



โครงการ
การเรียนการสอนเพื่อเสริมประสบการณ์

ชื่อโครงการ การเรียนรู้บนแมนิโฟลด์และการประยุกต์ใช้ในวิทยาศาสตร์
Manifold Learning and its Applications in Science

ชื่อนิสิต นายอภิมุข สรแสง

เลขประจำตัว 5933452723

ภาควิชา ฟิสิกส์

ปีการศึกษา 2562

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

Manifold Learning and its Applications in Science

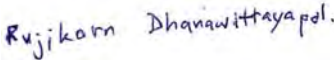
Mr. Apimuk Soransaeng

A report submitted to the Department of Physics of Chulalongkorn
University in partial fulfillment of the requirements for the degree of
Bachelor of Science in Physics
Academic Year 2019


Project Title Manifold Learning and its Applications in Science
By Apimuk Sornsaeng
Field of Study Physics
Project Advisor Thiparat Chotibut, Ph.D.
Academic Year 2019

This report is submitted to the Department of Physics, Faculty of Science, Chulalongkorn University, in partial fulfillment of the requirements for the degree of Bachelor of Science.

This report has been approved by the committee:


..... Chairman
(Rujikorn Dhanawittayapol, Ph.D.)


..... Project Advisor
(Thiparat Chotibut, Ph.D.)


..... Committee
(Associate Professor Surachate Limkunnerd, Ph.D.)

Project Title	Manifold Learning and its Applications in Science
By	Apimuk Soransaeng
Field of Study	Physics
Project Advisor	Thiparat Chotibut, Ph.D.
Academic Year	2019

Abstract

In the age of big data, unsupervised machine learning plays crucial roles in detecting statistical patterns hidden in gigantic dataset. Taking root in statistical physics of random walks and heat diffusion on networks, *Diffusion Maps* are one of the most efficient modern classical unsupervised algorithms for clustering high-dimensional dataset. Not only it can automatically discover hidden statistical structure in a high-dimensional dataset, but it can also projects the data into a lower dimensional embedding where the majority of the data structure reside. Such projections are termed *nonlinear dimensionality reduction* or *manifold learning* in machine leaning literature. In the first part of this thesis, we begin by reviewing the physics of classical random walks on a graph which motivates the construction of Diffusion Maps. We will discuss how Diffusion Maps can perform clustering as well as nonlinear dimensionality reduction based on the properties of Markov transition matrix defined on a dataset-associated graph. We then showcase the usefulness of Diffusion Maps to learn low dimensional embedding in some real data samples. In the second part of this thesis, we bring diffusion maps into the realm of quantum algorithms. Motivated by advances in modern near-term quantum devices, we explore a construction of Quantum Diffusion Maps. By exploiting coherent state encoding scheme into Quantum RAM, we outline how to achieve both quantum computational speedup as well as quantum storage capacity reduction for quantum computations of Diffusion Maps on a quantum device. Lastly, it's known that quantum walks can spread faster than its classical counterparts; we construct quantum walk protocols that perhaps can provide an alternative way to perform unsupervised data clustering, given that one can create quantum walks on quantum devices or quantum simulators.

Acknowledgement

I'd like to express my gratitude to my advisor, Dr. Thiparat Chotibut for introducing me to the beautiful world of statistical physics and quantum technologies, and for being supportive at all time on research and on the writing of the thesis. I'm thankful for his aspiring supervision, helpful critiques, and friendly advices. Secondly, I'd like to thank the Development and Promotion of Science and Technology Talents Project (DPST) for the scholarship to conduct research overseas, and for exciting scientific experience and journeys. Thirdly, I especially thank Prof. Dimitris Angelakis from the Centre for Quantum Technologies (CQT) in Singapore for the research opportunity, hospitality, and useful advices on quantum algorithms. Also, a special gratitude to Dr. Jirawat Tangpanitanon and Supanut Thanasilp from CQT for a great experience in quantum technologies research, and for helping me navigate through my thesis. Many pieces of knowledge I learned from the three CQT's. Lastly, I'd like to thank my parents and friends who have been nurturing, encouraging, and pushing me in the right path.

Contents

Abstract	ii
Acknowledgement	iii
1 Introduction	1
1.1 Basic Ideas of Random Walk	5
1.2 Identifying Clusters in a Graph: Random Walk Approach	8
2 Diffusion Map	10
2.1 Basic Definition of Graph	10
2.2 Random Walk on Graph	11
2.3 Diffusion Map	11
2.4 Properties of the Eigenspace of the Transition Matrix . .	13
2.5 Random Walk and Heat Diffusion	14
2.6 Implementation of Diffusion Map	16
3 Quantum Algorithm for Diffusion Map	23
3.1 Encoding Data into Quantum Random Access Memory .	24
3.2 Constructing Transition Matrix from Coherent States . .	25
3.3 Quantum Algorithms to Find Eigenvalues and Eigenvec- tors	27
4 Quantum Walk for Data Clustering	29
4.1 Limiting Distribution of a Quantum Walk	31
4.2 Data Clustering with Coined Quantum Walk	32
5 Conclusion	35
Bibliography	36

Appendices	38
Appendix A Coherent States	38
Appendix B Quantum Phase Estimation (QPE)	40
B.1 An Alternative Method for Finding Eigenvalues and Eigen- vectors	42

List of Figures

1.1	Comparisons of non-linear dimensionality reduction methods and their visualizations. Similar color code represents data in proximity (in the same or nearby clusters). Diffusion maps reveal underlying proximity structure better than PCA and tSNE for the high-dimensional gene expression patterns classification during cell-type differentiation processes (B). (Figure adapted from [7]) . . .	3
2.1	Classical algorithm for diffusion maps computation . . .	12
2.2	(Left) A finite one-dimensional lattice as a toy dataset to test the manifold learning algorithm. (Right) The dataset colored by the second component of the diffusion map, showing that nearby points in the original space is mapped to nearby points in the diffusion space (second component).	17
2.3	First four components of the diffusion map (2.4) with diffusion time $t = 1$ (vertical axis) versus the location along the one dimensional line of each data point (horizontal axis). The color codes are 1st (Blue), 2nd (Orange), 3rd (Green), and 4th (Red) components. Only the second component has a one-to-one mapping between the dataset and their coordinates in the diffusion space. . . .	17

2.4	A non-linear one-dimensional data manifold embedded in a two dimensional space. The dataset is colored by (left) the second component of the diffusion map, and (right) the first component of PCA. Clearly, diffusion map discovers the notion of “proximity along the lower-dimensional non-linear data manifold” while PCA fails to do so!	18
2.5	(Left) First four components of the diffusion map with diffusion time $t = 1$ (vertical) of each data point index (horizontal) corresponding to the color blue, orange, green, red, respectively, for dataset in fig. 2.4. Again only the second component shows a one-to-one mapping between the data points and the coordinate in the diffusion space. (Right) The first two principal components of PCA (vertical) for each data point index (horizontal)	18
2.6	(Left) A toroidal helix, and (Right) a Swiss roll. Color code label different data points. Nearby colors label nearby points on the lower-dimensional embedding.	19
2.7	Data points projected onto the first two components of the diffusion map with the diffusion time $t = 1$ of (Left) a toroidal helix, and (Right) a Swiss roll. One can see that the notion of proximity in the original space is preserved (color variation) and that the shape reflect the flattened out version of the original non-linear data manifolds. Interestingly, for the diffusion map of the swiss roll, the map appears to discover that the two-dimensional roll can be generated from a one-dimensional line, with extra layers in the z direction.	19
2.8	Non-overlapping doughnut-shape data clouds projected onto two dimensions (each doughnut lives on a different z value). Note that the right cluster is more dense than the left.	20

2.9	(Left) Data points labeled by the second component of the diffusion map by picking a small σ , but not according to (2.21). However, by setting the scale of σ according to (2.21), the second component can identify the correct clusters within diffusion time $t = 1$! (Right). This is because the density of points on the right cluster is higher than the left, and the two clusters are in fact quite separated in the z directions. By picking the global parameter σ according to (2.21), the second component of the diffusion map with $t = 1$ can separate the two clusters by their associated density of points.	21
2.10	Diffusion map automatically discovers the relationship between three cultivars of wines grown in the same region of Italy. The dataset from [18] consists of 178 data samples, each consists of 13 quantities (features) extracted from the chemical analysis of wines. (Left) Projection of the dataset onto 2 (of 13)-dimensional feature space. Different colors encode different cultivars of wines. (Right) Projection of the dataset onto the second and the third component of the diffusion map shows almost perfect identification of the three cultivars. In addition, diffusion map also reveals the relationship between the three cultivars of wines. Namely, the green type is closely related to the other two.	22
3.1	Comparison between classical and quantum diffusion map.	28
4.1	Generic behavior of the probability of a quantum walker starting at a vertex i to remain in the state i in the subsequent time steps. The figure shows large fluctuations due to quantum interference in the time-evolved probability; however, the time average tends to converge to a stationary value, verifying (4.13).	32
4.2	A toy dataset consisting of 4 clusters ($k = 4$) on which we will run a coined Quantum walk algorithm for clustering.	32

4.3	The time-average probability of a quantum walker starting at vertex 0 on a weighted graph defined by the synthetic dataset of Fig. 4.2 (Left) at $t = 1$ and (Right) at $t = 10$. Only vertices that lie in the same cluster as the initial vertex gain a non-zero time-average probability!	33
4.4	Successful data clustering of the dataset in Fig. 4.2 using a coined quantum walk scheme with 4 reinitializations. Different colors identify different clusters.	34
B.1	The quantum circuit of QPE, taken from [19].	41

Chapter 1

Introduction

Modern machine learning techniques are extraordinarily efficient at learning and representing high dimensional statistics of labeled dataset, provided a large number of input-label pairs $\{(\mathbf{x}^{(m)}, y^{(m)})\}$ are given. Among the class of supervised learning, a biologically inspired feed-forward neural network architecture, called Deep Learning, has reigned and empowered modern AI tasks, ranging from visual recognitions to automatic speech recognitions. Given abundant input-label data pairs, Deep Learning can learn and represent a complex conditional probability distribution $p_{\theta}(y|\mathbf{x})$ fairly efficiently, where $\mathbf{x} \in \mathbb{R}^n$ is an input data, y is an associated data label, and θ are the parameters that parametrize the deep neural network. For instance, a well-trained deep network will convert a vector \mathbf{x} encoding pixel colors and pixel intensities of a picture of a dog into a label “dog” with high probability, even though the network itself has never seen such instance of the dog picture during training.

Although Deep Learning is a powerhouse of modern machine learning applications, it requires a large amount of labeled data to build a useful representation. In fact, exhaustively labeling every data correctly is a major obstacle in constructing a useful deep neural network. In addition, many realistic tasks, such as identifying gene expressions patterns in response to new drugs, do not have predefined or well-defined labels. This lack of domain knowledge (lack of labels) urges for automatic detection or classification of patterns in an unsupervised way, which is the task of an unsupervised learning algorithm. In unsupervised learning, a large number of high-dimensional data $\{\mathbf{x}^{(m)}\}$

will be automatically classified into different classes according to the proximity to other data points. For example, a set of weights sampled from 100 patients might show a bimodal distribution, one with a low average weight and the other with a high average weight; a useful unsupervised learning algorithm will be able to separate the dataset into two groups automatically and correctly. The aforementioned example might seem uninspiring; however, automatic pattern detection and classification plays a crucial role in many realistic tasks whose data lives in high dimension, such as in classifying gene expression patterns in response to drug therapy, which is a typical “big data” problem in bioinformatics.

