalues $\{\lambda_k\}$.

Now, order the eigenvalues $\{\lambda_k\}$ in decreasing order according to their absolute values. Since Q irreducible and aperiodic, there exists an R such that $Q^r(x,y) > 0$ for all $x, y \in \mathcal{C}$ and $r > R$. The Perron-Frobenius theorem implies that λ_1^r is positive for all $r > R$ so that λ_1 is positive, and implies that λ_1 has exactly one eigenvector, f_1, also which is positive. Furthermore, $|\lambda_z| < \lambda_1$ for all $z \neq 1$.

It's a well-known fact that since Q is substochastic (the jump process determined by P must leave $\mathcal{C}$ eventually) and that $Q^\ell \to 0$ so that $\lambda_1 < 1$.

Finally, to prove (A.7), let $\delta(y)$ denotes the indicator function on $y \in \mathcal{C}$. Then,

$$\delta(y) = \sum_{z \in \mathcal{C}} \langle \delta(y), f_z \rangle \, f_z = \sum_{z \in \mathcal{C}} f_z(y) \sigma(y) f_z \tag{A.17}$$

so that

$$Q^\ell(i,j) = \left(Q^\ell \delta(j)\right)(i) = \sum_{z \in \mathcal{C}} f_z(j)\sigma(j) \left(Q^\ell f_z(i)\right) = \sum_{z \in \mathcal{C}} f_z(j)\sigma(j)\lambda_z^\ell f_z(i). \tag{A.18}$$

Now,

$$N_{i,j} = \sum_{\ell=0}^{\infty} Q^{\ell}(i,j) = \sum_{\ell=0}^{n} Q^{\ell}(i,j) + \sum_{\ell=n+1}^{\infty} Q^{\ell}(i,j)$$

$$= \sum_{\ell=0}^{\infty} Q^{\ell}(i,j) + \sum_{\ell=n+1}^{\infty} \sum_{z=1}^{|\mathcal{C}|} f_z(i) f_z(j) \sigma(j) \lambda_z^{\ell}$$

$$= \sum_{\ell=0}^{n} Q^{\ell}(i,j) + \sum_{z=1}^{|\mathcal{C}|} f_z(i) f_z(j) \sigma(j) \frac{\lambda_z^{n+1}}{1-\lambda_z} \quad \text{(A.19)}$$

and

$$g_{i,j,n} = \sum_{\ell=0}^{n} Q^{\ell}(i,j) + \sum_{\ell=n+1}^{\infty} f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell}$$

$$= \sum_{\ell=0}^{n} Q^{\ell}(i,j) + f_1(i) f_1(j) \sigma(j) \frac{\lambda_1^{n+1}}{1-\lambda_1}$$

$$\Rightarrow N_{i,j} - g_{i,j,n} = \sum_{z=2}^{|\mathcal{C}|} f_z(i) f_z(j) \sigma(j) \frac{\lambda_z^{n+1}}{1-\lambda_z}$$

$$\Rightarrow \frac{|N_{i,j} - g_{i,j,n}|}{\sigma(j)} \leq \frac{|\lambda_2|^{n+1}}{1-|\lambda_2|} \sum_{z=2}^{|\mathcal{C}|} |f_z(i) f_z(j)|$$

$$\leq \frac{|\lambda_2|^{n+1}}{1-|\lambda_2|} \left[\sum_{z=1}^{|\mathcal{C}|} |f_z(i)|^2 \right]^{\frac{1}{2}} \left[\sum_{z=1}^{|\mathcal{C}|} |f_z(j)|^2 \right]^{\frac{1}{2}} \quad \text{(A.20)}$$

$$\sigma(x) = \langle \delta(x), \delta(x) \rangle = \left\langle \sum_{z \in \mathcal{C}} f_z(x) \sigma(x) f_z, \sum_{z \in \mathcal{C}} f_z(x) \sigma(x) f_z \right\rangle = \sigma^2(x) \sum_{z \in \mathcal{C}} |f_z(x)|^2 \quad \text{(A.21)}$$

$$\Rightarrow \frac{|N_{i,j} - g_{i,j,n}|}{\sigma(j)} \leq \frac{|\lambda_2|^{n+1}}{1-|\lambda_2|} \frac{1}{\sqrt{\sigma(i)\sigma(j)}} \tag{A.22}$$

$$\Rightarrow |N_{i,j} - g_{i,j,n}| \leq \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^{n+1}}{1-|\lambda_2|}. \tag{A.23}$$

5. To determine the density $f_{i,j}(x)$ for τ_j starting at i, notice that

$$\mathbb{P}_i[\tau_j < t] = \sum_{\ell=1}^{\infty} \mathbb{P}_i[\tau_j < t | s_j = \ell] \mathbb{P}_i[s_j = \ell] = \sum_{\ell=1}^{\infty} \mathbb{P}[\Gamma(\ell, 1) < t] \mathbb{P}_i[s_j = \ell] \tag{A.24}$$

Where $\Gamma(\ell, 1)$ denotes a Γ distribution with parameters $(\ell, 1)$. Differentiating with respect to t, we obtain,

$$f_{i,j}(x) = \sum_{\ell=1}^{\infty} \gamma_\ell(x) \mathbb{P}_i[s_j = \ell]. \tag{A.25}$$

where $\gamma_\ell(x) = \frac{x^{\ell-1}e^{-x}}{(\ell-1)!}$ denotes the density of a $\Gamma(\ell, 1)$.

Now,

$$f_{i,j}(x) = \sum_{\ell=1}^{\infty} \gamma_\ell(x) \mathbb{P}_i[s_j = \ell] = \sum_{\ell=1}^{n} \gamma_\ell(x) \mathbb{P}_i[s_j = \ell] + \sum_{\ell=n+1}^{\infty} \gamma_\ell(x) \mathbb{P}_i[s_j = \ell] \tag{A.26}$$

so that

$$\int \left| f_{i,j}(x) - \left[\sum_{\ell=1}^{n} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] + \sum_{\ell=n+1}^{\infty} \gamma_\ell(x) \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right] \right|$$

$$= \sum_{\ell=n+1}^{\infty} \left| \mathbb{P}_i \left[s_j = \ell \right] - \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right|. \quad \text{(A.27)}$$

Since

$$f_{i,j,n}(x) = \rho_{i,j,n} \left[\frac{1}{\rho_{i,j,n}} \sum_{\ell=1}^{n} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] \right] + (1 - \rho_{i,j,n}) \sum_{\ell=0}^{\infty} \gamma_{n+1+\ell}(x) \lambda_1^\ell (1 - \lambda_1) \quad \text{(A.28)}$$

$$= \sum_{\ell=1}^{n} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] + (1 - \rho_{i,j,n}) \sum_{\ell=0}^{\infty} \gamma_{n+1+\ell}(x) \lambda_1^\ell (1 - \lambda_1) \quad \text{(A.29)}$$

where $\rho_{i,j,n} = \sum_{\ell=1}^{n} \mathbb{P}_i \left[s_j = \ell \right]$,

$$|f_{i,j}(x) - f_{i,j,n}(x)| \leq \left| f_{i,j}(x) - \left[\sum_{\ell=1}^{n} \gamma_\ell(x) \mathbb{P}_i \left[s_j = \ell \right] + \sum_{\ell=n+1}^{\infty} \gamma_\ell(x) \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right] \right|$$

$$+ \left| \sum_{\ell=n+1}^{\infty} \gamma_\ell(x) \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} - (1 - \rho_{i,j,n}) \sum_{\ell=0}^{\infty} \gamma_{n+1+\ell}(x) \lambda_1^\ell (1 - \lambda_1) \right| \quad \text{(A.30)}$$

$$\Rightarrow \int |f_{i,j}(x) - f_{i,j,n}(x)| \leq \sum_{\ell=n+1}^{\infty} \left| \mathbb{P}_i \left[s_j = \ell \right] - \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right|$$

$$+ \sum_{\ell=n+1}^{\infty} \int \gamma_\ell(x) \left| \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} - (1 - \rho_{i,j,n}) \lambda_1^{\ell-n-1} (1 - \lambda_1) \right| \quad \text{(A.31)}$$

$$= \sum_{\ell=n+1}^{\infty} \left| \mathbb{P}_i \left[s_j = \ell \right] - \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right|$$

$$+ \sum_{\ell=n+1}^{\infty} \lambda_1^{\ell-1} \left| \frac{f_1(i) f_1(j) \sigma(j)}{N_{i,j}} - (1 - \rho_{i,j,n}) \lambda_1^{-n} (1 - \lambda_1) \right| \quad \text{(A.32)}$$

$$= \sum_{\ell=n+1}^{\infty} \left| \mathbb{P}_i \left[s_j = \ell \right] - \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right|$$

$$+ \frac{\lambda_1^n}{1 - \lambda_1} \left| \frac{f_1(i) f_1(j) \sigma(j)}{N_{i,j}} - (1 - \rho_{i,j,n}) \lambda_1^{-n} (1 - \lambda_1) \right| \quad \text{(A.33)}$$

$$= \sum_{\ell=n+1}^{\infty} \left| \mathbb{P}_i \left[s_j = \ell \right] - \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right| + \left| \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^n}{N_{i,j}(1 - \lambda_1)} - (1 - \rho_{i,j,n}) \right| \quad \text{(A.34)}$$

$$\leq 2 \sum_{\ell=n+1}^{\infty} \left| \mathbb{P}_i \left[s_j = \ell \right] - \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right|. \quad \text{(A.35)}$$

So,

$$||f_{i,j} - f_{i,j,n}||_{TV} \leq \sum_{\ell=n+1}^{\infty} \left| \mathbb{P}_i \left[s_j = \ell \right] - \frac{f_1(i) f_1(j) \sigma(j) \lambda_1^{\ell-1}}{N_{i,j}} \right|. \quad \text{(A.36)}$$

