MANIFOLD INTEGRATION:

DATA INTEGRATION ON MULTIPLE MANIFOLDS

A Dissertation

by

HEE YOUL CHOI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2010

Major Subject: Computer Science

MANIFOLD INTEGRATION:
DATA INTEGRATION ON MULTIPLE MANIFOLDS

A Dissertation

by

HEE YOUL CHOI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Yoonsuck Choe
Committee Members,	Vivek Sarin
	Tracy Hammond
	Mikyoung Jun
Head of Department,	Valerie E. Taylor

May 2010

Major Subject: Computer Science

ABSTRACT

Manifold Integration: Data Integration on Multiple Manifolds. (May 2010)

Hee Youl Choi, B.S., Pohang University of Science and Technology;

M.S., Pohang University of Science and Technology

Chair of Advisory Committee: Dr. Yoonsuck Choe

In data analysis, data points are usually analyzed based on their relations to other points (e.g., distance or inner product). This kind of relation can be analyzed on the manifold of the data set. *Manifold learning* is an approach to understand such relations. Various manifold learning methods have been developed and their effectiveness has been demonstrated in many real-world problems in pattern recognition and signal processing. However, most existing manifold learning algorithms only consider one manifold based on one dissimilarity matrix. In practice, multiple measurements may be available, and could be utilized. In pattern recognition systems, *data integration* has been an important consideration for improved accuracy given multiple measurements. Some data integration algorithms have been proposed to address this issue. These integration algorithms mostly use statistical information from the data set such as uncertainty of each data source, but they do not use the structural information (i.e., the geometric relations between data points). Such a structure is naturally described by a manifold.

Even though manifold learning and data integration have been successfully used for data analysis, they have not been considered in a single integrated framework. When we have multiple measurements generated from the same data set and mapped onto different manifolds, those measurements can be integrated using the structural information on these multiple manifolds. Furthermore, we can better understand the structure of the data set by combining multiple measurements in each manifold using data integration techniques.

In this dissertation, I present a new concept, *manifold integration*, a data integration method using the structure of data expressed in multiple manifolds. In order to achieve manifold integration, I formulated the manifold integration concept, and derived three manifold integration algorithms. Experimental results showed the algorithms' effectiveness in classification and dimension reduction. Moreover, for manifold integration, I showed that there are good theoretical and neuroscientific applications.

I expect the manifold integration approach to serve as an effective framework for analyzing multimodal data sets on multiple manifolds. Also, I expect that my research on manifold integration will catalyze both manifold learning and data integration research.

To my parents

ACKNOWLEDGMENTS

During my Ph.D. study, I received valuable support from many people. Especially I would like to thank my advisor, Dr. Yoonsuck Choe, for his guidance and support on my study. Also I thank the committee members: Dr. Vivek Sarin (and Dr. Jianer Chen), Dr. Tracy Hammond and Dr. Mikyoung Jun for the valuable questions and discussions. Also I thank my previous advisor, Dr. Seungjin Choi from POSTECH, for his advice and help. In addition, I thank Dr. Anup Katake from StarVision Technologies for all the fun and interesting discussions about research, work and life. I also thank Dr. Ricardo Gutierrez-Osuna, who helped me a lot especially for the first year in Aggieland, Dr. Yunhee Kang (BaekSeok University) who has supported me since I first met him, and Dr. Sung Yang Bang (POSTECH) and Dr. Byungkyun Kang (POSTECH), who were my excellent references, my lab mates and the department staff who made my time at TAMU enjoyable.

In addition, there have been others I am thankful for. Here is a very short list of the people who supported me during my time in Aggieland: Gail Mills, who has helped me a lot in the office, the church, and has been a good counsellor to me, my roommate, Changyoung, from whom I received a great deal of assistance with cooking and all the “girl” problems, Youngkwon Hyung-Nim, who is a big brother to me, Sooyun and Tren, who helped me get used to the life in CNS and in Boston, Hyekyoung Un-Nee, who is the kindest person to me ever, Dr. Yum, who was like an angel sent by Him when I was in Boston, my little brother, Hailong, and sister, Kijoeng, and my old friends, Taeho, Ohan, Byounguk, Jaehoon, Wonman, Chulwon, and many, many more. Last but not least, my mom and dad, my older brother and sister-in-law and little sister and brother-in-law and nephew and nieces, my family, my love.

Remembering people around me and their support and love through the time, I cannot help but confess that I have been very blessed through these people. It was Him who designed this seemingly terrible time and gave me all the troubles and tears. It was also Him who has never stopped blessing me.