Prototypical unsupervised learning include Principal Component Analysis (PCA), Kernel Principal Component Analysis (Kernel-PCA), k-mean clustering, T-distributed Stochastic Neighbour Embedding (t-SNE), and Support Vector Machine (SVM). These algorithms minimize cost functions, whose minimization procedure yields the “decision boundaries” that suppose to automatically label different “data clusters.” It is known that these classes of algorithms, although easy to implement on classical computers, suffer from the curse-of-dimensionality, in which the higher the dimensions the data lives in, the slower and the less accurate it is for the algorithm to find the decision boundaries. In this regards, Diffusion maps [9] were proposed as a modern unsupervised learning algorithm that not only performs clustering without suffering from the curse of dimensionality, but also performs a non-linear dimensionality reduction, automatically discovering the lower-dimensional manifold in which the data is embedded, see the fig. 1.1.

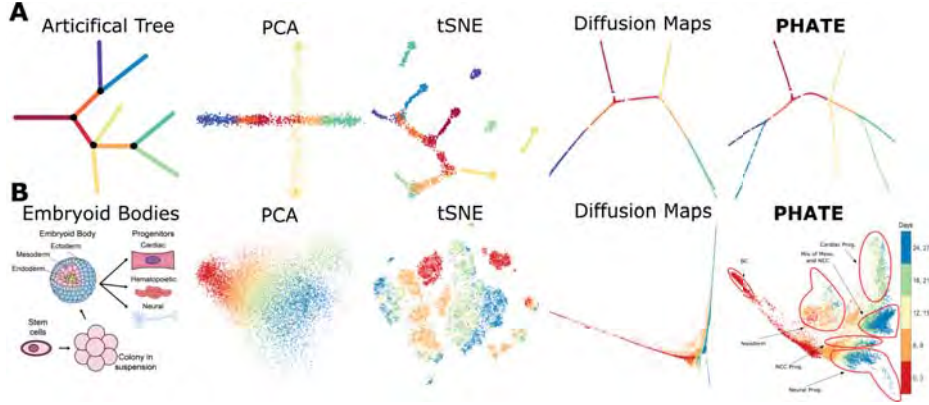


Figure 1.1: Comparisons of non-linear dimensionality reduction methods and their visualizations. Similar color code represents data in proximity (in the same or nearby clusters). Diffusion maps reveal underlying proximity structure better than PCA and tSNE for the high-dimensional gene expression patterns classification during cell-type differentiation processes (B). (Figure adapted from [7])

Classical diffusion maps are simple, implementable, and inspired from non-equilibrium statistical mechanics of random walkers. The main idea is that random walkers on a graph will spend most of their time on a well-connected component, before transitioning to other well-connected components in the graph. If we assign a graph structure to the dataset of interests where each vertex corresponds to each data point and the weighted edge between any two vertices encodes the distance (in the Euclidean space the data lives in) between the two data points, then random walkers, on such graph associated to the dataset, will spend most of their time in a well-connected component and perhaps build up a unique stationary distribution around a strongly connected component, automatically exploring a “cluster” of data points living in the same strongly connected component. The degree of connectedness can be tuned in the weighted edges between any pair of data points, which typically take the form of a Gaussian Kernel function:

$$W_{ij} = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{\sigma}\right) \quad (1.1)$$

where $\|\cdot\|$ denotes the Euclidean L_2 norm and σ here assigns the degree of proximity between the two data points. This problem of finding stationary distributions can be solved by block-diagonalizing the associated transition matrix, which is a row stochastic matrix whose columns

sum to one. More explicitly, we would block-diagonalize the transition probability (transition matrix) of a walker to move from vertex j to vertex i , defined in one time-step as

$$P_{ij} = \frac{W_{ij}}{\sum_i W_{ij}}, \quad (1.2)$$

where the denominator ensures a proper normalization (i.e. the transition out of j or remains in j in one time step has probability 1). Such row stochastic matrix has all eigenvalues lie in $[0, 1]$, and only the eigenvectors associated with the degenerate eigenvalue 1 encapsulates the stationary distributions; all other eigenvectors with eigenvalue less than 1 will decay exponentially away with time, a feature shared by eigenfunction expansions of heat equations, either in Euclidean space or in curved space. As a result, numerically solving for the degenerate subspace with eigenvalue 1 not only provide the identification of data clusters, but also extract the lower-dimensional linear manifold where most of the data lives in without suffering from the curse of dimensionality!

In the past few years, quantum counterparts of classical unsupervised learning have been proposed. Notable algorithms are quantum PCA [1] and quantum Support Vector Machine [P. Rebentrost, et. al., 2014], which promise a quantum computational speedup over the classical counterparts, provided a fault-tolerant quantum computing can be achieved. In this subtheme, we will explore the quantum counterparts of the classical diffusion maps, which, to the best of our knowledge, has not been investigated. It is well known that quantum walks can provide a quadratic speed up on the spread of walkers; namely, for a symmetric random walk, the width of the classical distribution spread as \sqrt{t} whereas the width of quantum walk distribution spreads as t , where t is the number of steps taken by the walkers. [17]. It is thus plausible to hypothesize that random walk based-clustering algorithm can achieve quantum computational speedup by implementing quantum walkers on a graph associated with classical high-dimensional dataset on quantum devices.

Another relevant direction concerns quantum computational speedup that can be achieved by adopting the Quantum Phase Estimation (QPE) algorithm and its variants [3], to find the degenerate subspace with

eigenvalue 1 of the transition matrix. In addition, our preliminary results show that we can efficiently construct the aforementioned Gaussian kernel exploiting the coherent state-based qRAM encoding of classical data. Combining both QPE and our coherent state representation of the kernel will enable us to both perform quantum data compression and to achieve quantum computational speedup in Quantum Diffusion Maps, which hopefully is implementable on near-term devices.

The outline of this thesis is as follows. We begin in the next section by reviewing statistical physics of classical random walkers, with the emphasis on the convergence to the stationary distribution which will be the crux of diffusion maps. In chapter 2, we review diffusion maps algorithms, with the inspiration from random walks on graphs. We discuss how to embed a dataset into a graph structure, and how random walks on a data-associated graph that defines Markov transition matrix enables data clustering and dimensionality reduction. In the end, this non-linear dimensionality reduction algorithm requires only the computation of the eigenspaces corresponding to a few largest eigenvalues of the Markov transition matrix. The implementation of diffusion maps on some data samples are provided in section 2.6. We then explore quantum algorithms for classical diffusion maps in chapter 3 and 4. We first explore how to compress classical data into the quantum random access memory (qRAM) via coherent state representation. After reviewing quantum phase estimation (QPE) which provides quantum computational speedup for finding eigenspaces, we discuss how to quantum mechanically perform dimensionality reduction by diffusion map using both QPE and coherent state representation of the data. This leads to the proposal of *Quantum Diffusion Map*. Lastly, in Chapter 4, we propose quantum random walk algorithm as an alternative means for data clustering on a quantum device, and showcase successful implementations via simulations on a toy dataset.

1.1 Basic Ideas of Random Walk

Classical random walk is a stochastic process that describes a time evolution of random events. Its framework enables the understanding of irregular motions of pollen grains suspended in liquids, to the under-

standing of noise in semiconductor devices, to mathematical modeling of stock market price fluctuations, to name a few. A prototypical example of random walk is the random walk on a one-dimensional lattice. Consider an integer number line \mathbb{Z} with a random walker starting at the origin. The random walker has the same transition probability to hop to the left or to the right by one step per unit time. In this setting, the probability of the walker to be on the lattice site n at time t is

$$p(n, t) = \frac{1}{2^t} \binom{t}{\frac{t+n}{2}}. \quad (1.3)$$

At long times, the binomial distribution converges to the normal distribution:

$$p(n, t) \approx \frac{2}{\sqrt{2\pi t}} \exp\left\{-\frac{n^2}{2t}\right\}. \quad (1.4)$$

This is a realization of the Central Limit Theorem. It's clear that the normal distribution is centered at $n = 0$, which implies the expectation value is $\langle n \rangle = 0$. On the other hand, the standard deviation grows with the square-root of time,

$$\sigma(t) = \sqrt{\int_n n^2 p(n, t) dn} = \sqrt{t}. \quad (1.5)$$

In the long-time limit on a finite one-dimensional lattice, for a periodic boundary condition, say, we have $\lim_{t \rightarrow \infty} p(n, t) = \frac{1}{N}$ where N is a number of lattice points. This is an instance of Ergodic hypothesis, stating that at thermal equilibrium all states are equally likely. In other words, the distribution of a random walker defined above converges to a (stationary) uniform distribution.

The above description of random walk is an example of a well-known stochastic process called a Markov chain. A Markov chain describes a transition probability of events in which the transition probability depends on the states only at the latest time. We denote the probability to transition from lattice j to i in a one-time step as P_{ij} , termed a transition probability. In the previous example of random walk, the finite one-dimensional lattice, we can write the transition probability

as

$$P_{ij} = \begin{cases} 1/2 & \text{if } j = i + 1, \text{ or } j = i - 1, \\ 0 & \text{otherwise.} \end{cases} \quad (1.6)$$

Let $p(n = 0, t = 0) = 1$ be the probability of the random walker starting on the origin at the initial time, then $p(n \neq 0, t = 0) = 0$. The probability on $n = 1$ in the subsequent time can be written as

$$p(n = 1, t = 1) = \sum_n P_{1,n} p(n, t = 0) = 1/2. \quad (1.7)$$

We can write the equation above more compactly in the matrix form. Denote the transition matrix $\mathbf{P} = (P_{ij})$, and the probability distribution at time t in the vector form $\mathbf{p}(t) = (p(\cdot, t))^T$. Then, the probability distribution at time 1 is

$$\mathbf{p}(1) = \mathbf{P}\mathbf{p}(0). \quad (1.8)$$

By the Markovian assumption, the probability distribution at time t is

$$\mathbf{p}(t) = \mathbf{P}^t \mathbf{p}(0). \quad (1.9)$$

We can extend random walk to hypercubic lattice or graph naturally by mapping each index of the transition matrix to the corresponding lattice or the corresponding vertex in a graph. The important property of the transition matrix is that the summation over all columns is unity by the conservation of probability, i.e.,

$$\sum_i P_{ij} = 1. \quad (1.10)$$

In the matrix form, this means that $\mathbf{v} = \mathbf{1}$ is a trivial eigenvector of \mathbf{P} with eigenvalue 1

$$\mathbf{P}\mathbf{1} = \mathbf{1}. \quad (1.11)$$

The trivial eigenvector is the uniform distribution over all vertices, which can be reached as a possible stationary distribution.

Another important property of the eigenvalues of the transition matrix is that they live in $[0, 1]$. This can be seen as follows. Let $v_i^{(k)}$ be the largest component of an i^{th} -eigenvector corresponding to the eigenvalue λ_i . Then,

$$\lambda_i v_i^{(k)} = \sum_j P_{kj} v_i^{(j)} \quad (1.12)$$

or, equivalently, by triangle inequality,

$$|\lambda_i| \leq \sum_j P_{kj} \frac{|v_i^{(j)}|}{|v_i^{(k)}|} \leq \sum_j P_{kj} = 1. \quad (1.13)$$

Since λ_i is positive, so $0 \leq \lambda_i \leq 1$.