Now,

$$\sum_{\ell=n+1}^{\infty}\left|\mathbb{P}_i\left[s_j=\ell\right]-\frac{f_1(i)f_1(j)\sigma(j)\lambda_1^{\ell-1}}{N_{i,j}}\right|$$

$$=\frac{1}{N_{i,j}}\sum_{\ell=n+1}^{\infty}\left|Q^{\ell-1}(i,j)-f_1(i)f_1(j)\sigma(j)\lambda_1^{\ell-1}\right|$$

$$\leq\frac{1}{N_{i,j}}\sum_{z=2}^{|\mathcal{C}|}\sum_{\ell=n+1}^{\infty}|f_z(i)||f_z(j)|\sigma(j)|\lambda_z|^{\ell-1}=\frac{1}{N_{i,j}}\sum_{z=2}^{|\mathcal{C}|}|f_z(i)|f_z(j)|\sigma(j)\frac{|\lambda_2|^n}{1-|\lambda_2|}$$

$$\leq\sqrt{\frac{\sigma(j)}{\sigma(i)}}\frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)} \quad \text{(A.37)}$$

so that we finally get

$$||f_{i,j}-f_{i,j,n}||_{TV}\leq\sqrt{\frac{\sigma(j)}{\sigma(i)}}\frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)}.$$

6. Let $\phi_n=*_{i=1}^{n}\phi$ be the n-fold convolution of ϕ. Since

$$f_{i,j}(x)=\sum_{\ell=1}^{\infty}\phi_\ell(x)\mathbb{P}_i\left[s_j=\ell\right],$$

$$f_{i,j,n}(x)=\rho_{i,j,n}\left[\frac{1}{\rho_{i,j,n}}\sum_{\ell=1}^{n}\phi_\ell(x)\mathbb{P}_i\left[s_j=\ell\right]\right]+(1-\rho_{i,j,n})\sum_{\ell=0}^{\infty}\phi_{n+1+\ell}(x)\lambda_1^\ell(1-\lambda_1)$$

so that by directly following the proof of #5,

$$||f_{i,j}-f_{i,j,n}||_{TV}\leq\sqrt{\frac{\sigma(j)}{\sigma(i)}}\frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)}.$$

Now, let

$$\tilde{f}_{i,j,n} = \rho_{i,j,n}\left[\frac{1}{\rho_{i,j,n}}\sum_{\ell=1}^{n}\tilde{\phi}_\ell(x)\mathbb{P}_i\left[s_j=\ell\right]\right] + (1-\rho_{i,j,n})\sum_{\ell=0}^{\infty}\tilde{\phi}_{n+1+\ell}(x)\lambda_1^\ell(1-\lambda_1)$$

where $\tilde{\phi} = *_{i=1}^{n}\tilde{\phi}$, and $\tilde{\phi}$ has

$$\left|\left|\phi - \tilde{\phi}\right|\right|_{TV} < \epsilon.$$

Then,

$$\begin{aligned}
\left|\left|f_{i,j} - \tilde{f}_{i,j,n}\right|\right|_{TV} &\leq ||f_{i,j} - f_{i,j,n}||_{TV} + \left|\left|f_{i,j,n} - \tilde{f}_{i,j,n}\right|\right|_{TV} \\
&\leq \sqrt{\frac{\sigma(j)}{\sigma(i)}}\frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)} \\
&+ \rho_{i,j,n}\left[\frac{1}{\rho_{i,j,n}}\sum_{\ell=1}^{n}\left|\left|\phi_\ell - \tilde{\phi}_\ell\right|\right|_{TV}\mathbb{P}_i\left[s_j=\ell\right]\right] \\
&+ (1-\rho_{i,j,n})\sum_{\ell=0}^{\infty}\left|\left|\phi_{n+1+\ell} - \tilde{\phi}_{n+1+\ell}\right|\right|_{TV}\lambda_1^\ell(1-\lambda_1) \quad \text{(A.38)}
\end{aligned}$$

By the coupling argument in theorem 3,

$$\left|\left|\phi_\ell - \tilde{\phi}_\ell\right|\right|_{TV} \leq 1-(1-\epsilon)^\ell$$

so that

$$\left|\left|f_{i,j} - \tilde{f}_{i,j,n}\right|\right|_{TV} \leq ||f_{i,j} - f_{i,j,n}||_{TV} + \left|\left|f_{i,j,n} - \tilde{f}_{i,j,n}\right|\right|_{TV}$$

$$\leq \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)}$$

$$+ \rho_{i,j,n} \left[\frac{1}{\rho_{i,j,n}} \sum_{\ell=1}^{n} \left[1 - (1-\epsilon)^{\ell}\right] \mathbb{P}_i \left[s_j = \ell\right] \right]$$

$$+ (1-\rho_{i,j,n}) \sum_{\ell=0}^{\infty} \left[1 - (1-\epsilon)^{n+1+\ell}\right] \lambda_1^{\ell}(1-\lambda_1) \quad \text{(A.39)}$$

$$\leq \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)} + \rho_{i,j,n} \left[1 - (1-\epsilon)^n\right] + (1-\rho_{i,j,n}) \left[1 - \frac{(1-\lambda_1)(1-\epsilon)^{n+1}}{1-(1-\epsilon)\lambda_1}\right]$$

$$= \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)} + 1 - \left[\rho_{i,j,n}(1-\epsilon)^n + (1-\rho_{i,j,n}) \frac{(1-\lambda_1)(1-\epsilon)^{n+1}}{1-(1-\epsilon)\lambda_1}\right]$$

$$= \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)} + 1 - (1-\epsilon)^n + \frac{(1+\rho_{i,j,n})\epsilon(1-\epsilon)^n}{1-(1-\epsilon)\lambda_1}.$$

after simplifying the final term in brackets.

Now, $1-(1-\epsilon)^n \leq n\epsilon$ so that

$$\left|\left|f_{i,j} - \tilde{f}_{i,j,n}\right|\right|_{TV} \leq n\epsilon + \sqrt{\frac{\sigma(j)}{\sigma(i)}} \frac{|\lambda_2|^n}{N_{i,j}(1-|\lambda_2|)} + \frac{(1+\rho_{i,j,n})\epsilon(1-\epsilon)^n}{1-(1-\epsilon)\lambda_1}.$$

7. Since

$$\mathbb{P}_i \left[s_j = \ell\right] = \frac{Q^{\ell-1}(i,j)}{N_{i,j}}$$

and

$$Q^{\ell-1}(i,j) = \sum_z f_z(i) f_z(j) \sigma(j) \lambda_z^{\ell-1}$$

$$f_{i,j}(x) = \sum_{\ell=1}^{\infty} \frac{x^{\ell-1} e^{-x}}{(\ell-1)!} \frac{1}{N_{i,j}} \sum_z f_z(i) f_z(j) \sigma(j) \lambda_z^{\ell-1}$$

$$= \frac{1}{N_{i,j}} \sum_z f_z(i) f_z(j) \sigma(j) e^{-x} \sum_{\ell=1}^{\infty} \frac{(x\lambda_z)^{\ell-1}}{(\ell-1)!} \quad \text{(A.40)}$$

$$= \frac{1}{N_{i.j}} \sum_z f_z(i) f_z(j) \sigma(j) e^{-x} e^{x\lambda_z} = \frac{1}{N_{i,j}} \sum_z f_z(i) f_z(j) \sigma(j) e^{-x(1-\lambda_z)}$$

$$= \frac{1}{N_{i,j}} \sum_z \frac{f_z(i) f_z(j) \sigma(j)}{1-\lambda_z} (1-\lambda_z) e^{-x(1-\lambda_z)}. \quad \text{(A.41)}$$

8. Let r be the number of jumps taken to transition from x to y. Then,

$$q_t(x,y) = \mathbb{P}_x \left[\mathcal{Q}_t = y\right] = \sum_{n=0}^{\infty} \mathbb{P}_x \left[\mathcal{Q}_t = y | r = n\right] \mathbb{P}\left[r = n\right]$$

$$= \sum_{n=0}^{\infty} Q^n(x,y) \mathbb{P}\left[\mathcal{J}(n) \leq t, \mathcal{J}(n+1) > t\right] \quad \text{(A.42)}$$

where $\mathcal{J}(n) \sim \Gamma(n,1)$ denotes the time of the n^{th} jump.

Recall that the probability density function of a $\Gamma(n,1)$ is $f_n(x) = \frac{x^{n-1}e^{-x}}{(n-1)!}$. Since $\mathcal{J}(n+1) = \mathcal{J}(n) + X$ where $X \sim exp(1)$,

$$\mathbb{P}\left[\mathcal{J}(n) \leq t, \mathcal{J}(n+1) > t\right] = \mathbb{P}\left[\mathcal{J}(n) \leq t, X > t - \mathcal{J}(n)\right]$$

$$= \int_0^t \left[\int_{t-x}^{\infty} e^{-s} ds\right] \frac{x^{n-1} e^{-x}}{(n-1)!} dx$$

$$= \int_0^t e^{-(t-x)} \frac{x^{n-1} e^{-x}}{(n-1)!} dx = e^{-t} \int_0^t \frac{x^{n-1}}{(n-1)!} dx = e^{-t} \frac{t^n}{n!} \quad \text{(A.43)}$$

Thus,

$$q_t(x,y) = \sum_{n=0}^{\infty} Q^n(x,y) \frac{e^{-t} t^n}{n!}. \quad \square$$

A.1.2 Theorem 2

Theorem 2. *Let $x \in \mathcal{C}_1$ and $y \in \mathcal{C}_2$ be in two clusters and $p_t(x,y) = \mathbb{P}_x\left[Z_t = y\right]$ where Z_t is a Markov process with exp(1) hold times and jump matrix P on a finite state space V with boundary vertices ∂V, and $Q_i = P_{\mathcal{C}_i, \mathcal{C}_i}$.*

Let $q_t^{(i)}$ be $|\mathcal{C}_i| \times |\mathcal{C}_i|$ matrices defined by

$$q_t^{(i)}(i,j) = \sum_{n=0}^{\infty} Q_i^n(i,j) \frac{e^{-t} t^n}{n!},$$

r_t be a $|\mathcal{C}_1| \times |\partial V|$ matrix with entries

$$r_t(x,a) = \sum_{n=0}^{\infty} (Q_1^n P)(x,a) \frac{e^{-t} t^n}{n!}$$

and s_t be a $|\partial V| \times |\partial V|$ matrix with entries:

$$s_t(\omega_0, \omega_n) = \sum_{n=1}^{\infty} \sum_{|\omega|=n} \mathbb{P}_{\omega_0}\left[Z_{T_1} = \omega_1\right] \cdots \mathbb{P}_{\omega_{n-1}}\left[Z_{T_n} = \omega_n\right] dT_n(t)(\omega)$$

where $dT_n(x)(\omega)$ is the density of T_n, the time of the n^{th} jump on the cluster-cluster process defined pathwise, and

$$\mathbb{P}_{\omega_{i-1}}\left[Z_{T_i} = \omega_i\right] = \sum_{j \in \partial \mathcal{C}_{\omega_{i-1}}} N_{\omega_{i-1},j} P(j, \omega_i)$$

where $N = (I - Q)^{-1}$ and $Q = P|_{\mathcal{C},\mathcal{C}}$ for a cluster $\mathcal{C}$.