Finally, I would like to thank Starvision Technologies (Michael Jacox) and the Korean Science Foundation and my friends for their financial support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

More than two thousand years ago, the Greek philosopher Heraclitus (Fig. 1) said "you cannot step twice into the same river", referring to the fact that the world is in eternal flux. Even in this kind of flux, however, we can recognize the river as the same river based on the past experience since there are invariant features in the flowing river. Such invariant features can be found using computational methods like manifold learning using structural information of the measurement of the flux [1].

Fig. 1. "The weeping philosopher", Heraclitus (535 - 475 BC) by Johannes Moreelse. Heraclitus said "you cannot step twice into the same river."

However, the situation is more complex, if we look at the river closely from within. In the river, we can see, hear, and feel (with our sense of touch) the river flowing. Our brain integrates such sensory information to recognize the river as a sin-

This dissertation follows the style of *IEEE Transactions on Neural Networks*.

gular entity. Manifold learning is not possible to deal with such multimodal data in its original formulation. Can it be extended to utilize such a rich data stream?

In this dissertation, I will address this very question. To continue with the above example, what we want ultimately is to discover invariant features of the river based on the multiple sources of sensory information. In data analysis terms, measurements of the data stream varies over time but there are invariant properties in the measurement. Such invariant properties are inherent in the multiple measurements. The problem is to integrate multiple measurements and extract invariant features. Earlier approaches to solve the problem were based on either manifold learning for invariant features from one measurement, or data integration of multiple measurements without considering the structural properties of the data (data integration does not use the structural information contrary to manifold learning). In this dissertation, I propose a new framework to solve the problem by integrating the two approaches.

A. Manifold Learning

In data analysis, data points such as images or speech signals are analyzed based on their relations to other points (e.g., distance or inner product) rather than their absolute position in the coordinates as commonly done in principal component analysis (PCA) or multidimensional scaling (MDS) (see [2, 3, 4]). From these relations, we can get coordinate-invariance property, with which the analysis is invariant against any choice of coordinate system for the data points. This kind of relation can be better understood on the manifold of the data set (see [5, 6]) as shown in Fig. 2.

For example, suppose that a facial image data set is generated by varying the orientation of a face like the river flowing. Then, the resulting facial images could occupy a nonlinear low-dimensional manifold (e.g., a curve) embedded in the image

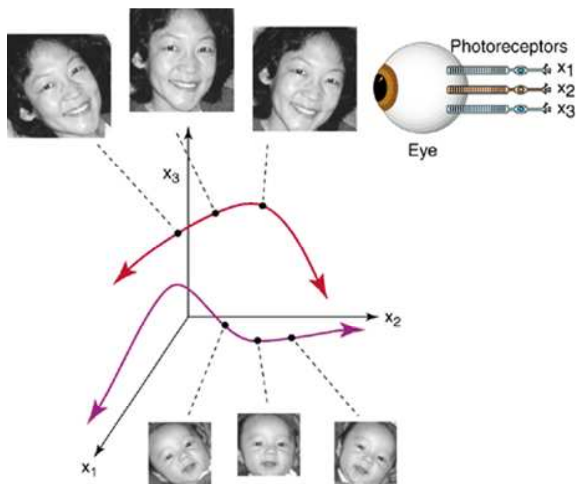


Fig. 2. Manifold way of perception. When we rotate a facial image, the resulting facial images could occupy a nonlinear low-dimensional manifold (e.g., a curve) embedded in the image space. Adapted from [1].

space as in Fig. 2. If we can find the embedded manifold, we can have orientation-invariant facial image representation. So, it is important to understand the manifold structure of the data. *Manifold learning* involves inducing a smooth nonlinear low-dimensional manifold on which data points are organized, where the data points are originally drawn from a high-dimensional space [1]. Various methods have been developed including Isomap and locally linear embedding (LLE) (for example see [7, 8, 9]) and their effectiveness has been demonstrated in many real-world tasks in pattern recognition and signal processing [10, 11, 12]. Also, note that kernel methods [13] can be viewed as manifold learning since a kernel matrix can be derived from a Riemannian metric and vice versa [14, 15].