1.2 Identifying Clusters in a Graph: Random Walk Approach

Consider the following scenario, where we define random walk on a graph consisting of k disconnected components. In this case, the transition matrix can be block diagonalized, such that the subspace corresponding to the transitions within the same connected component is disjointed from the subspace corresponding to the transitions in other connected component (except for the shared zero vector):

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_1 & & & \\ & \mathbf{P}_2 & & \\ & & \cdots & \\ & & & \mathbf{P}_k \end{pmatrix}. \quad (1.14)$$

The characteristic equation of this matrix is

$$\det(\mathbf{P}_1 - \lambda \mathbf{I}_1) \det(\mathbf{P}_2 - \lambda \mathbf{I}_2) \cdots \det(\mathbf{P}_k - \lambda \mathbf{I}_k) = 0. \quad (1.15)$$

One trivial solution is when λ is the trivial eigenvalue 1 of each block matrix \mathbf{P}_i . In this case, we have k -degenerate subspace with eigenvalue 1. Note that the matrix \mathbf{P} can be written as the direct sum as

$$\mathbf{P} = \bigoplus_{i=1}^k \mathbf{P}_i, \quad (1.16)$$

so the eigenvalue equation becomes

$$\left(\bigoplus_{i=1}^k \mathbf{P}_i \right) \mathbf{v}_j = \lambda_j \mathbf{v}_j. \quad (1.17)$$

Observe that the direct sum between trivial eigenvector of a submatrix \mathbf{P}_i and null vectors of other subspaces, i.e.,

$$\mathbf{v}_i = \mathbf{0} \oplus \cdots \oplus \mathbf{1}_i \oplus \cdots \oplus \mathbf{0} = \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{1}_i \\ \vdots \\ \mathbf{0} \end{pmatrix}, \quad (1.18)$$

is also an eigenvector of \mathbf{P} with eigenvalue 1 where $\mathbf{1}_i$ is one vector corresponding to a block matrix \mathbf{P}_i . Therefore, the number of trivial eigenvectors is the number of the connected components in the graph. Therefore, if we initialize the distribution in one connected component, the long-time stationary distribution is the uniform distribution over such connected component; this is a realization of Ergodic hypothesis in a disconnected subspace of a graph. In general the initial distribution could span over all connected components. In such scenario the stationary distribution is the uniform distribution over all vertices.

Chapter 2

Diffusion Map

Diffusion map exploits random walks on a graph to find lower-dimensional data embedding. Each dataset is assigned a weighted graph, which defines data-associated transition matrix. Data that is nearby each other in the feature space shall live in the same well-connected component of a graph. By initializing random walkers on such graph, the nearby data points in the feature space will also have a nearby distance in the diffusion space. We now define diffusion maps for nonlinear dimensionality reduction.

2.1 Basic Definition of Graph

A graph \mathcal{G} is a set of vertices and edges which connect any two vertices, which in general can also connect the same vertex. From this definition we can start analyzing the property of a graph. The total weighted edges connected to each vertices can be represented by the *degree matrix* whereas the *adjacency matrix* accounts for the weighted between any two vertices.

The degree matrix is the diagonal matrix in which each diagonal element represents the total weighted edges at the associated vertex. For the adjacency matrix \mathbf{A} , we define its A_{ij} element as the weight of the edge connecting vertex j to vertex i . Hence, the degree of each vertex is the summation of the weighted edges connected to that vertex. The graph in which every vertex is connected to every other vertices is called a complete graph.

2.2 Random Walk on Graph

We now construct a graph associated to a given dataset, then define a random walk on it. Let $X = \left\{ \mathbf{x}^{(i)} \right\}_{i=1}^N$ be a dataset where $\mathbf{x}^{(i)} \in \mathbb{R}^n$. Also, $N = |X|$ is the size of the dataset. Associate a vertex i of a graph to $\mathbf{x}^{(i)}$, then associate a weighted edge by the Gaussian kernel connecting the two vertices as

$$W_{ij} = \exp \left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{\sigma} \right), \quad (2.1)$$

where σ is a global parameter to be chosen. The weight between the data points that are faraway in the original feature space (large euclidean distance) compared to σ will be exponentially suppressed. If σ is too large, we will have a complete graph but that's due to a poor choice of σ . We'll discuss how to choose σ later.

From this construction of data-associated graph, we have the adjacency matrix to be $\mathbf{W} = (W_{ij})$. Now we can define the transition probability from vertex j to i of a random walker on this graph. Accounting for normalization, the transition probability of the random walker to move from vertex j to vertex i is

$$P_{ij} = \frac{W_{ij}}{\sum_i W_{ij}}. \quad (2.2)$$

Denote the transition matrix of this graph as $\mathbf{P} = (P_{ij})$, which can be rewritten in terms of the adjacency matrix \mathbf{W} and the degree matrix \mathbf{D} as

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}, \quad (2.3)$$

where $\mathbf{D} = \text{diag}\{d_1, \dots, d_N\}$ and $d_i = \sum_j W_{ij}$.

2.3 Diffusion Map

The diffusion map ϕ projects each data $\mathbf{x}^{(i)} \in \mathbb{R}^n$ of the original feature space into a lower-dimensional embedding \mathbb{R}^k called the *diffusion space*,

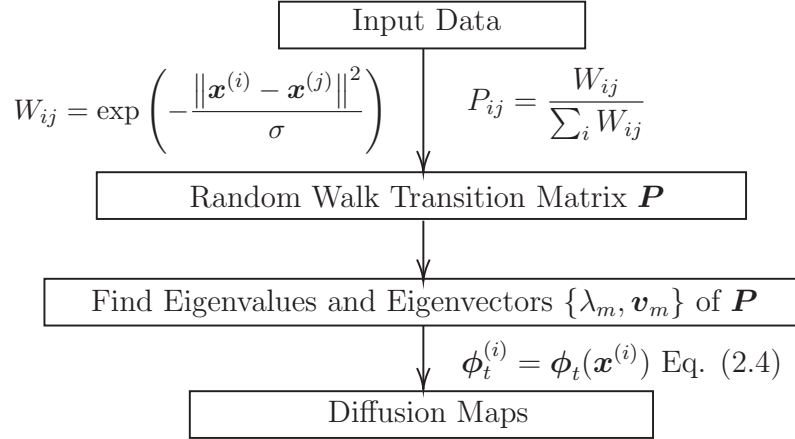


Figure 2.1: Classical algorithm for diffusion maps computation

where typically $k \ll n$. The time-dependent coordinate in the diffusion space of data i , once time has passed for t steps, is defined as

$$\phi_t^{(i)} = \phi_t(\mathbf{x}^{(i)}) = \begin{pmatrix} \lambda_1^t v_1^{(i)} \\ \vdots \\ \lambda_k^t v_k^{(i)} \end{pmatrix}, \quad (2.4)$$

where $v_j^{(i)}$ is the i^{th} -component of the eigenvector \mathbf{v}_j of the transition matrix \mathbf{P} with eigenvalue λ_j , where $1 = \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N \geq 0$. Here the lower-dimensional embedding depends on the number of the first k eigenvectors (first k largest eigenvalue), which, in the long time limit, only the eigenvector with eigenvalue close to 1 survive. So the number of non-zero components will in general be such that $k \ll n$ at long times, roughly corresponding to the number of clusters in the dataset. However, sometimes it is useful also to keep t short to discover the geometric structure of the lower dimensional manifold on which the data lies, as we will see in the implementation section. Note that the first coordinate of the diffusion map is always 1 as $\mathbf{1}$ is the trivial eigenvector with $\lambda_1 = 1$. Hence, the first coordinate of (2.4) is less informative since it's identically 1 for every data point $\mathbf{x}^{(i)}$.

Now we explain *diffusion distance* in the diffusion space. The notion of proximity between data points on a graph can be defined based on how fast random walks starting at each data points can visit each other. Intuitively, two points that are connected by multiple paths on a graph should be near in the diffusion space; whereas two points that do not

have many paths connecting them should lie far from each other in the diffusion space. This notion of diffusion distance can be made more precise by the definition

$$\text{Dis}_t^2(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sum_{\mathbf{y} \in X} \left(p_t(\mathbf{y}, \mathbf{x}^{(i)}) - p_t(\mathbf{y}, \mathbf{x}^{(j)}) \right)^2 / \phi_1(\mathbf{y}), \quad (2.5)$$

where $p_t(\mathbf{y}, \mathbf{x}^{(i)})$ is the probability distribution of a random walker starting at $\mathbf{x}^{(i)}$ to be found at an intermediate vertex \mathbf{y} at time t , and the normalization $\phi_1(\mathbf{y})$ is the trivial component of the diffusion map of \mathbf{y} constructed from the trivial eigenvector of the transition matrix, i.e.

$$\phi_1(\mathbf{y}) = \frac{d_{\mathbf{y}}}{\sum_z d_z}. \quad (2.6)$$

From (2.5), the two data points that are well-connected on a graph will have a large overlap between their diffusive-spreading distributions even when t is small, encapsulating that the two points are nearby in the diffusion space. On the other hand, the two data points that are rarely connected on a graph will have a relatively small overlap, if at all, between their diffusive-spreading distributions. These observations illustrate the notion of proximity in the diffusion space of (2.5). In fact, as shown in [9], the diffusion distance between any two data points is a proper distance; it is the Euclidean distance between the diffusion maps in the lower-dimensional embedding (diffusion space):

$$\text{Dis}_t^2(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \|\phi_t^{(i)} - \phi_t^{(j)}\|^2. \quad (2.7)$$

2.4 Properties of the Eigenspace of the Transition Matrix

This section study the properties of the eigenvectors of the transition matrix. Because the transition matrix is not necessarily symmetric, its eigenvectors are not necessarily orthogonal. However, we can construct a symmetric representation via the following similarity transformation from a symmetric \mathbf{D}^{-1} and a symmetric weight \mathbf{W} :

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W} = \mathbf{D}^{-1/2}\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{D}^{1/2} = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{1/2}, \quad (2.8)$$

where

$$\mathbf{S} \equiv \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2} \quad (2.9)$$

is symmetric. Thus we can choose the eigenvectors of \mathbf{S} to be a complete real orthonormal basis.

Now, we show that the eigenvalues of \mathbf{S} are also the eigenvalues of \mathbf{P} and the eigenvectors are related by the degree matrix. Consider the eigenvalue equation of \mathbf{S} , i.e.,

$$\mathbf{S} \mathbf{u}_i = \lambda_i \mathbf{u}_i. \quad (2.10)$$

Since $\mathbf{S} = \mathbf{D}^{1/2} \mathbf{P} \mathbf{D}^{-1/2}$, it follows that

$$\mathbf{P} \mathbf{D}^{-1/2} \mathbf{u}_i = \lambda_i \mathbf{D}^{-1/2} \mathbf{u}_i. \quad (2.11)$$

Hence, the eigenvector of \mathbf{P} are related to the eigenvector of \mathbf{S} by

$$\mathbf{v}_i = \mathbf{D}^{-1/2} \mathbf{u}_i, \quad (2.12)$$

with the same eigenvalue λ_i . Because $\mathbf{D}^{-1/2}$ is real, and \mathbf{u}_i is also real, the eigenvector \mathbf{v}_i of \mathbf{P} is also real. In addition, because $\{\mathbf{u}_i\}$ is an orthonormal set, i.e., $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$, it follows that the eigenvectors of the transition matrix \mathbf{P} satisfy

$$\mathbf{v}_i^T \mathbf{D} \mathbf{v}_j = \delta_{ij}. \quad (2.13)$$

In the implementation section, we will first compute the eigenvectors of \mathbf{S} since numerical methods for finding the eigenvectors of symmetric matrix are fast. Then we can transform the eigenvectors according to (2.12) to obtain the corresponding eigenvectors of the transition matrix \mathbf{P} .