Then,

$$p_t(x, y) = q_t^{(1)}(x, y) + (rsq^{(2)})(t, x, y) \tag{A.44}$$

where matrix products are convolutions.

Proof of Theorem 2:

Given $x \in \mathcal{C}_1$ and $y \in \mathcal{C}_2$, let R_1 be the first exit time from $\mathcal{C}_1$. Then,

$$\mathbb{P}_x\left[Z_t = y\right] = \mathbb{P}_x\left[Z_t = y, R_1 > t\right] + \mathbb{P}_x\left[Z_t = y, R_1 \leq t\right].$$

In the first term, the process is in $\mathcal{C}_1$ for all times $s \leq t$ so that

$$\mathbb{P}_x\left[Z_t = y, R_1 > t\right] = q_t^{(1)}(x, y) = \sum_{n=0}^{\infty} Q_1^n(x, y) \frac{e^{-t} t^n}{n!}.$$

For the second term,

$$\mathbb{P}_x\left[Z_t = y, R_1 \leq t\right] = \sum_a \mathbb{P}_x\left[Z_{R_1+(t-R_1)} = y, R_1 \leq t, Z_{R_1} = a\right]$$

$$= \sum_a \mathbb{P}_x\left[Z_{R_1+(t-R_1)} = y | R_1 \leq t, Z_{R_1} = a\right] \mathbb{P}_x\left[R_1 \leq t | Z_{R_1} = a\right] \mathbb{P}_x\left[Z_{R_1} = a\right] \tag{A.45}$$

Now, $\mathbb{P}_x\left[Z_{R_1} = a\right] = \sum_{k \in \partial \mathcal{C}_1} N_{x,k} P(k, a)$ and $\frac{r_t(x,a)}{\mathbb{P}_x\left[Z_{R_1} = a\right]}$ is pdf of the first exit time starting at x leaving to a where $r_t(x, a) = \sum_{n=0}^{\infty} (Q_1^n P)(x, a) \frac{e^{-t} t^n}{n!}$.

Using the strong Markov property then, we get:

$$\mathbb{P}_x\left[Z_t = y, R_1 \leq t\right] = \sum_a \int_0^t r_{t_1}(x,a)\mathbb{P}_a\left[Z_{t-t_1} = y\right] dt_1.$$

Now,

$$\mathbb{P}_a\left[Z_{t-t_1} = y\right] = \sum_{n=1}^{\infty} \sum_{|\omega|=n} \mathbb{P}_a\left[Z_{t-t_1} = y, \omega | T_n \leq t - t_1\right] \mathbb{P}_a\left[T_n \leq t - t_1\right]$$

where n is the number of cluster-cluster jumps necessary to reach $\mathcal{C}_2$ and hit y and ω is such a cluster-cluster path and T_n are the times of these jumps. Repeated use of the Markov property implies

$$\mathbb{P}_a\left[Z_{t-t_1}, \omega | T_n \leq t - t_1\right] = \mathbb{P}_a\left[Z_{T_1} = \omega_1\right] \cdots \mathbb{P}_{\omega_{n-1}}\left[Z_{T_n} = \omega_n\right] \mathbb{P}_{\omega_n}\left[\mathcal{Q}_{t-t_1-T_n} = y | T_n \leq t - t_1\right]$$

where $\mathcal{Q}_t$ is the process Z_t restricted to $\mathcal{C}_2$ so that

$$\mathbb{P}_a\left[Z_{t-t_1} = y\right] = \sum_{n=1}^{\infty} \sum_{|\omega|=n} \mathbb{P}_a\left[Z_{T_1} = \omega_1\right] \cdots \mathbb{P}_{\omega_{n-1}}\left[Z_{T_n} = \omega_n\right] \mathbb{P}_{\omega_n}\left[\mathcal{Q}_{t-t_1-T_n} = y, T_n \leq t - t_1\right]$$

$$= \sum_{\omega_n} \int_0^{t-t_1} s_{t_2}(a,\omega_n) q^{(2)}_{t-t_1-t_2}(\omega_n, y) dt_2$$

where

$$s_{t_2}(a,\omega_n) = \sum_{n=1}^{\infty} \sum_{|\omega|=n+1} \mathbb{P}_a\left[Z_{T_1} = \omega_1\right] \cdots \mathbb{P}_{\omega_{n-1}}\left[Z_{T_n} = \omega_n\right] dT_n(t_2)(\omega)$$

So,

$$\mathbb{P}_x\left[Z_t = y\right] = q_t^{(1)}(x,y) + \sum_{a,b}\int_0^t r_{t_1}(x,a)\left[\int_0^{t-t_1} s_{t_2}(a,b)q_{t-t_1-t_2}^{(2)}(b,y)dt_2\right]dt_1$$

$$= q_t^{(1)}(x,y) + \left(rsq^{(2)}\right)(t,x,y)$$

where r, s, q are matrices whose entries are functions and whose matrix product are convolutions. $\square$

A.1.3 Theorem 3

First, we recall the following definition.

Definition 1. *A continuous-time stochastic process is called a semi-Markov process or "Markov renewal process" if the embedded jump chain (the discrete process registering what values the process takes) is a Markov chain, and where the holding times (time between jumps) are random variables with any distribution, whose distribution function may depend on the two states between which the move is made.*

Theorem 3. *Let X_t and $\tilde{X}_t$ be two semi-Markov processes on a finite state space V with the same transition matrix P that is irreducible and aperiodic. Let $p_t(x,\cdot)$ and $\tilde{p}_t(x,\cdot)$ be the laws of X_t and $\tilde{X}_t$, respectively, starting at $x \in V$. Suppose that the hold times of X_t and $\tilde{X}_t$ agree in total variation distance within ϵ. Then, there exists constants $C > 0$ and $\alpha \in (0,1)$ computable completely in terms of P such that*

$$\max_x ||p_t(x,\cdot) - \tilde{p}_t(x,\cdot)||_{TV} \leq \epsilon\left(\frac{1}{2} + C\frac{1+\alpha}{(1-\alpha)(1-(1-\epsilon)\alpha)}\right).$$

Proof of Theorem 3:

Let $J_{x,y}(t)$ be the number of jumps by time t the process takes from x to y. Then,

$$p_t(x,y) = \mathbb{P}_x\left[X_t = y\right] = \sum_{n=0}^{\infty} P^n(x,y)\mathbb{P}\left[J_{x,y}(t) = n\right]$$

$$= \sum_{n=0}^{\infty} P^n(x,y) \int_0^t \left[\int_{t-x}^{\infty} f(z)dz \right] g_n(w)dw$$

where

$$f(z) = \sum_{\omega_{n+1}} P(y,\omega_{n+1}) f_{y,\omega_{n+1}}(z) \quad \text{and} \quad g_n(w) = \frac{1}{P^n(x,y)} \sum_{\omega(0)=x,\omega(n)=y} P^n[\omega] f_{n,y,\omega}(w) \tag{A.46}$$

are the densities of the jump time from y and the time from x to y in n jumps, given by the random variable, $\mathcal{J}_{x,y}(n)$.

Lemma 2. *If* $\mathbb{P}[A \cup B^c] = 1$ *then* $\mathbb{P}[A \cap B^c] = \mathbb{P}[A] - \mathbb{P}[B]$.

Proof of Lemma 2: By inclusion-exclusion:

$$\mathbb{P}[A \cup B^c] = \mathbb{P}[A] + \mathbb{P}[B^c] - \mathbb{P}[A \cap B^c],$$

so that

$$\mathbb{P}[A \cap B^c] = \mathbb{P}[A] + 1 - \mathbb{P}[B] - \mathbb{P}[A \cup B^c] = \mathbb{P}[A] - \mathbb{P}[B]. \quad \triangle$$

Apply the Lemma 2 to $A = \{\mathcal{J}_{x,y}(n) \leq t\}$ and $B = \{\mathcal{J}_{x,y}(n+1) \leq t\}$ then $A^c \subseteq B^c$, so that $A \cup B^c = \Omega$, so that

$$\mathbb{P}[J_{x,y}(t) = n] = \mathbb{P}[\mathcal{J}_{x,y}(n) \leq t, \mathcal{J}_{x,y}(n+1) > t] = \mathbb{P}[\mathcal{J}_{x,y}(n) \leq t] - \mathbb{P}[\mathcal{J}_{x,y}(n+1) \leq t].$$

The same holds for the process $\tilde{X}_t$ so that

$$p_t(x,y) - \tilde{p}_t(x,y) = \sum_{n=0}^{\infty} P^n(x,y) \left(\mathbb{P}\left[J_{x,y}(t) = n\right] - \mathbb{P}\left[\tilde{J}_{x,y}(t) = n\right] \right)$$

$$= \sum_{n=0}^{\infty} P^n(x,y) \cdot$$

$$\left[\mathbb{P}\left[\mathcal{J}_{x,y}(n) \leq t\right] - \mathbb{P}\left[\mathcal{J}_{x,y}(n+1) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n) \leq t\right] + \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n+1) \leq t\right] \right]$$

$$= \sum_{n=1}^{\infty} \left(P^n(x,y) - P^{n-1}(x,y) \right) \left(\mathbb{P}\left[\mathcal{J}_{x,y}(n) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n) \leq t\right] \right)$$

$$+ \delta_{x,y} \left(\mathbb{P}\left[\mathcal{J}_{x,y}(1) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(1) \leq t\right] \right). \quad \text{(A.47)}$$