B. Data Integration

However, most existing manifold learning algorithms only consider one manifold based on one dissimilarity matrix. As shown in Fig. 3, what if we have more than one

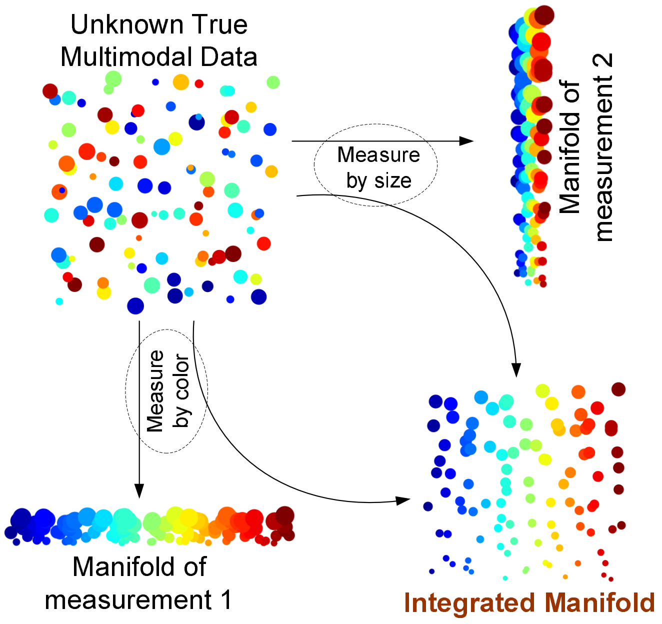

Unknown True Multimodal Data
Measure by size
Manifold of measurement 2
Measure by color
Manifold of measurement 1
Integrated Manifold

Fig. 3. An example of manifold integration when two manifolds are available from one underlying data set. The two manifolds are from different types of measurement (color or size), and, taken separately, are not suitable for fully understanding the data set. However, we can integrate the two measurements to obtain one integrated manifold that can give a complete picture of the data set.

measurements each of which generates one manifold as in the river example above with multiple sensors? Since the manner in which the data set is measured can be different and measurements are expected to have noise, more measurements will have a higher chance of achieving more accurate data analysis than just one measurement. Then, how can different measurements from different manifolds be used together to form an integrated manifold? A key question in this case is how different pieces of information can be integrated (data fusion).

In pattern recognition systems, *data integration* has been an important consideration for improved accuracy. Some data integration algorithms have been proposed to address this issue (see [16]), such as Bayesian inference [17], Dempster-Shafer theory (D-S theory or evidence theory) [18, 19], clustering algorithms [4], and neural networks [20]. Kernel-based integration methods have also been used for integration [21]. These integration algorithms mostly use statistical information from the data set such as uncertainty of each data source as in evidence theory, but they do not use the structural information (i.e., the geometric relations between data points). Such a structure is described effectively by a manifold.

C. Manifold Integration

Even though manifold learning and data integration have been successfully used for data analysis and there is an abundance of work in manifold learning and in data integration, they have not been considered together in a single integrated framework. When we have multiple measurements generated from the same data set and mapped onto different manifolds, those measurements can be integrated using the structural information in the multiple manifolds. Furthermore, we can better understand the structure of the data set by combining multiple measurements in each manifold using data integration techniques.

The goal of this dissertation is to consider the two concepts together: manifold learning and data integration, to develop effective manifold integration algorithms for multimodal data sets.

To fulfill the goal, a new concept is proposed, *manifold integration*, a data integration method using the structure of the data expressed in multiple manifolds. In order to handle multimodal data sets on multiple manifolds, the algorithms are based on the relationship between data points from each measurement and the relationship is obtained from the resulting manifolds. Finally, the relationships from different manifolds are merged. For manifold integration, three algorithms are derived in this dissertation: (1) *kernel oriented discriminant analysis* (KODA), (2) *random walk on multiple manifolds* (RAMS), and (3) *manifold α-integration* (MAI).

First, KODA integrates multiple manifolds assuming that all the manifolds are submanifolds of the same unknown manifold and that they are connected since the distances between points on different submanifolds are used. The method also maximizes the separability between different classes. In practice, however, multiple submanifolds may not be connected on one manifold. To overcome such a strong assumption in KODA, we need to obtain the structural information from each manifold independently and then integrate all the information while not connecting the manifolds geometrically. RAMS integrates the structural information statistically (not geometrically) based on the transition matrices of all manifolds where the transition matrices are used for random walk on the manifold to determine the distance between points in each measurement. It is assumed that the transition matrix of one manifold is statistically independent from those of the other manifolds. However, since statistical independence is still a strong assumption, I propose a more generalized integration method, MAI. MAI uses a generalized averaging method called α-integration to integrate dissimilarity matrices (or transition matrices), instead of just using the geometric mean of multiple transition matrices as in RAMS. This generalized method can overcome the independence assumption in RAMS using different α values which determine the relationship between multiple sources.