2.5 Random Walk and Heat Diffusion

Here we discuss how random walk on discrete states, such as on a graph, is related to standard heat diffusion in a continuous space. Consider the following one-dimensional lattice, where each lattice point is separated by a distance d . Suppose the transition probability to hop to the neighbors is $P < 1/2$. The probability of a random walker at the time

$t + 1$ to be at the position x can be written as the sum of the transition probabilities into the position x from the current time step t as,

$$p(x, t + 1) = P_{x,x}p(x, t) + P_{x,x+d}p(x + d, t) + P_{x,x-d}p(x - d, t). \quad (2.14)$$

For $d \ll 1$, we Taylor's expand

$$\begin{aligned} p(x, t+1) = & P_{x,x}p(x, t) + P_{x,x+d} \left(p(x, t) + d\partial_x p(x, t) + \frac{d^2}{2!}\partial_x^2 p(x, t) + \dots \right) \\ & + P_{x,x-d} \left(p(x, t) - d\partial_x p(x, t) + \frac{d^2}{2!}\partial_x^2 p(x, t) + \dots \right). \end{aligned} \quad (2.15)$$

Substituting $P_{x,x+d} = P_{x,x-d} = P$ and $P_{x,x} = 1 - 2P$ (conservation of probability) gives

$$\begin{aligned} p(x, t + 1) & \approx (1 - 2P)p(x, t) + 2Pp(x, t) + 2P\frac{d^2}{2!}\partial_x^2 p(x, t) \\ & = p(x, t) + \alpha\partial_x^2 p(x, t), \end{aligned} \quad (2.16)$$

where $\alpha = Pd^2$. If we promote the transition matrix \mathbf{P} to a continuous space representation, then we have

$$p(x, t + 1) = \mathbf{P}p(x, t) = p(x, t) + \alpha\partial_x^2 p(x, t). \quad (2.17)$$

We can thus write

$$-(\mathbf{I} - \mathbf{P})p(x, t) = \alpha\partial_x^2 p(x, t). \quad (2.18)$$

Therefore, $-(\mathbf{I} - \mathbf{P})$ is generally called a *Laplacian*, which can be generalized to a hypercubic lattice, where the second derivative becomes ∇^2 . In addition, the Laplacian operator can be generalized to the Laplace-Beltrami operator which can be defined on a curved space or manifold where a metric is well-defined [9,11]. In the context of discrete states such as random walk on graphs, $-(\mathbf{I} - \mathbf{P})$ is called the normalized *graph Laplacian* matrix, where both \mathbf{I} and \mathbf{P} are both well-defined on a graph.

In the continuous-time limit, we may write (2.17) and (2.18) as

$$\partial_t p(x, t) = \alpha\partial_x^2 p(x, t). \quad (2.19)$$

For discrete states, the above can thus be written in terms of the normalized graph Laplacian matrix as

$$\partial_t \mathbf{p}(t) = -(\mathbf{I} - \mathbf{P})\mathbf{p}(t). \quad (2.20)$$

The last two equalities show the relationship between heat diffusion on a continuous space and random walks on a graph. More generally, the transition probability \mathbf{P} on a graph does not need to be local, unlike the standard nearest neighbor hopping random walk in space defined at the beginning of this section.

2.6 Implementation of Diffusion Map

This section shows numerical results of Algorithm 1, implementing manifold learning and non-linear dimensionality reduction via diffusion maps in some illustrative dataset.

Algorithm 1 Diffusion Map

- calculate Euclidean distance between every data points
 - calculate the global σ associated with the dataset according to (2.21)
 - calculate the weight matrix \mathbf{W} according to (2.1)
 - calculate the degree matrix \mathbf{D}
 - calculate the symmetric matrix \mathbf{S} from (2.9)
 - find the eigenvalues $\{\lambda_i\}$ and eigenvectors $\{\mathbf{u}_i\}$ of \mathbf{S}
 - we may pick the first k largest eigenvalues.
 - calculate the eigenvectors of \mathbf{P} : $\mathbf{v}_i = \mathbf{D}^{-1/2}\mathbf{u}_i$
 - calculate Diffusion Map for each data point according to (2.4).
 - For manifold learning, take time-step t to be relatively small
 - For data clustering, take time-step $t \gg 1$.
-

Diffusion Map for Manifold Learning

Consider the simplest toy dataset consisting of finitely many points on a one-dimensional line as in fig. 2.2 (left), we found that the components of the diffusion map are the eigenfunctions of the laplacian operator (∂_x^2 in this case). Here we take diffusion time $t = 1$. As shown in fig. 2.3, the second component is the only component with a one-to-one mapping between the dataset and their coordinates in the diffusion space. Thus, when assigning color codes to denote the magnitude of the second component of in the diffusion space, we can see that nearby

points in the original feature space (fig. 2.2 (left)) are mapped into nearby points in the diffusion space (fig. 2.2 (right)).

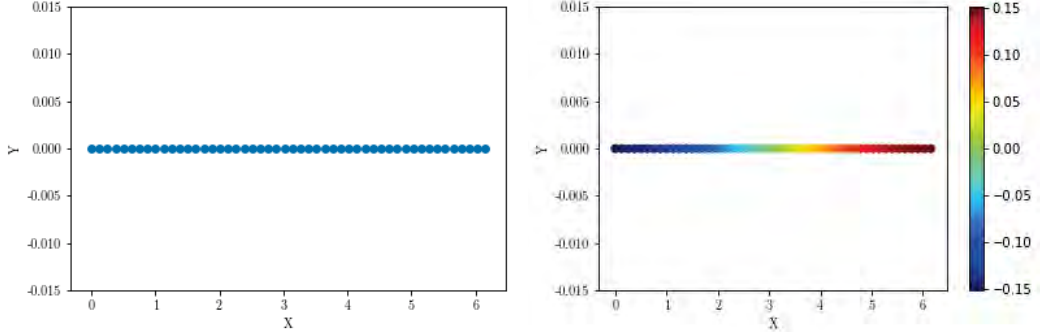


Figure 2.2: (Left) A finite one-dimensional lattice as a toy dataset to test the manifold learning algorithm. (Right) The dataset colored by the second component of the diffusion map, showing that nearby points in the original space is mapped to nearby points in the diffusion space (second component).

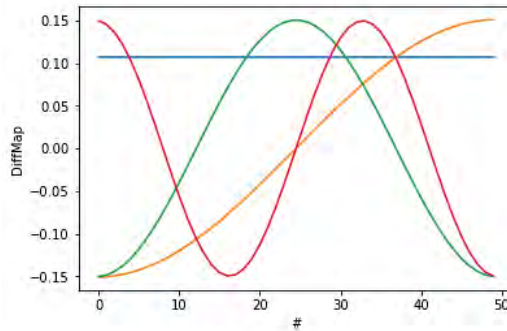


Figure 2.3: First four components of the diffusion map (2.4) with diffusion time $t = 1$ (vertical axis) versus the location along the one dimensional line of each data point (horizontal axis). The color codes are 1st (Blue), 2nd (Orange), 3rd (Green), and 4th (Red) components. Only the second component has a one-to-one mapping between the dataset and their coordinates in the diffusion space.

Next, we analyze a *two-dimensional non-linear* dataset. Again we take the diffusion time $t = 1$. Comparing with the standard linear dimensionality reduction method of Principal Component Analysis (PCA), we can see in fig. 2.4 that diffusion map can automatically discover the notion of the proximity between the data lying on a one-dimensional manifold embedded in a two-dimensional space correctly whereas PCA fails to do so! The second component of the diffusion map again shows a one-to-one correspondence between the dataset and the value in the

diffusion space, and PCA again fails to do so. From numerical explorations, it appears that, for a one-dimensional manifold embedded in a two dimensional space, the second component of the diffusion map seems to be the best component to quantify the notion of distance on the lower-dimensional embedding. These numerical experiments verify that diffusion map can perform manifold learning and non-linear dimensionality reduction, at least in \mathbb{R}^2 .

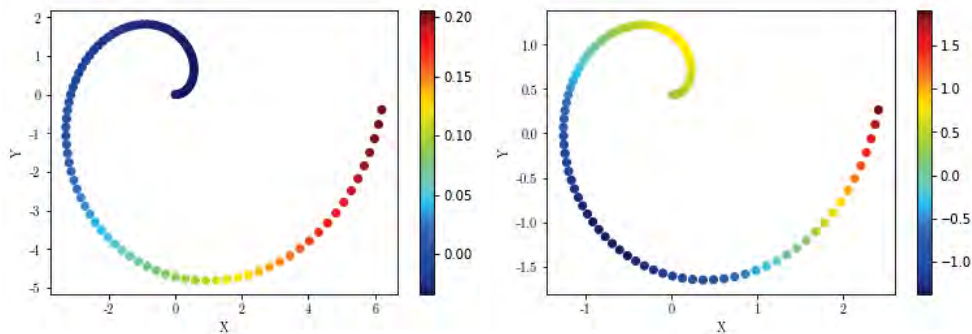


Figure 2.4: A non-linear one-dimensional data manifold embedded in a two dimensional space. The dataset is colored by (left) the second component of the diffusion map, and (right) the first component of PCA. Clearly, diffusion map discovers the notion of “proximity along the lower-dimensional non-linear data manifold” while PCA fails to do so!

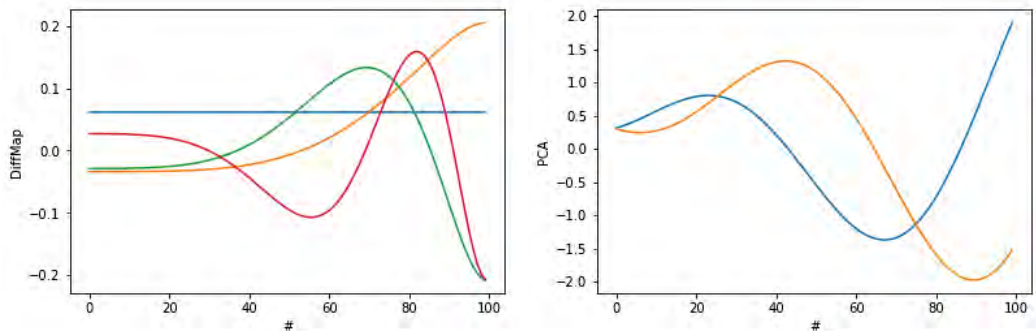


Figure 2.5: (Left) First four components of the diffusion map with diffusion time $t = 1$ (vertical) of each data point index (horizontal) corresponding to the color blue, orange, green, red, respectively, for dataset in fig. 2.4. Again only the second component shows a one-to-one mapping between the data points and the coordinate in the diffusion space. (Right) The first two principal components of PCA (vertical) for each data point index (horizontal)

Next, we consider slightly more complex manifolds. These are low-dimensional embeddings in 3 dimensional space. Namely, the datasets

consisting of point clouds that look like a toroidal helix and a Swiss roll, see fig. 2.6. After we project the data into the first two components of the diffusion map with the diffusion time $t = 1$, the result in fig. 2.7 show that the notion of proximity in the original feature space is preserved. In addition, the first two diffusion components also sketch the flattened version of the original non-linear manifold correctly.