To bound this quantity, we first take care of the terms of the form

$$\left| \mathbb{P}\left[\mathcal{J}_{x,y}(n) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n) \leq t\right] \right|.$$

Consider a jump path ω of length n starting at x and ending at y, with jumps through $(i_0, \ldots, i_n)$, and let $\tau_{X,j}$ be the random variable corresponding to the time taken to go from i_{j-1} to i_j for X_t and similarly for $\tau_{\tilde{X},j}$ and $\tilde{X}_t$. If $\mathcal{J}_X(n,\omega)$ and $\mathcal{J}_{\tilde{X}}(n,\omega)$ are the random variables with densities $f_{y,n,\omega}$ and $\tilde{f}_{y,n,\omega}$ as in (A.46), then

$$\mathcal{J}_X(n,\omega) = \sum_{j=1}^{n} \tau_{X,j}$$

and

$$\mathcal{J}_{\tilde{X}}(n,\omega) = \sum_{j=1}^{n} \tau_{\tilde{X},j}.$$

Now, let $\mu_j \left[\tau_{X,j} = \cdot, \tau_{\tilde{X}_j} = \cdot\right]$ be the optimal coupling for $(\tau_{X,j}, \tau_{\tilde{X},j})$ for $j = 1, \ldots, n$ and let μ be the product measure of these optimal couplings. Since μ is a

product measure, and μ_j is each an optimal coupling, μ is a coupling of $\{\tau_{X,j}, \tau_{\tilde{X},j}\}_{j=1}^n$. Since $\{\tau_{\tilde{X},j}\}_{j=1}^n$ are independent and $\{\tau_{X,j}\}_{j=1}^n$ are independent, it's clear that this is also a coupling of $\mathcal{J}_X(n,\omega)$ and $\mathcal{J}_{\tilde{X}}(n,\omega)$.

Now,

$$\bigcap_{j=1}^{n}\{\tau_{X,j} = \tau_{\tilde{X},j}\} \subseteq \left\{\sum_{j=1}^{n}\left(\tau_{X,j} - \tau_{\tilde{X},j}\right) = 0\right\}$$

so that

$$\begin{aligned}
||\mathcal{J}_X(n,\omega) - \mathcal{J}_{\tilde{X}}(n,\omega)||_{TV} &\leq \mu\left[\mathcal{J}_X(n,\omega) \neq \mathcal{J}_{\tilde{X}}(n,\omega)\right] \\
&= \mu\left\{\sum_{j=1}^{n}\tau_{X,j} \neq \sum_{j=1}^{n}\tau_{\tilde{X},j}\right\} = \mu\left\{\sum_{j=1}^{n}\left(\tau_{X,j} - \tau_{\tilde{X},j}\right) \neq 0\right\} \\
&= 1 - \mu\left[\sum_{j=1}^{n}\left(\tau_{X,j} - \tau_{\tilde{X},j}\right) = 0\right] \leq 1 - \mu\left[\bigcap_{j=1}^{n}\{\tau_{X,j} = \tau_{\tilde{X},j}\}\right] \\
&= 1 - \prod_{j=1}^{n}\left(1 - ||\tau_{X,j} - \tau_{\tilde{X},j}||_{TV}\right) \quad \text{(A.48)}
\end{aligned}$$

Since $||\tau_{X,j} - \tau_{\tilde{X},j}||_{TV} \leq \epsilon$,

$$||\mathcal{J}_X(n,\omega) - \mathcal{J}_{\tilde{X}}(n,\omega)||_{TV} \leq 1 - (1-\epsilon)^n$$

.

Now, notice that $||\mathcal{J}_X(n,\omega) - \mathcal{J}_{\tilde{X}}(n,\omega)||_{TV} = ||f_{n,y,\omega} - \tilde{f}_{n,y,\omega}||_{TV}$ as in (A.46) so that

$$\left|\mathbb{P}\left[\mathcal{J}_{x,y}(n) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n) \leq t\right]\right| \leq ||\mathcal{J}_{x,y}(n) - \tilde{\mathcal{J}}_{x,y}(n)|_{TV} = ||g_n - \tilde{g}_n||_{TV} \leq 1-(1-\epsilon)^n.$$

This takes care of the temporal differences in (A.47). For the differences in the jump chain, we use the convergence theorem for Markov chains:

Proposition 7. *Suppose that P is irreducible and aperiodic with stationary distribution π. Then, there exists $\alpha \in (0,1)$ and $C > 0$ such that*

$$d(t) = \max_x ||P^t(x,\cdot) - \pi||_{TV} \leq C\alpha^t.$$

For the proof, see Levin and Peres (2009).

Let π denote the stationary distribution for P, and C and α be any two such constants as in (7). Then, (A.47) becomes

$$\begin{aligned}
|p_t(x,y) - \tilde{p}_t(x,y)| &\leq \delta_{x,y} \left| \mathbb{P}\left[\mathcal{J}_{x,y}(1) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(1) \leq t\right] \right| \\
&\quad + \sum_{n=1}^{\infty} \left| P^n(x,y) - P^{n-1}(x,y) \right| \left| \mathbb{P}\left[\mathcal{J}_{x,y}(n) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n) \leq t\right] \right| \\
&\leq \epsilon\delta_{x,y} + \sum_{n=1}^{\infty} |P^n(x,y) - \pi(y)| + \left| P^{n-1}(x,y) - \pi(y) \right| \left| \mathbb{P}\left[\mathcal{J}_{x,y}(n) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n) \leq t\right] \right| \\
&\leq \epsilon\delta_{x,y} + \sum_{n=1}^{\infty} |P^n(x,y) - \pi(y)| + \\
&\quad \left| P^{n-1}(x,y) - \pi(y) \right| \max_y \left| \mathbb{P}\left[\mathcal{J}_{x,y}(n) \leq t\right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n) \leq t\right] \right| \quad \text{(A.49)}
\end{aligned}$$

$$\Rightarrow ||p_t(x,\cdot) - \tilde{p}_t(x,\cdot)||_{TV} \leq \frac{\epsilon}{2} + \sum_{n=1}^{\infty} \left(||P^n(x,\cdot) - \pi||_{TV} + \left|\left| P^{n-1}(x,\cdot) - \pi \right|\right|_{TV} \right) \cdot$$

$$\max_y \left| \mathbb{P}\left[\mathcal{J}_{x,y}(n) \leq t \right] - \mathbb{P}\left[\tilde{\mathcal{J}}_{x,y}(n) \leq t \right] \right|$$

$$\leq \frac{\epsilon}{2} + \sum_{n=1}^{\infty} \left(C\alpha^n + C\alpha^{n-1} \right) \left(1 - (1-\epsilon)^n \right) = \frac{\epsilon}{2} + C(1+\alpha) \sum_{n=1}^{\infty} \alpha^{n-1} \left(1 - (1-\epsilon)^n \right)$$

$$= \epsilon \left(\frac{1}{2} + C \frac{1+\alpha}{(1-\alpha)(1-(1-\epsilon)\alpha)} \right). \quad \square \quad \text{(A.50)}$$

A.1.4 Theorem 4

Theorem 4. *Let X_t and $\tilde{X}_t$ be two semi-Markov processes on a finite state space V with the same transition matrix P that is irreducible and aperiodic. Let $\mathcal{C} \subseteq V$ be a subset of the state space on which the jump chain of X_t and $\tilde{X}_t$ are irreducible and aperiodic.*

Define $q_t(x,y)$ and $\tilde{q}_t(x,y)$ to be the laws of the process X_t and $\tilde{X}_t$ restricted to $x, y \in \mathcal{C}$. We can turn q_t and $\tilde{q}_t$ into probability measures, p_t and $\tilde{p}_t$, by closing the system $\mathcal{C}$ by a coffin state ∂ representing when the processes X_t and $\tilde{X}_t$ leaving $\mathcal{C}$. Suppose that the hold times of X_t and $\tilde{X}_t$ agree in total variation distance within ϵ. Then, there exists $C > 0$ and $\alpha \in (0,1)$ computable complete in terms of $Q = P|_{\mathcal{C},\mathcal{C}}$ such that

$$\max_x ||p_t(x,\cdot) - \tilde{p}_t(x,\cdot)||_{TV} \leq \epsilon \left(\frac{1}{2} + C \frac{1+\alpha}{(1-\alpha)(1-(1-\epsilon)\alpha)} \right).$$

Proof of Theorem 4: We can repeat the same argument as in Theorem 3, replacing Proposition 7 with the following analogue.

Proposition 8. *Let P be the transition matrix for a time-homogeneous Markov chain*

that is irreducible and aperiodic on the state space V. Let $\mathcal{C} \subseteq V$ on which P is also irreducible and aperiodic. Define $Q = P|_{\mathcal{C},\mathcal{C}}$.

Let $\tilde{V} = \mathcal{C} \cup \{\partial\}$ where ∂ is a coffin state representing when the chain leaves $\mathcal{C}$. We can define a new Markov chain on $\tilde{V}$ by

$$\tilde{P} = \begin{bmatrix} Q & S \\ 0 \cdots & 1 \end{bmatrix} \tag{A.51}$$

where $S(i, |\mathcal{C}| + 1) = \sum_{j \in \mathcal{C}^c} P(i, j)$.

Since there are no other absorbing states, it's clear that $\pi = [0, \cdots, 1]$ is the stationary distribution for $\tilde{P}$. Then, there exists $\alpha \in (0, 1)$ and $C > 0$ such that

$$d(t) = \max_x ||\tilde{P}^t(x, \cdot) - \pi||_{TV} \leq C\alpha^t.$$

The proof of this proposition uses the following lemma:

Lemma 3. *In the same notation as proposition 8, let τ be the random variable of the exit time from $\mathcal{C}$. Then,*

$$\mathbb{P}_x [\tau = n] = \left(Q^{n-1} S\right)(x) \quad \text{for } n = 1, \ldots.$$

Now, given $\delta > 0$, there exists n such that

$$\sum_{m=0}^{n-1} \left(Q^m S\right)(x) \geq \delta$$

for all $x \in \mathcal{C}$. Thus,

$$M^n = \delta \Pi + (1 - \delta) T$$

defines a stochastic matrix T. The proof of proposition 8 now follows immediately from the proof of proposition 7. $\square$

A.2 Chapter 3

A.2.1 Theorem 5

In this section, we prove theorem 5 regarding properties of P_{E1}.