All the proposed algorithms above for manifold learning and data integration as well as manifold integration are tested with synthetic and real-world data sets, showing superb performance, compared to competing approaches such as DISTATIS, kernel fusion and random walks. These tests demonstrate the algorithms' effectiveness in classification and dimension reduction. Moreover, for manifold integration, I show that there are good theoretical and neuroscientific applications. First, I apply MAI to multiple kernels from multiple kernel methods such as kernel principal component analysis (KPCA) so that MAI can integrate the multiple distance matrices converted from the corresponding kernel matrices and recover an integrated kernel matrix from the integrated distance matrix. Second, MAI is proposed as a new framework for sensorimotor integration as well as multi-sensory integration. By manifold integration, the sensory and motor maps are integrated into the same space so that they can be compared to each other directly.

I expect the manifold integration methods I developed to serve as an effective framework for analyzing multimodal data sets on multiple manifolds including the sensor and motor maps in the human brain. My prior work, especially kernel Isomap, has been recognized as a useful tool in scientific applications such as emotion analysis [22] and sketch recognition [23]. Moreover, some researchers have applied kernel Isomap in their algorithm development such as supervised kernel Isomap [24] and weighted kernel Isomap [25]. Since this dissertation extends the performance and the ability of kernel Isomap for multiple manifolds, this research can help achieve better performance in applications where multiple measurements are available, and can open new directions in manifold learning research as well as in data integration research.

D. Organization

The rest of this dissertation is organized as follows. In Chapter II, I will review manifold learning and data integration with a focus on some topics related to manifold integration. In Chapter III, I will describe the theoretical bases for manifold integration: manifold learning and data integration, especially, my prior work on kernel Isomap and extension of α-integration. I will present manifold integration in Chapter IV with the concept, algorithms, and experiments. In Chapter V, as an advanced application, I propose manifold integration as a new framework for kernel integration and sensorimotor integration. Discussion and future work follow in Chapter VI. In Chapter VII, conclusions are drawn.

CHAPTER II

BACKGROUND

Before we begin, some background knowledge of manifold learning and data integration is required, as two key ingredients of manifold integration. Manifold learning and data integration have a large volume of references (see [1, 16] and references therein), so, in this chapter, I will only provide a focused review.

A. Manifold Learning

1. Manifold

A manifold is a topological space which is locally Euclidean. In order words, a manifold is a topological space locally homeomorphic to an open subset of Euclidean space $\mathbb{R}^n$, where n is a non-negative integer[1]. For example, if we think of some surface region of the Earth as in Fig. 4, it is on a manifold, which is different from the Euclidean plane. In the Euclidean plane, the sum of angles inside of a triangle should be 180^o while in a general manifold it is not always the case. But zooming into a sufficiently small local region shows that the surface of the Earth is a Euclidean plane with the sum of angles inside a triangle equal to 180^o.

Mathematically speaking, points in a local region on a manifold can be indexed by a subset of a low-dimensional Euclidean space as shown in Fig. 5. The index in this case is called the coordinate of the point on the manifold. However, there may not be such a coordinate system globally. The goal of manifold learning algorithms is to obtain a global coordinate system with some loss of information while trying to minimize such a loss. In other words, assuming that the manifold in a high dimen-

[1]For more mathematical definitions, see [5, 6].

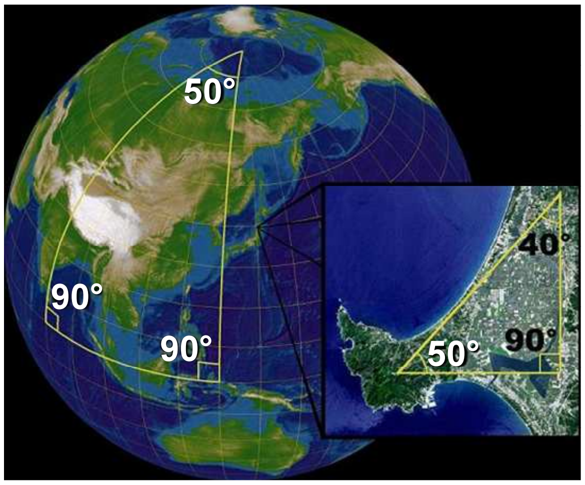


Fig. 4. An example of manifold by Lars H. Rohwedder. In a Euclidean space, the sum of angles inside of a triangle should be 180^o. In more global figure, it is not a Euclidean space since the sum of angles is 230^o. But when zooming in a place locally enough, it is a Euclidean space with the sum of angles of 180^o.

sional space is generated by a function f which is called a coordinate patch, manifold learning tries to find its low-dimensional representation.