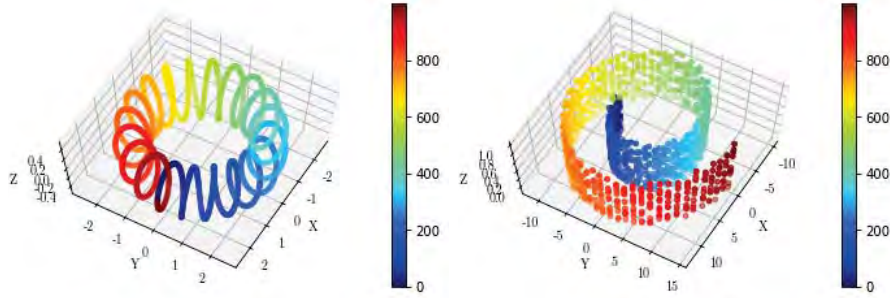


Figure 2.6: (Left) A toroidal helix, and (Right) a Swiss roll. Color code label different data points. Nearby colors label nearby points on the lower-dimensional embedding.

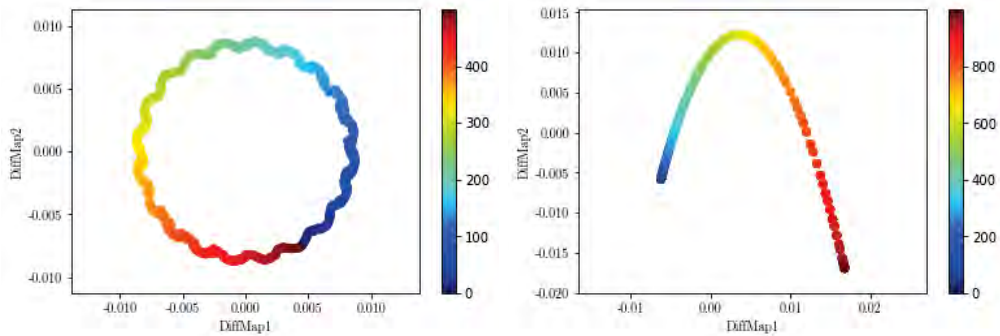


Figure 2.7: Data points projected onto the first two components of the diffusion map with the diffusion time $t = 1$ of (Left) a toroidal helix, and (Right) a Swiss roll. One can see that the notion of proximity in the original space is preserved (color variation) and that the shape reflect the flattened out version of the original non-linear data manifolds. Interestingly, for the diffusion map of the swiss roll, the map appears to discover that the two-dimensional roll can be generated from a one-dimensional line, with extra layers in the z direction.

Diffusion Map for Data Clustering

For manifold learning, we took the diffusion time $t = 1$. For data clustering; however, we shall take the diffusion time t to be sufficiently large such that the distributions initialized at the data points in the same

well-connected component of a graph almost reach the same stationary distribution, corresponding to the uniform distribution over all the vertices in the same connected component (cluster) of the graph. On the other hand, distributions initialized on data points belonging to a different cluster will converge toward a different stationary distribution.

We first need to set the scale of proximity in the original feature space σ of the weight matrix (2.1). If σ is too large, the data-induced graph will be a complete graph, and every data point will belong to the same cluster. If σ is too small, every point can constitute its own cluster. Ref [8] proposes a method to pick the scale σ by taking the average of the minimum Euclidean norm between each data point and its neighbors, i.e.,

$$\sigma = \frac{1}{N} \sum_{i \in X} \min_{j \sim i} \left\| \mathbf{x}^{(i)} - \mathbf{x}^{(j)} \right\|^2. \quad (2.21)$$

In this way, the mean distance will set the global scale for the notion of “neighbors” in a dataset. Two data points that do not live in the same Euclidean ball of radius $\sqrt{\sigma}$ are considered far apart as their weights will be exponentially suppressed. To test the algorithm, we perform clustering of two doughnut-shape data clouds with different point density where each doughnut lives on a different plane in the z direction. Fig. 2.8 shows the data clouds projected onto a two dimensional plane. Using (2.21), we find that the transition matrix can be written as a block diagonal form, and its eigenvectors satisfy (1.18).

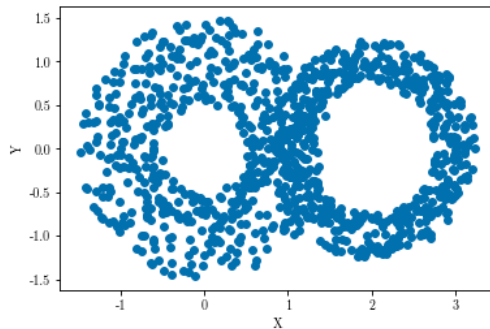


Figure 2.8: Non-overlapping doughnut-shape data clouds projected onto two dimensions (each doughnut lives on a different z value). Note that the right cluster is more dense than the left.

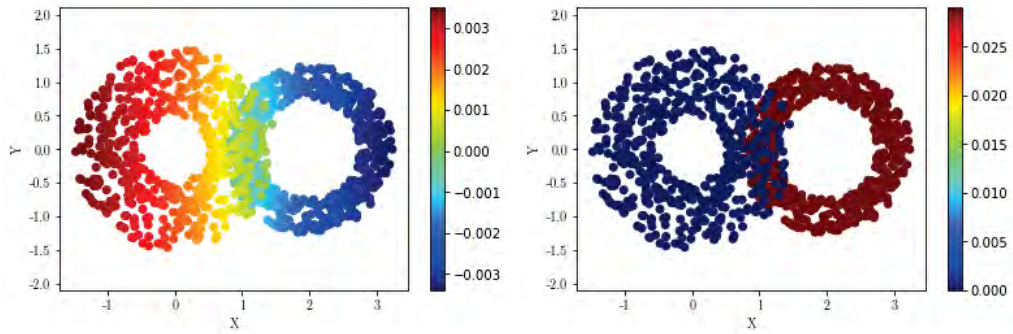


Figure 2.9: (Left) Data points labeled by the second component of the diffusion map by picking a small σ , but not according to (2.21). However, by setting the scale of σ according to (2.21), the second component can identify the correct clusters within diffusion time $t = 1$! (Right). This is because the density of points on the right cluster is higher than the left, and the two clusters are in fact quite separated in the z directions. By picking the global parameter σ according to (2.21), the second component of the diffusion map with $t = 1$ can separate the two clusters by their associated density of points.

Now we apply diffusion map for clustering a high-dimensional real-world dataset from [18]. The dataset consists of 178 data samples of wines grown in the same region in Italy, but derived from three different cultivars. Each data sample consists of 13 quantities (features) extracted from chemical analysis of wines. Using only the first two components of the diffusion map, the result shown in fig. 2.10 (Right) reveals the distinction between the 3 classes of wines almost perfectly. In addition, the two dimensional diffusion map embedding also reflects proximity between different classes of cultivars. Namely, one cultivar (green) are closely related to the other two (blue, and red), whereas the other two are very distinct types. This unsupervised learning algorithm leads to the discovery of knowledge hidden in 13-dimensional chemical components space concerning the relationship between cultivars, which can help suggest scientists to further investigate the causes of such relationship.

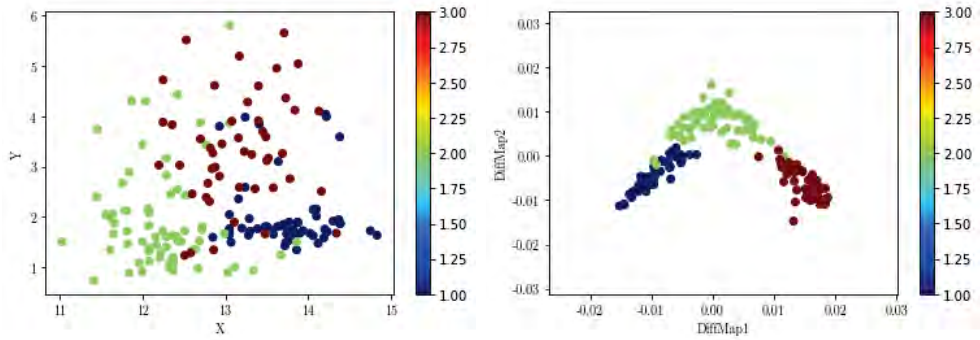


Figure 2.10: Diffusion map automatically discovers the relationship between three cultivars of wines grown in the same region of Italy. The dataset from [18] consists of 178 data samples, each consists of 13 quantities (features) extracted from the chemical analysis of wines. (Left) Projection of the dataset onto 2 (of 13)-dimensional feature space. Different colors encode different cultivars of wines. (Right) Projection of the dataset onto the second and the third component of the diffusion map shows almost perfect identification of the three cultivars. In addition, diffusion map also reveals the relationship between the three cultivars of wines. Namely, the green type is closely related to the other two.

Chapter 3

Quantum Algorithm for Diffusion Map

Recent advances in quantum information science provide promising prospects for useful near-term quantum computation devices. As such, many quantum algorithms of classical Machine Learning have been proposed to offer quantum computational speedup over its classical counterparts. For example, routines in Machine Learning algorithms that involve finding eigenvalues and eigenvectors can benefit from quantum computational speedup from the Quantum Phase Estimation (QPE) algorithm, which is specifically designed to find the eigenvalues and eigenvectors of a unitary operator. Classically, the number of operations required to find eigenvalues and eigenvectors of a matrix of size $N \times N$ is $O(N^3)$; however, QPE requires $O(N^2)$ operations [3].

In this chapter, by combining useful quantum computational tricks, we will propose a quantum algorithm for classical diffusion map. Specifically, by harnessing quantum computational speed up of QPE in finding eigensubspaces, using qRAM data compression during classical data encoding, and employing coherent state tricks to efficiently compute the weight matrix, we propose a *Quantum Diffusion Map* for nonlinear dimensionality reduction, which, to the best of our knowledge, has not been studied.

3.1 Encoding Data into Quantum Random Access Memory

For classical data $X = \{\mathbf{x}^{(i)}\}_{i=1}^N$, ref. [4] proposed a method to compress the classical dataset into qRAM via the identification $\mathbf{x}^{(i)} \rightarrow |\mathbf{x}^{(i)}\rangle$. A natural way to encode classical data into qRAM is using bit strings representation. For d features, this encoding scheme requires $n = \log_2(d)$ qubits to represent one data point by the identification

$$|\mathbf{x}^{(i)}\rangle = \frac{1}{\|\mathbf{x}^{(i)}\|} \sum_{p=0}^{2^n-1} x_p^{(i)} |p\rangle, \quad (3.1)$$

where p is a base-2 number and $|p\rangle = \bigotimes_{m=0}^{n-1} |p_m\rangle$, where $|p_m\rangle \in \{(|\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\rangle), (|\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\rangle)\}$ as an m^{th} qubit representation.

However, a more efficient way to encode the data is to define the basis on qRAM using the coherent state as

$$|\mathbf{x}^{(i)}\rangle = \bigotimes_{p=0}^{2^n-1} |x_p^{(i)}\rangle, \quad (3.2)$$

where $|x_p^{(i)}\rangle$ is the canonical coherent state representing the p^{th} feature of data $\mathbf{x}^{(i)}$, i.e.,

$$|x_p^{(i)}\rangle = \exp\left(-\frac{(x_p^{(i)})^2}{2}\right) \sum_{n=0}^{\infty} \frac{(x_p^{(i)})^n}{\sqrt{n!}} |n\rangle, \quad (3.3)$$

where $|n\rangle$ is an n^{th} eigenstate of a harmonic oscillator.