Theorem 5. *Let P be defined by $P(x,y) = \frac{1}{2}\sum_w (\pi_{x,E}Q)(e_{w,y})$.*

1. *Define $\pi_{VE}(x) = \frac{1}{2}\sum_z \pi_E(e_{x,z})$. Then, π_{VE} is a stationary measure for P and $\mu P \to \pi_{VE}$ as $t \to \infty$ for any probability measure μ on V.*

2. *$\pi_{VE}(x)P(x,y) = \pi_{VE}(y)P(y,x)$.*

Proof of Theorem 5:

1. $(\pi_{VE}P)(y) = \sum_x \pi_{VE}(x)P(x,y) = \frac{1}{4}\sum_{x,w}\left(\sum_z \pi_E(e_{x,z})\right)(\pi_{x,E}Q)(e_{w,y})$

   $= \frac{1}{4}\sum_{x,w,q}\left(\sum_z \pi_E(e_{x,z})\right)\frac{\pi_E(e_{x,q})}{\sum_z \pi_E(e_{x,z})}Q(e_{x,q},e_{w,y}) = \frac{1}{4}\sum_{x,w,q}\pi_E(e_{x,q})Q(e_{x,q},e_{w,y})$

   $= \frac{1}{2}\sum_w \pi_E(e_{w,y}) = \pi_{VE}(y)$.

   Thus, π_{VE} is a stationary distribution of P.

   Now, without loss of generality, let's restrict to a connected component of P. On this component, P is irreducible, and has a stationary distribution π_{VE} restricted to this component. It suffices to show that on this component, P is aperiodic, so that by the ergodic theorem, $\mu P \to \pi_{VE}$ as $t \to \infty$.

   To see this, we need to show that $P^t(x,x) > 0$ for any t. This is because starting from x, we can choose an edge connected to x by $\pi_{x,E}$, say $e_{x,y}$, and then transition to $e_{x,z}$ through one iteration of Q. Since the definition of P sums through all edges connected to x, we conclude that $P(x,x) > 0$. This immediately implies $P^t(x,x) > 0$ for any t.

2. Since Q is reversible with respect to π_E:

$$\begin{aligned}
\pi_{VE}(x)P(x,y) &= \frac{1}{4}\left(\sum_z \pi_E(e_{x,z})\right)\sum_w (\pi_{x,E}Q)(e_{w,y}) \\
&= \frac{1}{4}\left(\sum_z \pi_E(e_{x,z})\right)\sum_{w,q} \frac{\pi_E(e_{x,q})}{(\sum_z \pi_E(e_{x,z}))} Q(e_{x,q}, e_{w,y}) \\
&= \frac{1}{4}\left(\sum_z \pi_E(e_{y,z})\right)\sum_{w,q} \frac{\pi_E(e_{y,w})}{(\sum_z \pi_E(e_{y,z}))} Q(e_{y,w}, e_{q,x}) \\
&= \frac{1}{4}\left(\sum_z \pi_E(e_{y,z})\right)\sum_q (\pi_{y,E}Q)(e_{q,x}) \\
&= \pi_{VE}(y)P(y,x). \quad \square \quad \text{(A.52)}
\end{aligned}$$

A.2.2 Theorem 6

First, we require the following proposition [Pinelis (1991), Pinelis (1994)].

Proposition 9. (Pinelis' Inequalities)

Let Y_i, $i = 1, \ldots, n$ be a sequence of n zero mean random variables with values in a Hilbert space $(H, \langle \cdot, \cdot \rangle_H)$ satisfying

$$\mathbb{E}\left[||Y_i||_H^p\right] \leq \frac{1}{2}p!b^2L^{p-2}, \;\; p \geq 2,$$

for two positive constants b and L. Then, for all $t > 0$,

$$\mathbb{P}\left[\left|\left|\frac{1}{n}\sum_{i=1}^n Y_i\right|\right|_H \leq \frac{Lt^2}{n} + \frac{\sqrt{2}bt}{\sqrt{n}}\right] \geq 1 - 2e^{-t^2}.$$

If $||Y_i|| \leq M$ for all $i = 1 \ldots, n$ then,

$$\mathbb{P}\left[\left|\left|\frac{1}{n}\sum_{i=1}^n Y_i\right|\right|_H \leq \frac{\sqrt{2}Mt}{\sqrt{n}}\right] \geq 1 - 2e^{-t^2}.$$

Finally, we prove theorem 6.

Theorem 6. *Let $p_n = (p^{(n)}_{i_1,\ldots,i_m})$ be the empirical joint distribution vector for m random variables, $X_1, \ldots, X_m$ obtained from n samples, and let $p = (p_{i_1,\ldots,i_m})$ be the actual joint distribution. More precisely,*

$$p_{i_1,\ldots,i_m} = \mathbb{P}\left[X_1 = i_1, \ldots, X_m = i_m\right]$$

where $i_k = 1, \ldots, N_{X_k}$, and $p_n = \frac{1}{n}\sum_{k=1}^{n} \mathbf{X_k}$ are the empirical probabilities obtained from n samples, where $\mathbf{X_k}$ are independent multivariate Bernoulli random variables with probabilities $p_{i_1,\ldots,i_m}$. Furthermore, define the total variation distance between probability mass functions p and q by

$$||p - q||_{TV} = \frac{1}{2}\sum_{i=1}^{N} |p(i) - q(i)|.$$

Then,

$$\mathbb{P}\left[||p_n - p||_{TV} > \epsilon\right] \leq 2e^{-\frac{n\epsilon^2}{2\prod_{k=1}^{m} N_{X_k}}}.$$

Proof of Theorem 6:

$$||p_n - p||_{TV} = \frac{1}{2}\sum_{i_1,\ldots,i_m} |p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}| = \frac{1}{2}\left|\left|\frac{1}{n}\sum_{k=1}^{n}\mathbf{X_k} - p\right|\right|_1 = \frac{1}{2}\left|\left|\frac{1}{n}\sum_{k=1}^{n}(\mathbf{X_k} - p)\right|\right|_1$$
$$\leq \frac{1}{2}\left|\left|\frac{1}{n}\sum_{k=1}^{n}(\mathbf{X_k} - p)\right|\right|_2 \sqrt{\prod_{k=1}^{m} N_{X_k}} \quad \text{(A.53)}$$

where the last inequality follows by Cauchy-Schwartz inequality. Now, notice that

$$||\mathbf{X_k} - p||_2 \leq ||\mathbf{X_k}||_2 + ||p||_2 \leq 2$$

so that by Pinelis' inequality:

$$\mathbb{P}\left[||p_n - p||_{TV} > \epsilon\right] \leq \mathbb{P}\left[\left|\left|\frac{1}{n}\sum_{k=1}^{n}\mathbf{X_k} - \mathbf{X}\right|\right|_2 > \frac{2\epsilon}{\sqrt{\prod_{k=1}^{m} N_{X_k}}}\right] \leq 2e^{-\frac{n\epsilon^2}{2\prod_{k=1}^{m} N_{X_k}}}. \quad \square$$

A.2.3 Theorem 7

In this section, we prove theorem 7 regarding concentration of mutual information.

Theorem 7. *Given the same notation in Theorem 6, define*

$$H_n(X_1, \ldots, X_m) = \sum_{i_1,\ldots,i_m} p^{(n)}_{i_1,\ldots,i_m} \log \frac{p^{(n)}_{i_1,\ldots,i_m}}{\prod_{k=1}^{m} p^{(n)}_{X_k}(i_k)}$$

and

$$H(X_1, \ldots, X_m) = \sum_{i_1,\ldots,i_m} p_{i_1,\ldots,i_m} \log \frac{p_{i_1,\ldots,i_m}}{\prod_{k=1}^{m} p_{X_k}(i_k)}$$

to be the m-wise Shannon entropy of $X_1, \ldots, X_m$, where p_{X_k} and $p^{(n)}_{X_k}$ denote the k^{th} marginal distributions of p and p_n, respectively. Let

$$Z^{(n)}_{i_1,\ldots,i_m} = \sum_{k=1}^{n} (\mathbf{X_k})_{i_1,\ldots,i_m}$$

be the multinomial distribution with parameters $(n, p_{i_1,\ldots,i_m})$ giving the number of the n samples that equal the m-tuple, $(i_1, \ldots, i_m)$. Then, if $p_{i_1,\ldots,i_m} \neq 0$, $e \leq \mathbb{E}\left[Z^{(n)}_{i_1,\ldots,i_m}\right]$ and $e^2 \leq n$,

$$\mathbb{P}\left[|H_n(X_1, \ldots, X_m) - H(X_1, \ldots, X_m)| > \epsilon\right] \leq e^{-8\frac{n}{(\log n - 1)^2} \cdot \frac{\epsilon^2}{(m+1)^2 \prod_{k=1}^{m} N_{X_k}}}.$$

Proof of Theorem 7:

To minimize the notation, let $H_n = H_n(X_1, \ldots, X_m)$ and $H = H(X_1, \ldots, X_m)$, and

$$p \log p = \sum_{i_1, \ldots, i_m} p_{i_1, \ldots, i_m} \log p_{i_1, \ldots, i_m}$$

for vectors $p = (p_{i_1, \ldots, i_m})$. Then,

$$\mathbb{P}\left[|H_n - H| > \epsilon\right] = \mathbb{P}\left[\left| p \log p - p_n \log p_n + \sum_{k=1}^{m} p_{X_k} \log p_{X_k} - p_{X_k}^{(n)} \log p_{X_k}^{(n)} \right| > \epsilon \right]$$

$$\leq \mathbb{P}\left[|p \log p - p_n \log p_n| + \sum_{k=1}^{m} \left| p_{X_k} \log p_{X_k} - p_{X_k}^{(n)} \log p_{X_k}^{(n)} \right| > \epsilon \right] \tag{A.54}$$

Now, the function $f(x) = x \log x$ is continuous on $[0, 1]$ and locally Lipschitz on $(0, 1)$, so that by the mean value theorem,

$$|f(x) - f(y)| \leq |f'(a)||x - y|, \quad \text{for} \quad x, y \in [0, 1]$$

for some $a \in (0, 1)$, and where $f'(a) = 1 + \log a$.