Note that in data analysis, a manifold of data set is conceptually the same as the mathematical manifold, especially topological manifold. Actually, however, a manifold in data analysis cannot be exactly the same as the mathematical one, because in data samples we cannot define even an *open set*. So, we adapt the concepts from mathematical theories on manifold but implement the concept as a discrete approximation of the true manifold. For example, a locally open set can be implemented by neighborhood in the graph made of data points, and geodesic distances can be implemented by shortest pathes in the graph. If the data set is dense enough (ideally with infinite data samples), then the discrete approximation will asymptotically converge to the true manifold.

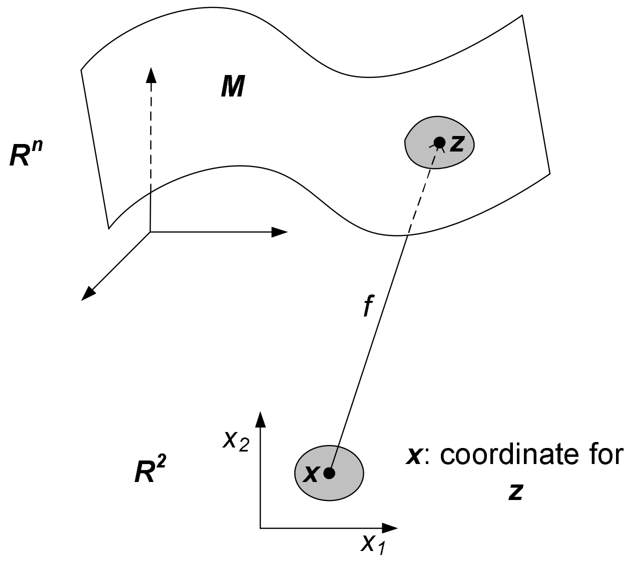


Fig. 5. Parameterization of a manifold. Nonlinear space can be parameterized with a subspace of a Euclidean space. Manifold learning is an approach to obtain a global parameterization.

2. Algorithms for Manifold Learning

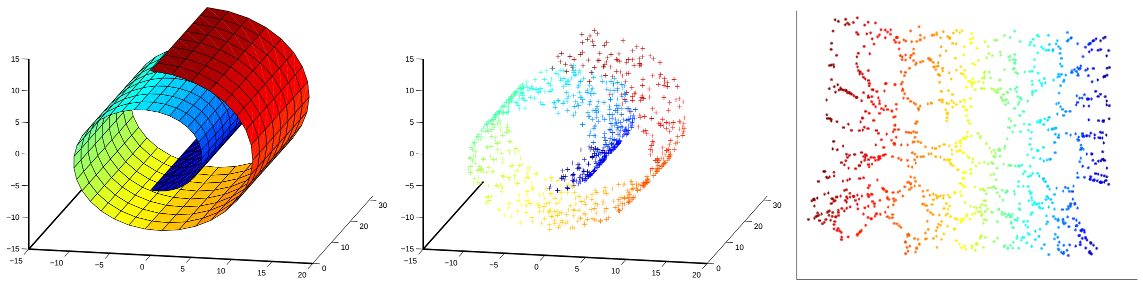
In data analysis, a data set can be given in a high-dimensional vector format. However, the number of dominant factors that are expected to have generated the high-dimensional data set might be much less than the intrinsic dimensionality of the data set. Principal component analysis (PCA) is one of the most popular methods to find such factors from the data set [2]. PCA works by maximizing the variance of the data set in the projected space. As shown later, classical scaling (an instance of metric multidimensional scaling (MDS)) is closely related to PCA [3]. The projection of the centered data onto the eigenvectors of the data sample covariance matrix returns the classical scaling solution. Classical scaling, where Euclidean distance is employed as a dissimilarity measure, can be explained in the context of PCA. In the same manner, non-Euclidean dissimilarity can be used, although there is no guarantee that the eigenvalues will be nonnegative. A relationship between kernel principal component analysis (KPCA, [13]) and metric MDS was also established in [26].