3.2 Constructing Transition Matrix from Coherent States

Consider an inner product of two canonical coherent states of two data points

$$\begin{aligned}
\langle x_p^{(i)} | x_p^{(j)} \rangle &= \exp\left(-\frac{(x_p^{(i)})^2}{2}\right) \exp\left(-\frac{(x_p^{(j)})^2}{2}\right) \sum_{m,n=0}^{\infty} \frac{(x_p^{(i)})^m}{\sqrt{m!}} \frac{(x_p^{(j)})^n}{\sqrt{n!}} \langle m|n \rangle \\
&= \exp\left(-\frac{(x_p^{(i)})^2 + (x_p^{(j)})^2}{2}\right) \sum_{n=0}^{\infty} \frac{(x_p^{(i)} x_p^{(j)})^n}{n!} \\
&= \exp\left(-\frac{(x_p^{(i)})^2 + (x_p^{(j)})^2}{2}\right) \exp(x_p^{(i)} x_p^{(j)}) = \exp\left(-\frac{(x_p^{(i)} - x_p^{(j)})^2}{2}\right).
\end{aligned} \tag{3.4}$$

Hence, the inner product of the two data points in qRAM is the Gaussian kernel

$$\begin{aligned}
\langle \mathbf{x}^{(i)} | \mathbf{x}^{(j)} \rangle &= \prod_{p=0}^{2^n-1} \langle x_p^{(i)} | x_p^{(j)} \rangle \\
&= \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2}\right) = K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}).
\end{aligned} \tag{3.5}$$

Here, we take $\sigma = 1$ for simplicity (though σ can be incorporated by a scaling factor during state encoding). Next, we exploit quantum parallelism in the auxiliary Hilbert space to label the state of the position Hilbert space,

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \otimes |\mathbf{x}^{(i)}\rangle \in \mathcal{H} = \mathcal{H}_{\text{aux}} \otimes \mathcal{H}_{\text{position}}. \tag{3.6}$$

In this case, the density matrix is

$$\hat{\rho} = \frac{1}{N} \sum_{i,j=0}^{N-1} |i\rangle\langle j| \otimes |\mathbf{x}^{(i)}\rangle\langle \mathbf{x}^{(j)}|. \tag{3.7}$$

Taking a partial trace of the position space yields

$$\text{Tr}_{\text{position}}(\hat{\rho}) = \frac{1}{N} \sum_{i,j=0}^{N-1} \langle \mathbf{x}^{(i)} | \mathbf{x}^{(j)} \rangle |i\rangle\langle j| = \frac{1}{N} \sum_{i,j=0}^{N-1} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) |i\rangle\langle j| = \frac{\hat{\mathbf{K}}}{N}, \quad (3.8)$$

and

$$\text{Tr}(\hat{\mathbf{K}}) = \sum_{i=0}^{N-1} K(\mathbf{x}^{(i)}, \mathbf{x}^{(i)}) = N. \quad (3.9)$$

Now, we have the kernel operator, which allows us to construct the degree matrix $\hat{\mathbf{D}} = \sum_{i=0}^{N-1} D_i |i\rangle\langle i|$ where $D_i = \sum_j K_{ij}$. Finally, from the coherent state representation of classical data, the transition matrix can be constructed as $\hat{\mathbf{P}} = \hat{\mathbf{D}}^{-1} \hat{\mathbf{K}} = \sum_{i,j=0}^{N-1} P_{ij} |i\rangle\langle j|$, identical to (2.2).

However, since $\hat{\mathbf{P}}$ is not necessarily Hermitian, we can not find its eigenvalues and eigenvectors of $\hat{\mathbf{P}}$ directly from Quantum Phase Estimation (QPE). However, this problem can be alleviated via the following similarity transformation:

$$\hat{\mathbf{P}} = \hat{\mathbf{D}}^{-1} \hat{\mathbf{K}} = \hat{\mathbf{D}}^{-1/2} \hat{\mathbf{D}}^{-1/2} \hat{\mathbf{K}} \hat{\mathbf{D}}^{-1/2} \hat{\mathbf{D}}^{1/2} = \hat{\mathbf{D}}^{-1/2} \hat{\mathbf{S}} \hat{\mathbf{D}}^{1/2}, \quad (3.10)$$

where $\hat{\mathbf{S}} \equiv \hat{\mathbf{D}}^{-1/2} \hat{\mathbf{K}} \hat{\mathbf{D}}^{-1/2}$. Since $\hat{\mathbf{K}}$ is Hermitian, $\hat{\mathbf{S}}$ is also Hermitian. To achieve quantum computational speedup, we will adopt QPE to first find eigenvalues and eigenvectors of $\hat{\mathbf{S}}$. Then, the eigenvalues and the eigenvectors of $\hat{\mathbf{S}}$ and $\hat{\mathbf{P}}$ are related as follows. Consider the eigenvalue equation for $\hat{\mathbf{P}}$:

$$\hat{\mathbf{P}} |v_j\rangle = \lambda_j |v_j\rangle. \quad (3.11)$$

Since $\hat{\mathbf{P}}$ can be rewritten as $\hat{\mathbf{D}}^{-1/2} \hat{\mathbf{S}} \hat{\mathbf{D}}^{1/2}$, we have

$$\hat{\mathbf{S}} \hat{\mathbf{D}}^{1/2} |v_j\rangle = \lambda_j \hat{\mathbf{D}}^{1/2} |v_j\rangle. \quad (3.12)$$

Therefore, we can find the eigenvectors of $\hat{\mathbf{P}}$ with eigenvalue λ_j from the eigenvectors of $\hat{\mathbf{S}}$ via the mapping

$$|v_j\rangle = \hat{\mathbf{D}}^{-1/2} |u_j\rangle, \quad (3.13)$$

where $|u_j\rangle$ is an eigenvector of $\hat{\mathbf{S}}$ with eigenvalue λ_j .

3.3 Quantum Algorithms to Find Eigenvalues and Eigenvectors

We now have a Hermitian operator $\hat{\mathbf{S}}$, which we can then employ the QPE to find the eigenvalues, see Appendix B. The result of QPE applying to an input state $|\psi_0\rangle = \sum_j c_j |u_j\rangle$ is

$$|0\rangle |\psi_0\rangle = \sum_j c_j |0\rangle |u_j\rangle \xrightarrow{\text{QPE}} |\psi_1\rangle = \sum_j c_j |\tilde{\lambda}_j\rangle |u_j\rangle, \quad (3.14)$$

where $|\tilde{\lambda}_j\rangle$ is the estimated binary representation of the eigenvalue of $\hat{\mathbf{S}}$. Now, we have the superposition of the eigenvectors of $\hat{\mathbf{S}}$ with their estimated eigenvalues. The eigenvectors of the transition matrix $\hat{\mathbf{P}}$ can then be computed from (3.13). Now we can extract the eigenvalues and the eigenvectors from the measurement operator, see the details in Appendix B. Finally, the diffusion map of (2.4) can be readily constructed from the eigenvalues and the eigenvectors. We summarize the quantum algorithm for *Quantum Diffusion Map* in the diagram of fig. 3.1 below.

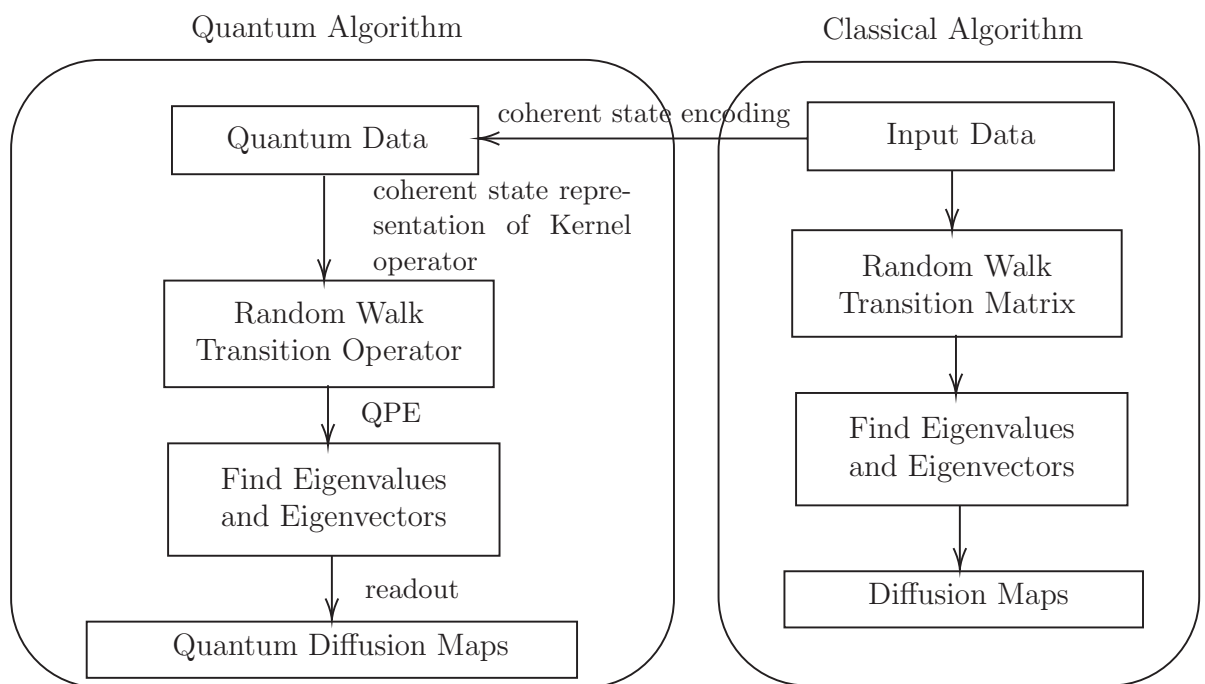


Figure 3.1: Comparison between classical and quantum diffusion map.

Chapter 4

Quantum Walk for Data Clustering

In the previous chapters, manifold learning is achieved via finding eigenvalues and eigenvectors of a dataset-associated transition matrix. This chapter will explore the possibility to perform random walk on graph (obeying rules from dataset-associated transition matrix) for manifold learning, clustering in particular. The motivation here is to avoid the computation of eigenvalues and eigenvectors, which even with a quantum computational speedup of quantum diffusion map, requires at least $O(N^2)$ operations.

We will explore the usage of quantum walk for cluster identifications. Motivated by the fact that the width of the distribution of a quantum walk on a one-dimensional lattice grows as t instead of \sqrt{t} as in a classical random walk [17], we investigate how we might implement quantum walk on a graph for data clustering here. We focus on a class of *discrete-time coined quantum walk* on a graph. A quantum state in a coined quantum walk model has two components: the spatial Hilbert space which keeps track of the state of the walker, and the coin Hilbert space which stores the information about transition matrix to other vertices in the next time evolution. In other words, the coin is more like a dice with the number of faces equal to the number of vertices in a graph of interest.

For a graph with N vertices, let $|i\rangle$ denote the spatial state i of a quantum walker in an N -dimensional position Hilbert space \mathcal{H}_P and $|c\rangle$ denote the coin state which lives in an N -dimensional coin Hilbert space \mathcal{H}_C . A state of a coined-quantum walker on the i th vertex on a

graph is a tensor product state:

$$|\psi\rangle = |i\rangle \otimes |c\rangle \equiv |i, c\rangle. \quad (4.1)$$

To develop time-evolution of a quantum walker on any vertex i , a common approach is to attach the weighted graph (2.2) by the correct normalization of the coined state

$$|\psi_i\rangle = |i\rangle \otimes \sum_{c \sim i} \sqrt{P_{ci}} |c\rangle \equiv |i\rangle \otimes |\phi_i\rangle, \quad (4.2)$$

where the transition probability is obtained from (2.2). The coin state attached to the vertex $|i\rangle$ tells which vertices the vertex i can transit to in the next time-step, with the associated transition probabilities.