Let's consider the first term under the probability sign in (A.54).

$$|p \log p - p_n \log p_n| \leq \sum_{i_1, \ldots, i_m} \left| p_{i_1, \ldots, i_m} \log p_{i_1, \ldots, i_m} - p_{i_1, \ldots, i_m}^{(n)} \log p_{i_1, \ldots, i_m}^{(n)} \right|.$$

We have the following cases:

1. If $p_{i_1, \ldots, i_m}^{(n)}, p_{i_1, \ldots, i_m} \in [e^{-2}, 1]$ then this $|f'(a)| \leq 1$, so that

$$|p_{i_1, \ldots, i_m}^{(n)} \log p_{i_1, \ldots, i_m}^{(n)} - p_{i_1, \ldots, i_m} \log p_{i_1, \ldots, i_m}| \leq |p_{i_1, \ldots, i_m}^{(n)} - p_{i_1, \ldots, i_m}|. \tag{A.55}$$

2. If, on the other hand, either $p^{(n)}_{i_1,\ldots,i_m}, p_{i_1,\ldots,i_m} \in (0, e^{-2})$, then

$$|p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m}| \leq [-1 - \log(a)]|p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}|.$$

Now, we can make $-\log(a)$ bigger by replacing a by the smaller of $p_{i_1,\ldots,i_m}$ and $p^{(n)}_{i_1,\ldots,i_m}$, so that

$$|p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m}|$$

$$\leq [-1 - \log(\min\{p_{i_1,\ldots,i_m}, p^{(n)}_{i_1,\ldots,i_m}\})]|p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}|$$

Finally, notice that in this case, $p^{(n)}_{i_1,\ldots,i_m} \geq \frac{1}{n}$, so that

$$|p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m}| \leq [-1 - \log c_n]\, |p_{i_1,\ldots,i_m} - p^{(n)}_{i_1,\ldots,i_m}| \tag{A.56}$$

where $c_n = \min\{\frac{1}{n}, p_{i_1,\ldots,i_m} : p_{i_1,\ldots,i_m} \neq 0\}$.

3. Finally, if $p^{(n)}_{i_1,\ldots,i_m} = 0$, since $p_{i_1,\ldots,i_m} \neq 0$, we have

$$|p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m} - p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m}|$$
$$= -p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m} \leq [-1 - \log c_n]]p_{i_1,\ldots,i_m} \Leftrightarrow p_{i_1,\ldots,i_m} \geq e \cdot c_n. \tag{A.57}$$

We can ensure $p_{i_1,\ldots,i_m} \geq e \cdot c_n$ by taking $c_n = \frac{1}{n}$, which will happen when $1 \leq np_{i_1,\ldots,i_m}$, so that

$$|p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m} - p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m}| \leq [\log n - 1] \left| p_{i_1,\ldots,i_m} - p^{(n)}_{i_1,\ldots,i_m} \right| \tag{A.58}$$

when $e \leq np_{i_1,\ldots,i_m} = \mathbb{E}\left[Z^{(n)}_{i_1,\ldots,i_m}\right]$.

Combining (A.56) and (A.58), we get

$$|p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m}| \leq [\log n - 1]\, |p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}|$$

when $e \leq \mathbb{E}\left[Z^{(n)}_{i_1,\ldots,i_m}\right]$. We can also include (A.55) here when $e^2 \leq n$. To see this, notice that when $c_n = \frac{1}{n}$,

$$1 \leq -1 - \log c_n = \log n - 1 \Leftrightarrow e^2 \leq n$$

so that when $e^2 \leq n$,

$$|p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m}| \leq |p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}| \leq [\log n - 1]\, |p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}|.$$

So, in general, we have,

$$|p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m}| \leq [\log n - 1]\, |p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}|$$

when $e \leq \mathbb{E}\left[Z^{(n)}_{i_1,\ldots,i_m}\right]$ and $e^2 \leq n$, so that,

$$\begin{aligned} |p \log p - p_n \log p_n| &\leq \sum_{i_1,\ldots,i_m} |p^{(n)}_{i_1,\ldots,i_m} \log p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m} \log p_{i_1,\ldots,i_m}| \\ &\leq [\log n - 1] \sum_{i_1,\ldots,i_m} |p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}| \quad \text{(A.59)} \end{aligned}$$

when $e \leq \mathbb{E}\left[Z^{(n)}_{i_1,\ldots,i_m}\right]$ and $e^2 \leq n$.

The same argument holds for each term in the summand of (A.54),

$$\left|p_{X_k}\log p_{X_k} - p^{(n)}_{X_k}\log p^{(n)}_{X_k}\right| \leq \sum_{i_k}\left|p_{X_k}(i_k)\log p_{X_k}(i_k) - p^{(n)}_{X_k}(i_k)\log p^{(n)}_{X_k}(i_k)\right|$$

$$\leq [\log n - 1]\sum_{i_k}\left|p_{X_k}(i_k) - p^{(n)}_{X_k}(i_k)\right| \leq [\log n - 1]\sum_{i_1,\ldots,i_m}|p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}| \quad \text{(A.60)}$$

when $e \leq np_{X_k}(i_k)$ and $e^2 \leq n$, and the last inequality follows via marginalization. Notice,

$$e \leq np_{i_1,\ldots,i_m} \leq n\sum_{i_j, j\neq k} p_{i_1,\ldots,i_m} = np_{X_k}(i_k)$$

so that (A.60) follows when $e \leq \mathbb{E}\left[Z^{(n)}_{i_1,\ldots,i_m}\right]$ and $e^2 \leq n$.

From (A.54) then, and theorem 6,

$$\mathbb{P}\left[|H_n - H| > \epsilon\right] \leq \mathbb{P}\left[\sum_{i_1,\ldots,i_m}|p^{(n)}_{i_1,\ldots,i_m} - p_{i_1,\ldots,i_m}| > \frac{\epsilon}{(\log n - 1)(m+1)}\right]$$

$$\leq e^{-8\frac{n}{(\log n-1)^2}\cdot\frac{\epsilon^2}{(m+1)^2\prod_{k=1}^m N_{X_k}}}. \quad \text{(A.61)}$$

when $e \leq np_{i_1,\ldots,i_m} = \mathbb{E}\left[Z^{(n)}_{i_1,\ldots,i_m}\right]$ and $e^2 \leq n$. □

A.2.4 Theorem 8

In this section, we wish to prove theorem 8 regarding concentration of trivariances.

Theorem 8. *Let X, Y and Z be random vectors in $\mathbb{R}^d$ bounded by M, and (x_i, y_i, z_i) be independent samples of (X, Y, Z). Define*

$$T(X,Y,Z) = \mathbb{E}\left[(X - \mathbb{E}\left[X\right]) \otimes (Y - \mathbb{E}\left[Y\right]) \otimes (Z - \mathbb{E}[Z])\right],$$

$$T(X_n, Y_n, Z_n) = \mathbb{E}_n\left[(X - \mathbb{E}_n[X]) \otimes (Y - \mathbb{E}_n[Y]) \otimes (Z - \mathbb{E}_n[Z])\right]$$

$$= \frac{1}{n}\sum_{i=1}^n (x_i - \mathbb{E}_n[X]) \otimes (y_i - \mathbb{E}_n[Y]) \otimes (z_i - \mathbb{E}_n[Z]), \quad \text{(A.62)}$$

$$C_n(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}[X]) \otimes (y_i - \mathbb{E}[Y]).$$

and the two norms

$$\langle T, T\rangle = \sum_{i,j,k} T(i,j,k)T(i,j,k), \quad \text{and } \langle C, C\rangle = \sum_{i,j} C(i,j)C(i,j).$$

Suppose

$$\mathbb{E}\left[||(X - \mathbb{E}[X]) \otimes (Y - \mathbb{E}[Y]) \otimes (Z - \mathbb{E}[Z]) - T(X,Y,Z)||^p\right] \leq \frac{1}{2}p!b^2L^{p-2}, \;\; p \geq 2,$$

for two positive constants b and L. Then there exists a constant C so that

$$\mathbb{P}\left[||T(X_n,Y_n,Z_n) - T(X,Y,Z) > \epsilon\right] \leq 10\exp\left(-\frac{n\epsilon^2}{CM^6d^3}\right).$$

First, we define some of the structure arising in the theorem that extend to more general results.

As in the statement of the theorem, if X, Y and Z are random vectors in $\mathbb{R}^d$ (not necessarily bounded) with respective means $\mathbb{E}[X]$, $\mathbb{E}[Y]$ and $\mathbb{E}[Z]$, let

$$T(X,Y,Z) = \mathbb{E}\left[(X - \mathbb{E}\left[X\right]) \otimes (Y - \mathbb{E}\left[Y\right]) \otimes (Z - \mathbb{E}[Z])\right],$$

$$C_n(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}[X]) \otimes (y_i - \mathbb{E}[Y]).$$

Let $\{(x_i, y_i, z_i)\}_{i=1}^n$ be n i.i.d. samples of (X,Y,Z), and $\mathbb{E}_n[f(X,Y,Z)]$ and $\mathbb{E}_n[g(X,Y)]$ be expectations with respect to the empirical measures $\frac{1}{n}\sum_{i=1}^n \delta_{(x_i,y_i,z_i)}$ and $\frac{1}{n}\sum_{i=1}^n \delta_{(x_i,y_i)}$ respectively.

Define,

$$T(X_n, Y_n, Z_n) = \mathbb{E}_n\left[(X - \mathbb{E}_n[X]) \otimes (Y - \mathbb{E}_n[Y]) \otimes (Z - \mathbb{E}_n[Z])\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}_n[X]) \otimes (y_i - \mathbb{E}_n[Y]) \otimes (z_i - \mathbb{E}_n[Z]). \quad \text{(A.63)}$$

Conceptually, $T(X, Y, Z)$ is the true empirical trivariance and $T(X_n, Y_n, Z_n)$ is the empirical trivariance computed with the *empirical* mean. We want to investigate concentration of $T(X_n, Y_n, Z_n)$ to $T(X, Y, Z)$. To do this, we make the set of 3-tensors on $\mathbb{R}^d$, $\mathcal{M}_{3,d}(\mathbb{R})$, into a Hilbert Space with inner product:

$$\langle A, B\rangle = \sum_{i,j,k} A(i,j,k)B(i,j,k), \quad \text{forA, B} \in \mathcal{M}_{3,\text{d}}(\mathbb{R}),$$

and the space of $d \times d$ matrices, $\mathcal{M}_{2,d}(\mathbb{R})$ into a Hilbert Space with inner product,

$$\langle A, B\rangle = \sum_{i,j} A(i,j)B(i,j), \quad \text{forA, B} \in \mathcal{M}_{2,\text{d}}(\mathbb{R}).$$

We begin with the following theorem,

Theorem 9. *The following inequality holds,*

$$||T(X_n, Y_n, Z_n)|| \leq \left|\left|\frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}[X]) \otimes (y_i - \mathbb{E}[Y]) \otimes (z_i - \mathbb{E}[Z])\right|\right|$$

$$+ \left|\left|\frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}[X]) \otimes (y_i - \mathbb{E}[Y])\right|\right| ||\mathbb{E}[Z] - \mathbb{E}_n[Z]||$$

$$+ \left|\left|\frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}[X]) \otimes (z_i - \mathbb{E}[Z])\right|\right| ||\mathbb{E}[Y] - \mathbb{E}_n[Y]||$$

$$+ \left|\left|\frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbb{E}[Y]) \otimes (z_i - \mathbb{E}[Z])\right|\right| ||\mathbb{E}[X] - \mathbb{E}_n[X]||$$

$$+ 2||\mathbb{E}[X] - \mathbb{E}_n[X]||||\mathbb{E}[Y] - \mathbb{E}_n[Y]||||\mathbb{E}[Z] - \mathbb{E}_n[Z]||. \quad \text{(A.64)}$$

In particular:

$$
\begin{aligned}
&||T(X_n, Y_n, Z_n) - T(X, Y, Z)|| \\
&\quad \leq \left|\left| \frac{1}{n} \sum_{i=1}^{n} (x_i - \mathbb{E}[X]) \otimes (y_i - \mathbb{E}[Y]) \otimes (z_i - \mathbb{E}[Z]) - T(X, Y, Z) \right|\right| \\
&\qquad + \left|\left| \frac{1}{n} \sum_{i=1}^{n} (x_i - \mathbb{E}[X]) \otimes (y_i - \mathbb{E}[Y]) \right|\right| ||\mathbb{E}[Z] - \mathbb{E}_n[Z]|| \\
&\qquad + \left|\left| \frac{1}{n} \sum_{i=1}^{n} (x_i - \mathbb{E}[X]) \otimes (z_i - \mathbb{E}[Z]) \right|\right| ||\mathbb{E}[Y] - \mathbb{E}_n[Y]|| \\
&\qquad + \left|\left| \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbb{E}[Y]) \otimes (z_i - \mathbb{E}[Z]) \right|\right| ||\mathbb{E}[X] - \mathbb{E}_n[X]|| \\
&\qquad\qquad + 2||\mathbb{E}[X] - \mathbb{E}_n[X]||||\mathbb{E}[Y] - \mathbb{E}_n[Y]||||\mathbb{E}[Z] - \mathbb{E}_n[Z]||. \quad \text{(A.65)}
\end{aligned}
$$

Proof of Theorem 9:

$$
\begin{aligned}
&T(X_n, Y_n, Z_n) \\
&\quad = \mathbb{E}_n\Big[(X - \mathbb{E}[X] + \mathbb{E}[X] - \mathbb{E}_n[X]) \otimes (Y - \mathbb{E}[Y] + \mathbb{E}[Y] - \mathbb{E}_n[Y]) \\
&\qquad\qquad \otimes (Z - \mathbb{E}[Z] + \mathbb{E}[Z] - \mathbb{E}_n[Z])\Big] \\
&\qquad = \mathbb{E}_n\left[(X - \mathbb{E}[X]) \otimes (Y - \mathbb{E}[Y]) \otimes (Z - \mathbb{E}[Z])\right] \\
&\qquad + \mathbb{E}_n\left[(X - \mathbb{E}[X]) \otimes (Y - \mathbb{E}[Y]) \otimes (\mathbb{E}[Z] - \mathbb{E}_n[Z])\right] \\
&\qquad + \mathbb{E}_n\left[(X - \mathbb{E}[X]) \otimes (\mathbb{E}[Y] - \mathbb{E}_n[Y]) \otimes (Z - \mathbb{E}[Z])\right] \\
&\qquad + \mathbb{E}_n\left[(X - \mathbb{E}[X]) \otimes (\mathbb{E}[Y] - \mathbb{E}_n[Y]) \otimes (\mathbb{E}[Z] - \mathbb{E}_n[Z])\right] \\
&\qquad + \mathbb{E}_n\left[(\mathbb{E}[X] - \mathbb{E}_n[X]) \otimes (Y - \mathbb{E}[Y]) \otimes (Z - \mathbb{E}[Z])\right] \\
&\qquad + \mathbb{E}_n\left[(\mathbb{E}[X] - \mathbb{E}_n[X]) \otimes (Y - \mathbb{E}[Y]) \otimes (\mathbb{E}[Z] - \mathbb{E}_n[Z])\right] \\
&\qquad + \mathbb{E}_n\left[(\mathbb{E}[X] - \mathbb{E}_n[X]) \otimes (\mathbb{E}[Y] - \mathbb{E}_n[Y]) \otimes (Z - \mathbb{E}[Z])\right] \\
&\qquad + \mathbb{E}_n\left[(\mathbb{E}[X] - \mathbb{E}_n[X]) \otimes (\mathbb{E}[Y] - \mathbb{E}_n[Y]) \otimes (\mathbb{E}[Z] - \mathbb{E}_n[Z])\right] \\
&\qquad = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mathbb{E}[X]) \otimes (y_i - \mathbb{E}[Y]) \otimes (z_i - \mathbb{E}[Z]) \\
&\qquad\qquad + C_n(X, Y) \otimes (\mathbb{E}[Z] - \mathbb{E}_n[Z]) \\
&\qquad\qquad + C_n(X, Z) \otimes (\mathbb{E}[Y] - \mathbb{E}_n[Y]) \\
&\qquad\qquad + C_n(Y, Z) \otimes (\mathbb{E}[X] - \mathbb{E}_n[X]) \\
&\qquad - 2(\mathbb{E}[X] - \mathbb{E}_n[X]) \otimes (\mathbb{E}[Y] - \mathbb{E}_n[Y]) \otimes (\mathbb{E}[Z] - \mathbb{E}_n[Z]). \quad \text{(A.66)}
\end{aligned}
$$

Now use the triangle inequality with the observation that $||u \otimes v|| \leq ||u||||v||$. □

For concentration of the means in theorem 9 we have the following theorem, proven in Little and Maggioni (2011):

Theorem 10. *Let Y be a random vector in $\mathbb{R}^d$. The following concentration around*

the mean hold for any $t > 0$*:*

$$\mathbb{P}\left[||E[Y] - \mathbb{E}_n[Y]|| \geq \frac{C}{\sqrt{n}}t^{\alpha}\right] \leq 2e^{-t^2},$$

where

1. $C = c_0 \cdot M = 2\sqrt{2}M$ *and* $\alpha = 1$ *if* $||Y|| \leq M$ *a.s.;*

2. $C = c_1 \cdot \sqrt{D} = 4c\sqrt{D}\log 2$ *and* $\alpha = 2$ *if* $\langle Y, \theta\rangle$ *is centered and subgaussian with moment* 1 *for every* $\theta \in \mathbb{S}^{d-1}$.

Finally, we have the following concentration inequality for the first term on the RHS in theorem 9 Pinelis (1991), Pinelis (1994):

Theorem 11. (Pinelis' Inequalities)

Let Y_i, $i = 1, \ldots, n$ *be a sequence of* n *zero mean random variables with values in a Hilbert space* $(H, \langle\cdot,\cdot\rangle_H)$ *satisfying*

$$\mathbb{E}\left[||Y_i||_H^p\right] \leq \frac{1}{2}p!b^2L^{p-2}, \;\; p \geq 2,$$

for two positive constants b *and* L*. Then, for all* $t > 0$*,*

$$\mathbb{P}\left[\left|\left|\frac{1}{n}\sum_{i=1}^{n} Y_i\right|\right|_H \leq \frac{Lt^2}{n} + \frac{\sqrt{2}bt}{\sqrt{n}}\right] \geq 1 - 2e^{-t^2}.$$

If $||Y_i|| \leq M$ *for all* $i = 1 \ldots, n$ *then,*

$$\mathbb{P}\left[\left|\left|\frac{1}{n}\sum_{i=1}^{n} Y_i\right|\right|_H \leq \frac{\sqrt{2}Mt}{\sqrt{n}}\right] \geq 1 - 2e^{-t^2}.$$

Finally, we present the proof of theorem 8.