To perform one-step unitary time-evolution, one can apply two operators in succession: the coin operator $\hat{\mathbf{C}}$ and the shift operator $\hat{\mathbf{S}}$. The coin operator is defined as

$$\hat{\mathbf{C}} = \sum_{i \in X} |i\rangle\langle i| \otimes \left(2|\phi_i\rangle\langle\phi_i| - \hat{\mathbf{I}} \right), \quad (4.3)$$

where the operator on the coin Hilbert space is the Grover's diffusion operator, used ubiquitously in search algorithms,

$$\hat{\mathbf{G}} = 2 \sum_{i \in X} |\phi_i\rangle\langle\phi_i| - \hat{\mathbf{I}}. \quad (4.4)$$

The shift operator swaps the vertices between the coin Hilbert space and the position Hilbert space

$$\hat{\mathbf{S}} = \sum_{i, j \in X} |j, i\rangle\langle i, j|. \quad (4.5)$$

Finally, the one-step unitary time-evolution operator is

$$\hat{\mathbf{U}} = \hat{\mathbf{S}}\hat{\mathbf{C}}, \quad (4.6)$$

which is a huge matrix of size $N^2 \times N^2$. The quantum state of a quantum walker at time t is then

$$|\psi(t)\rangle = \hat{\mathbf{U}}^t |\psi(0)\rangle. \quad (4.7)$$

As a result, the probability of a quantum walker to be on a vertex i at time t is

$$p_t(i) = \sum_{j \in X} |\langle i, j | \psi(t) \rangle|^2, \quad (4.8)$$

which can be shown to satisfy the conservation of probability condition

$$\sum_{i \in X} p_t(i) = 1. \quad (4.9)$$

4.1 Limiting Distribution of a Quantum Walk

One stark difference between classical random walk and quantum walk is that the stationary state may not exist in quantum walk. Namely, $\lim_{t \rightarrow \infty} |\psi(t)\rangle$ might not exist because

$$\begin{aligned} \|\psi(t+1)\rangle - |\psi(t)\rangle\|^2 &= \left| \hat{U}^t (\hat{U} - \hat{I}) |\psi(0)\rangle \right|^2 \\ &= \langle \psi(0) | (\hat{U}^\dagger - \hat{I}) \hat{U}^{t\dagger} \hat{U}^t (\hat{U} - \hat{I}) | \psi(0) \rangle \\ &= 2 - 2 \operatorname{Re} \left\{ \langle \psi(0) | \hat{U} | \psi(0) \rangle \right\}, \end{aligned} \quad (4.10)$$

which is in general non-zero for most graph structures. Alternatively, one should instead consider the time-average probability at each vertex i

$$\bar{p}_T(i) = \frac{1}{T} \sum_{t=0}^{T-1} p_t(i), \quad (4.11)$$

satisfying

$$\sum_{i \in X} \bar{p}_T(i) = 1. \quad (4.12)$$

It turns out that such time-average on any vertex i converges to the limiting distribution as T goes to infinity [17]. Namely,

$$\pi(i) = \lim_{T \rightarrow \infty} \bar{p}_T(i). \quad (4.13)$$

Fig. 4.1 shows a generic probability on a vertex i as time evolves from numerically simulating a 100-step quantum walk on a toy graph structure. Quantum interference leads to fluctuations in the time-evolved probability; however, the time-average tends to converge to a stationary value.

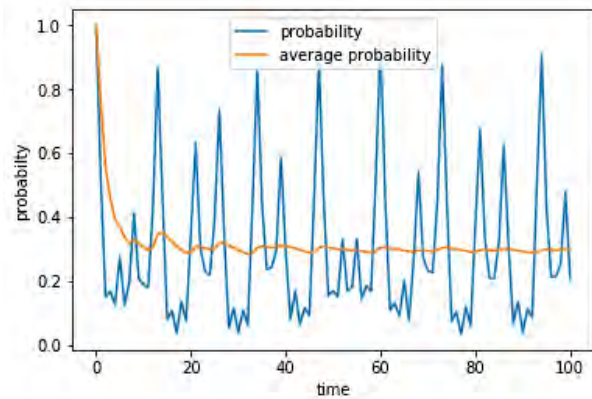


Figure 4.1: Generic behavior of the probability of a quantum walker starting at a vertex i to remain in the state i in the subsequent time steps. The figure shows large fluctuations due to quantum interference in the time-evolved probability; however, the time average tends to converge to a stationary value, verifying (4.13).

4.2 Data Clustering with Coined Quantum Walk

For the transition matrix that is (almost) a block diagonal matrix, we can write

$$P = \begin{pmatrix} P_1 & & \mathcal{O}(e^{-1/\sigma}) & \\ & P_2 & & \\ & & \ddots & \\ \mathcal{O}(e^{-1/\sigma}) & & & P_k \end{pmatrix}. \quad (4.14)$$

In the synthetic dataset shown in Fig. 4.2 consisting of 4 clusters in

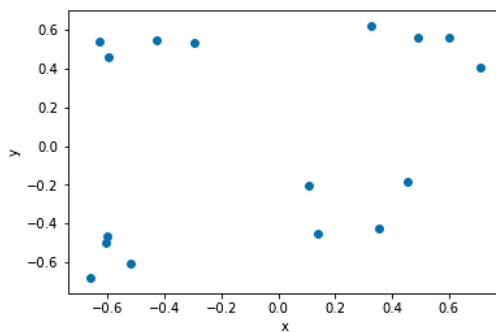


Figure 4.2: A toy dataset consisting of 4 clusters ($k = 4$) on which we will run a coined Quantum walk algorithm for clustering.

two-dimensional space ($k = 4$), we initialize the state of a quantum

walker at

$$|\psi(0)\rangle = |i\rangle \otimes |\phi_i\rangle, \quad (4.15)$$

where $|i\rangle$ represents the vertex of a graph associated with the initial data point, and $|\phi_i\rangle$ is the superposition of the coin states $|\phi_i\rangle = \sum_{c \sim i} \sqrt{P_{ci}} |c\rangle$ with the transition probability P_{ci} computed from (2.2).

We then evolve the state according to (4.7); namely, $|\psi(t)\rangle = (\hat{S}\hat{C})^t |\psi(0)\rangle$. When we compute the time-average probability of every vertices, only vertices that are in the *same* cluster with $|i\rangle$ will converge to a *non-zero* stationary distribution, see Fig. 4.3.

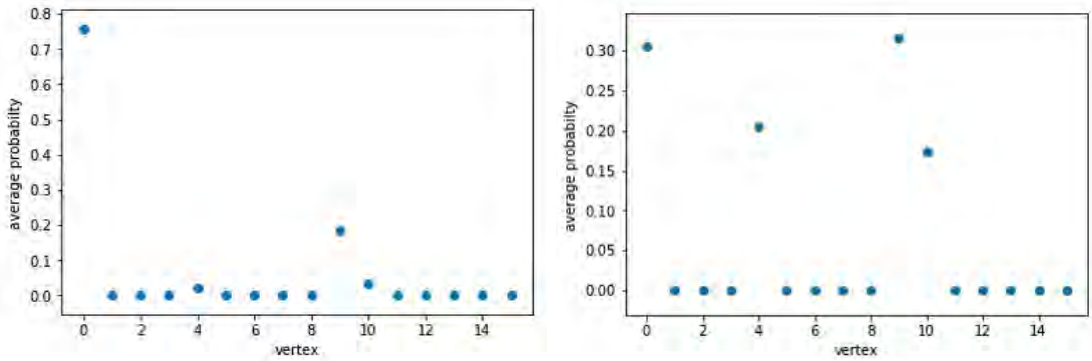


Figure 4.3: The time-average probability of a quantum walker starting at vertex 0 on a weighted graph defined by the synthetic dataset of Fig. 4.2 (Left) at $t = 1$ and (Right) at $t = 10$. Only vertices that lie in the same cluster as the initial vertex gain a non-zero time-average probability!

After we identify all the vertices that lie the same cluster (those with non-zero time-average probability) as the initial vertex, we can reinitialize the coined quantum walk state on other vertices that have not been identified with a cluster. By repeating this procedure until every vertices is identified with a cluster, the clustering scheme from coined quantum walk is complete. Note that reinitialization is required to be done only k times, where k is the number of clusters in a dataset. Fig. 4.4 shows a successful data clustering from the coined quantum walk scheme described above. There, only 4 reinitializations are required (4 clusters).

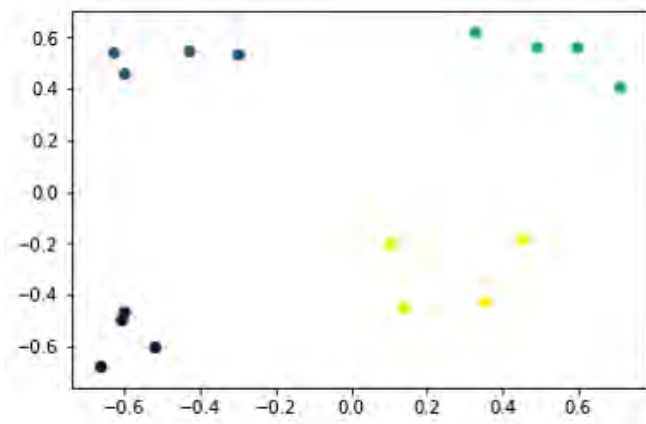


Figure 4.4: Successful data clustering of the dataset in Fig. 4.2 using a coined quantum walk scheme with 4 reinitializations. Different colors identify different clusters.

Chapter 5

Conclusion

In this thesis, we first review classical diffusion map, an unsupervised learning algorithm for non-linear dimensionality reduction and manifold learning. The connection to statistical physics of random walk and heat diffusion on graph is discussed. We also explain the intuition why random walk can lead to the identification of low-dimensional embedding of data structure as well as to automatically discovering data clusters. The method does not suffer from the curse of dimensionality as opposed to many unsupervised learning algorithms. Precise construction of diffusion map via projection onto eigensubspaces with large eigenvalues of the transition matrix on dataset-associated graph is discussed. We implement classical diffusion map on some example dataset, revealing the power to discover lower-dimensional data embeddings and to perform high-dimensional data clustering.

The second part of this thesis explain our efforts to construct quantum algorithm for manifold learning. We begin by harnessing the power of quantum computational speedup to bring classical diffusion map into quantum realms. Using the combination of coherent state classical data encoding scheme, construction of kernel operator from coherent states, and quantum computational speed up for finding eigenvalues/eigenvectors from Quantum Phase Estimation subroutine, we propose Quantum Diffusion Map, which hopefully will become useful as a quantum-enhanced non-linear dimensionality reduction and manifold learning algorithm. We then explore an algorithm based on coined quantum walk that may help accelerate data clustering. Numerical simulations of quantum walk on a toy dataset highlights the success of our proposal for automatic identifications of data clusters.

Bibliography

- [1] S. Lloyd, M. Mohseni, and P. Rebentrost, *Quantum principal component analysis*, Nature Physics **10**, 631-633, 2014.
- [2] A.W. Harrow, A. Hassidim, and S. Lloyd, *Quantum algorithm for linear systems of equations*, Phys. Rev. Lett. **103**, 150502, 2009.
- [3] M. Nielsen, and I. Chuang, *Quantum computation and quantum information*. Phys. Today, **54**, 60-2. 2001.
- [4] R. Chatterjee, and T. Yu, *Generalized coherent states, reproducing kernels, and quantum support vector machines*, Quantum Information and Communication **17**, 1292, 2017.
- [5] J. De la Porte, et. al., *An introduction to diffusion maps*. Proceedings of the 19th Symposium of the Pattern Recognition Association of South Africa (PRASA 2008), Cape Town, South Africa. 2008.
- [6] I. Cong, and L. Duan, *Quantum discriminant analysis for dimensionality reduction and classification*. New Journal of Physics **18.7**: 073011, 2016.
- [7] R. Moon, et al., *Visualizing transitions and structure for high dimensional data exploration*. bioRxiv: 120378. 2017.
- [8] B. Bah, *Diffusion Maps: Analysis and Applications*. MSc dissertation, University of Oxford, 2008
- [9] R. Coifman, and S. Lafon, *Diffusion maps*. Applied and Computational Harmonic Analysis, **21**:5-30, 2006.
- [10] U. von Luxburg, *A Tutorial on Spectral Clustering*. Statistics and Computing, **17**(4), 2007.