Proof of Theorem 8:

By theorem 9:

$$\begin{aligned}
&\mathbb{P}\left[||T(X_n,Y_n,Z_n)-T(X,Y,Z)||>\epsilon\right] \\
&\quad\leq \mathbb{P}\left[\left|\left|\frac{1}{n}\sum_{i=1}^{n}(x_i-\mathbb{E}[X])\otimes(y_i-\mathbb{E}[Y])\otimes(z_i-\mathbb{E}[Z])-T(X,Y,Z)\right|\right|>\frac{\epsilon}{5}\right] \\
&+\mathbb{P}\left[||C_n(X,Y)||\,||\mathbb{E}[Z]-\mathbb{E}_n[Z]||>\frac{\epsilon}{5}\right]+\mathbb{P}\left[||C_n(X,Z)||\,||\mathbb{E}[Y]-\mathbb{E}_n[Y]||>\frac{\epsilon}{5}\right] \\
&\quad+\mathbb{P}\left[||C_n(Y,Z)||\,||\mathbb{E}[X]-\mathbb{E}_n[X]||>\frac{\epsilon}{5}\right] \\
&\quad+\mathbb{P}\left[2||\mathbb{E}[X]-\mathbb{E}_n[X]||||\mathbb{E}[Y]-\mathbb{E}_n[Y]||||\mathbb{E}[Z]-\mathbb{E}_n[Z]||>\frac{\epsilon}{5}\right]. \quad \text{(A.67)}
\end{aligned}$$

Now, since X, Y, Z are all bounded by M, we have that

$$||(X-\mathbb{E}[X])\otimes(Y-\mathbb{E}[Y])\otimes(Z-\mathbb{E}[Z])-T(X,Y,Z)||\leq 16M^3\sqrt{d^3}$$

so by Pinelis' inequality (theorem 11),

$$\begin{aligned}
&\mathbb{P}\left[\left|\left|\frac{1}{n}\sum_{i=1}^{n}(x_i-\mathbb{E}[X])\otimes(y_i-\mathbb{E}[Y])\otimes(z_i-\mathbb{E}[Z])-T(X,Y,Z)\right|\right|>\frac{\epsilon}{5}\right] \\
&\qquad\leq 2\exp\left(-\frac{n\epsilon^2}{C_1M^6d^3}\right) \quad \text{(A.68)}
\end{aligned}$$

for some constant C_1.

By theorem 10,

$$\begin{aligned}
&\mathbb{P}\left[||C_n(X,Y)||\,||\mathbb{E}[Z]-\mathbb{E}_n[Z]||>\frac{\epsilon}{5}\right]\leq 2\exp\left(-\frac{n\epsilon^2}{C_2'M^2||C_n(X,Y)||^2}\right) \\
&\qquad\leq 2\exp\left(-\frac{n\epsilon^2}{C_2M^6d^2}\right) \quad \text{(A.69)}
\end{aligned}$$

and similarly for the other two terms for some (same) constant C_2. The last inequality follows from the observation that $||C_n(X,Y)||^2\leq(2M)^4d^2$.

For the last term, notice that

$$\begin{aligned}
&\mathbb{P}\left[||\mathbb{E}[X]-\mathbb{E}_n[X]||||\mathbb{E}[Y]-\mathbb{E}_n[Y]||||\mathbb{E}[Z]-\mathbb{E}_n[Z]|| > \frac{\epsilon}{10}\right] \\
&\qquad \leq \mathbb{P}\left[||\mathbb{E}[X]-\mathbb{E}_n[X]|| > \sqrt[3]{\frac{\epsilon}{10}}\right] + \mathbb{P}\left[||\mathbb{E}[Y]-\mathbb{E}_n[Y]|| > \sqrt[3]{\frac{\epsilon}{10}}\right] \\
&\qquad\qquad + \mathbb{P}\left[||\mathbb{E}[Z]-\mathbb{E}_n[Z]|| > \sqrt[3]{\frac{\epsilon}{10}}\right] \\
&\qquad\qquad\qquad \leq 6\exp\left(-\frac{n\epsilon^{\frac{2}{3}}}{C_3M^2}\right) \quad \text{(A.70)}
\end{aligned}$$

for some constant C_3, and where the last inequality comes from an application of theorem 10.

Putting them all together, and taking C to be the largest C_i, we get:

$$\mathbb{P}\left[||T(X_n,Y_n,Z_n)-T(X,Y,Z) > \epsilon\right] \leq 10\exp\left(-\frac{n\epsilon^2}{CM^6d^3}\right). \quad \square$$

A.2.5 Results of Benchmark Data Sets

Table A.1: Summary of percentage errors in the benchmark data sets using correlations and pairwise similarities.

	P Chain		π Chain		$E1$ Chain	
Data Set	CV Error	Test Error	CV Error	Test Error	CV Error	Test Error
USPS	6.83%	6.15%	18%	18.67%	18.67%	18.8%
BCI	43.92%	49.31%	49.17%	49.78%	49.25%	50%
g241c	43.5%	50.81%	43.75%	49.96%	45.5%	50.96%
COIL	19.25%	21.48%	66%	66.89%	66.83%	70.3%
g241n	36.25%	36.41%	47.75%	48.68%	48.5%	49.49%

Table A.2: Summary of mixing chain percentage errors in in the benchmark data sets using correlations and mixing different chains.

	$P-\pi$ Chain		$P-E1$ Chain	
Data Set	CV Error	Test Error	CV Error	Test Error
USPS	8%	7.58%	7.42%	6.96%
BCI	44.42%	48.44%	43.33%	48.39%
g241c	41.42%	50.48%	41.33%	50.93%
COIL	28.83%	29.63%	25.92%	26.91%
g241n	38.25%	38.38%	35.92%	38.03%

Table A.3: In the last column, for each data set, the test errors of the best performing method with model selection among those in Chapelle (2010) is given. In each of the remaining columns, the performance of diffusive transductive inference with model selection taken from Szlam (2008) is given. The purpose of this unfair comparison is to highlight the potential of our methods versus other methods on different data sets. FAKS stands for function dependent kernel smoothing, FAHC stands for function dependent harmonic classifier, and FAEF stands for function dependent eigenfunctions.

Data Set	FAKS	FAHC	FAEF	Best of Other Methods
USPS	4.0%	3.9%	3.3%	4.7% (LapRLS, Disc. Reg.)
BCI	45.5%	45.3%	47.8%	31.4% (LapRLS)
g241c	19.8%	21.5%	18.0%	22.0% (NoSub)
COIL	12.0%	11.1%	15.1%	9.6% (ClusterKernel)
g241n	11.0%	12.0%	9.2%	23.6% (LapSVM)

Bibliography

Alpaydin, E. (2010), *An Introduction to Machine Learning: Second Edition*, MIT Press.

Anderson, R. and Chung, F. (2006), "Local graph partitioning using pagerank vectors," Berkeley CA, IEEE, IEEE Computer Society.

Anderson, R. and Chung, F. (2007), "Local partitioning for directed graphs using PageRank," San Diego CA, NSF, Springer.

Ashburner, M. (2000), "Gene Ontology: tool for the unification of biology," *Nat Genet*, 25, 25–29.

Belkin, M. and Niyogi, P. (2003), "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation*, 15.

Borg, I. and Groenen, P. (2005), *Modern multidimensional scaling: theory and applications*, Springer-Verlag.

Chapelle, O. (2010), *Semi-Supervised Learning*, MIT Press.

Chung, F. (1997), *Spectral Graph Theory*, American Mathematical Society.

Chung, F. (2005), "Laplacians and Cheeger inequalities for directed graphs," *Annals of Combinatorics*, 9, 1–19.

Coifman, R. and Lafon, S. (2006), "Diffusion Maps," *Appl. Comput. Harmon. Anal.*, 21, 5–30.

Coifman, R. and Maggioni, M. (2006), "Diffusion Wavelets," *Appl. Comput. Harmon. Anal.*, 21, 53–94.

Devroye, L. and Gyorfi, L. (1985), *Nonparametric Density Estimation: The L1 View*, John Wiley & Sons.

Gray, R. M. (1990), *Entropy and Information Theory*, Springer-Verlag.

Horn, R. and Johnson, C. (1985), *Matrix Analysis*, Cambridge University Press.

Ideker, T. (2001), "Integrated genomic and proteomic analyses of a systemically perturbed metabolic network," *Science*, 292, 929–934.

LeCun, Y. (1998), "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 86, 2278–2324.

Levin, D. and Peres, Y. (2009), *Markov Chains and Mixing Times*, American Mathematical Society.

Little, A. V. and Maggioni, M. (2011), "Multiscale Geometric Methods for Data Sets I: Intrinsic dimension," Preprint.

Mahoney, M. (2010), "A Spectral Algorithm for Improving Graph Partitions with Applications to Exploring Data Graphs Locally," Preprint.

Nakagawa, T. (2008), "A tissue biomarker panel predicting systemic progression after PSA recurrence post-denitive prostate cancer therapy," *PLoS One*, 3, e2318.

Newman, M. (2010), *Networks: An Introduction*, Oxford University Press.

Norris, J. (1997), *Markov Chains*, Cambridge University Press.

Pinelis, I. (1991), "An Approach to Inequalities for the Distributions of Infinite-dimensional Martingales," *Progr. Probab.*, 30, 128–134.

Pinelis, I. (1994), "Optimum Bounds for the Distributions of Martingales in Banach Spaces," *Ann. Probab.*, 4, 1679–1706.

Spielman, D. and Teng, S. (2004), "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," *ACM STOC*, 4, 81–90.

Szlam, A. (2008), "Regularization on Graphs with Function-Adapted Diffusion Processes," *Journal of Machine Learning Research*, 9, 1711–1739.

Tenenbaum, J. B. (2000), "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, 290, 2319–2323.

Tsybakov, A. (2010), *Introduction to Nonparametric Estimation*, Springer.

Zhou, D. (2005), "Beyond Pairwise Classification and Clustering Using Hypergraphs," Tech. rep.

Biography

Prakash Balachandran was born on July 6, 1984 in Edmonton, Alberta, Canada. At two, he moved to Calgary, Alberta, Canada where he lived until he was 12 before moving to West Windsor, New Jersey, USA.

At 18, he attended Cornell University where in 2006 he earned his B.A., graduating *magna cum laude* in Mathematics and *cum laude* in Physics.

Upon graduation, he enrolled in Duke University's PhD program in Mathematics. He attained his M.A. in Mathematics in 2007, passed the Actuarial Exam P in 2008, and attained his PhD in Mathematics in 2011.

In 2010-2011, he was a graduate student fellow in SAMSI's year long program in Complex Networks. His next appointment will be working as a post-doctoral researcher at Boston University.