- [11] S. Lafon, *Diffusion maps and geometric harmonics*, PhD Thesis, Yale University, 2004.
- [12] C. M. Bishop. *Gaussian Processes*. Pattern Recognition and Machine Learning, Springer, 2006.
- [13] D. A. Spielman, *Laplacian Matrices of Graphs: Spectral and Electrical theory*, Yale University, 2011.
- [14] R. Socher, *Manifold Learning and Dimensionality Reduction with Diffusion Maps*, 2008.
- [15] M. Belkin, and P. Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*. Neural Comput, **15**:1373-1396, 2006.
- [16] S. Anlage, *Lecture note in the coherent states of the harmonic oscillator*, University of Maryland, 2016.
- [17] P. Renato, *Quantum walks and search algorithms*. New York: Springer, 2013.
- [18] D. Dua, and C. Graff, *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science. 2019.
<http://archive.ics.uci.edu/ml>
- [19] Qiskit, *Quantum Phase Estimation*, [Online], Available on:
qiskit.org/textbook

Appendix A

Coherent States

In the setting of a quantum simple harmonic oscillator, the state minimizing the uncertainty of a wave packet is the ground state $|0\rangle$, which satisfies $\hat{\mathbf{a}}|0\rangle = 0$ where $\hat{\mathbf{a}}$ is the annihilation (lowering) operator. If $|\alpha\rangle$ is an eigenstate of the annihilation operator such that

$$\hat{\mathbf{a}}|\alpha\rangle = \alpha|\alpha\rangle, \quad (\text{A.1})$$

such state $|\alpha\rangle$ also saturates the (Heisenberg) uncertainty bound and is called a *coherent state*. We now write a coherent state as a linear combination of harmonic oscillator eigenstates $\{|n\rangle\}$ where $\hat{\mathbf{a}}^\dagger\hat{\mathbf{a}}|n\rangle = n|n\rangle$ as

$$|\alpha\rangle = \sum_{n=0}^{\infty} \langle n|\alpha\rangle |n\rangle. \quad (\text{A.2})$$

Since

$$|n\rangle = \frac{(\hat{\mathbf{a}}^\dagger)^n}{\sqrt{n!}} |0\rangle, \quad (\text{A.3})$$

we have

$$|\alpha\rangle = \sum_{n=0}^{\infty} \left\langle 0 \left| \frac{(\hat{\mathbf{a}}^\dagger)^n}{\sqrt{n!}} \alpha \right. \right\rangle |n\rangle = \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} \langle 0|\alpha\rangle |n\rangle. \quad (\text{A.4})$$

Imposing the normalization condition $\langle\alpha|\alpha\rangle = 1$, then

$$1 = \sum_{n=0}^{\infty} \frac{|\alpha|^{2n}}{n!} |\langle 0|\alpha\rangle|^2 = e^{|\alpha|^2} |\langle 0|\alpha\rangle|^2 \rightarrow \langle 0|\alpha\rangle = e^{-|\alpha|^2/2}. \quad (\text{A.5})$$

So, the coherent state in the basis of eigenstates of quantum simple harmonic oscillator is [16]

$$|\alpha\rangle = e^{-|\alpha|^2/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle. \quad (\text{A.6})$$

In this thesis, we use coherent states to efficiently compress the classical data into qRAM according to Eqns. (3.2) - (3.3). It also facilitates the construction of a Gaussian kernel, see Section 3.2.

Appendix B

Quantum Phase Estimation (QPE)

QPE is designed to estimate the eigenvalues, with a quantum computational speedup, of a unitary operator \hat{U} :

$$\hat{U} |u_j\rangle = e^{2\pi i \theta_j} |u_j\rangle, \quad (\text{B.1})$$

where $\theta_j \in [0, 1]$; or more precisely, to obtain an n -bit approximation of $\theta_j \approx \tilde{\theta}_j = \sum_{m=1}^n \theta_{j,m} 2^{-m}$, where $\theta_{j,m} \in \{0, 1\}$. We may estimate the eigenvalue of a Hermitian matrix \hat{S} via the unitary time evolution $\hat{U} = e^{2\pi i \hat{S}}$ as follows. We first combine $O(t^2 \epsilon^{-1})$ copies of \hat{S} to construct $\hat{U} = e^{i \hat{S} t}$ with the accuracy of $O(\epsilon)$ [2]. The result of QPE applying to an input state $|\psi_0\rangle = \sum_j c_j |u_j\rangle$ is

$$|0\rangle |\psi_0\rangle = \sum_j c_j |0\rangle |u_j\rangle \xrightarrow{\text{QPE}} |\psi_1\rangle = \sum_j c_j |\tilde{\lambda}_j\rangle |u_j\rangle, \quad (\text{B.2})$$

where $|\tilde{\lambda}_j\rangle$ is the estimated binary representation of the eigenvalue of \hat{S} . Now, we have the superposition of the eigenvectors of \hat{S} with their estimated eigenvalues. The eigenvectors of the transition matrix \hat{P} can then be computed from (3.13).

We now explain the well-known QPE algorithm, which consists of three steps: create superposition, apply controlled-unitary operator, and apply the Quantum Fourier Transform. The algorithm's input consists of two registers: the first register has n qubits to store the estimated eigenvalue of the second register which is the exact eigen-

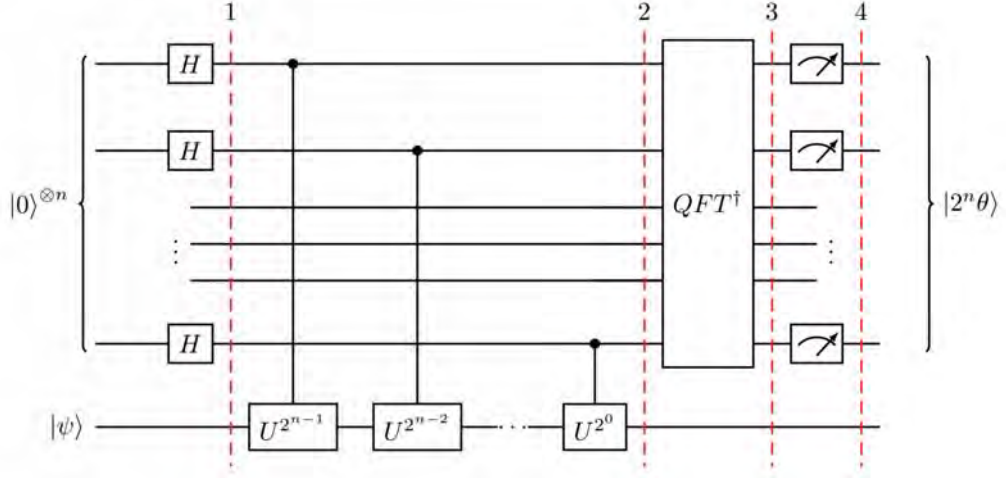


Figure B.1: The quantum circuit of QPE, taken from [19].

vector. First, we initialize the state at $|0\rangle|u_j\rangle$ where the second register is one of the exact eigenvectors of \hat{U} . Recalling that applying the Hadamard gate $\hat{U}_H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ to the computational basis $|0\rangle$ yields $\hat{U}_H |0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. Hence, applying the Hadamard gates to the first register of the initial state creates the superposition state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)^{\otimes n} |u_j\rangle$.

Next, we estimate the eigenvalue of the exact eigenstate in the second register by using the controlled-unitary operator. Recall that the controlled-unitary operator applies the unitary operator to the second register only when the first register (controlled-qubit) is in the computational basis $|1\rangle$. From (B.1), we have

$$\hat{U}^{2^k} |u_j\rangle = \hat{U}^{2^{k-1}} \hat{U} |u_j\rangle = \hat{U}^{2^{k-1}} e^{2\pi i \theta_j} |u_j\rangle = \dots = e^{2\pi i 2^k \theta_j} |u_j\rangle. \quad (\text{B.3})$$

We will apply the controlled-unitary operator \hat{U}^{2^k} to the $(n-k)^{\text{th}}$ qubit in the first register. The result is the state

$$\frac{1}{\sqrt{2}} \bigotimes_{m=1}^n (|0\rangle + e^{2\pi i 2^{n-m} \theta_j} |1\rangle) |u_j\rangle = \frac{1}{\sqrt{2}} \sum_{k=0}^{2^n-1} e^{2\pi i k \theta_j} |k\rangle |u_j\rangle, \quad (\text{B.4})$$

where k is a base-2 number.

The last step of QPE algorithm is to apply the inverse Quantum

Fourier Transform (QFT[†]) to the first register:

$$\text{QFT}^\dagger \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i k \theta_j} |k\rangle = \frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{2\pi i k \theta_j} e^{-2\pi i k x / 2^n} |x\rangle \quad (\text{B.5})$$

$$= \frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{-2\pi i k (x - 2^n \theta_j) / 2^n} |x\rangle. \quad (\text{B.6})$$

Note that only $x = 2^n \theta_j$ will have a non-zero amplitude; i.e., inverse QFT collapse the state of the first register to $|2^n \theta_j\rangle$. Therefore, the final output state of QPE is $|2^n \theta_j\rangle |u_j\rangle$. Since $\theta_j \approx \tilde{\theta}_j = \{\theta_{j,1} \theta_{j,2} \cdots \theta_{j,n}\} = \sum_{m=1}^n \theta_{j,m} 2^{-m}$ in the binary representation, x in the binary representation is thus $\tilde{\theta}$. If we initialize the second register as a superposition of the eigenvectors $\{|u_j\rangle\}$, $|\psi_0\rangle$, the result of QPE is (B.2). Next step, we will measure the first register of the algorithm. We will have the binary representation of an eigenvalue λ_j with the probability to get this eigenvalue $|c_j|^2$, the second register will be an eigenvector $|u_j\rangle$ corresponding to the eigenvalue λ_j which is coming from the first register.

B.1 An Alternative Method for Finding Eigenvalues and Eigenvectors

Ref. [1] proposed applying QPE algorithm to the density matrix $\hat{\rho}$ in (3.7), which results in the eigenvalues and the eigenvectors of $e^{-i\hat{\rho}t}$. To see this, consider the eigenvalue equation of a Hermitian operator \hat{A} with $\hat{A}|u_j\rangle = \lambda_j|u_j\rangle$. The spectral decomposition of \hat{A} reads $\hat{A} = \sum_j \lambda_j |u_j\rangle\langle u_j|$. If we use the state \hat{A} as the initial state with an eigenvalues register, i.e., $\hat{A}_0 = \sum_j \lambda_j |0\rangle\langle 0| \otimes |u_j\rangle\langle u_j|$, after QPE, we will obtain $\sum_j \lambda_j |\tilde{\lambda}_j\rangle\langle \tilde{\lambda}_j| \otimes |u_j\rangle\langle u_j|$ as a result.