In a kernel machine such as KPCA or kernel Fisher discriminant (KFD) [27], the kernel function can be considered as defining implicitly the corresponding Riemannian metric of the embedded low dimensional space in the high dimensional data space [14]. However, these kernel functions are not learned from the data set but are given by users after a number of experiments. In practice, it is nontrivial to find the proper kernel function with their proper parameters. Moreover, even a wisely chosen function may not reflect the real data structure while just emphasizing some of the distances. For example, if an exponential function is used as a kernel function, the distances with the neighbors are emphasized where long distances are relatively ignored.

Manifold learning finds a kernel matrix directly from the data set using the data structure and induces a smooth nonlinear low-dimensional manifold [15, 28]. Recently, various manifold learning methods (for example see [7, 8, 29, 9, 30, 31]) have been developed in the machine learning community and their performance has been demonstrated in many real-world problems in pattern recognition and signal processing. Since a manifold is defined as a locally Euclidean topological space, the manifold learning methods are commonly based on Euclidean distance between neighboring points in the data set. However, even though most manifold learning methods are based on locally Euclidean geometry, many different algorithms result, dependent on how the data structure defines and utilizes local information. Roughly, they can be categorized into two approaches: (1) global approaches preserving the global structure composed of local structures, and (2) local approaches preserving just the local structure. Isomap and locally linear embedding (LLE) are representative manifold learning methods for the global approach and the local approach, respectively. Even though I will be using the global approach throughout this dissertation, there is no specific reason for doing so and the main idea of this dissertation does not depend on a specific choice of the type of manifold learning approach. In the following sections, I briefly review Isomap and its relation to KPCA.

3. Review of Isomap

Classical scaling, a form of metric MDS, is a method of low-dimensional embedding based on pairwise similarity between data points. In general, Euclidean distance is used as a measure of dissimilarity (or similarity) in MDS. The basic idea in Isomap [7] is to use geodesic distances on a neighborhood graph in the framework of classical scaling, in order to utilize the nonlinear manifold structure, instead of subspace based on Euclidean distances. Fig. 6 shows Isomap on a Swiss roll data set. The sum of edge



(a) (b) (c)

Fig. 6. Isomap on a Swiss roll data set. (a) Swiss roll manifold, (b) data samples from the manifold, and (c) two dimensional space extracted by Isomap.

weights along the shortest path between two nodes is assigned as the geodesic distance. The top n eigenvectors of the geodesic distance matrix represent the coordinates in the n-dimensional feature space. Fig. 7 shows the result of Isomap applied to 64×64 pixel images of a face rendered with different poses and lighting directions.

Following the connection between classical scaling and PCA, metric MDS can be interpreted as KPCA [26]. In a similar manner to the above the geodesic distance

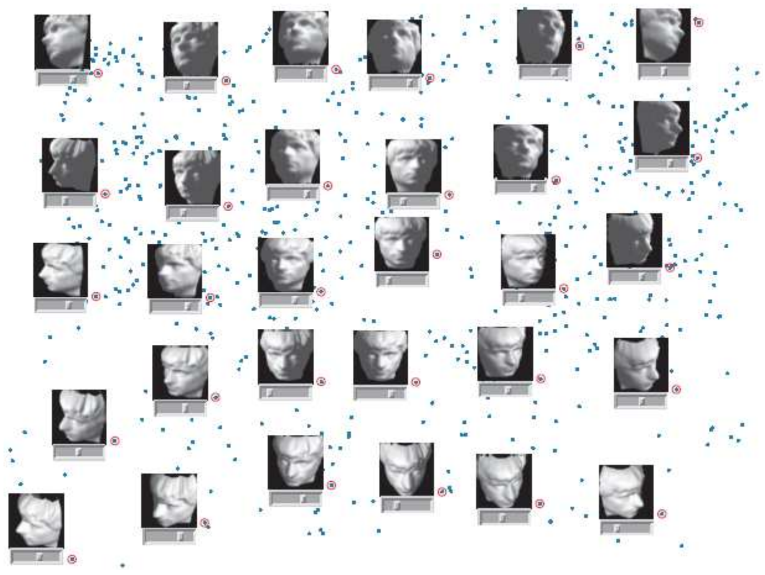


Fig. 7. The result of Isomap applied to 64×64 pixel images of a face rendered with different poses and lighting directions. Adapted from [7].

matrix in Isomap can be converted to a kernel matrix and Isomap can be seen as a kind of kernel machine [15].

Kernel machines are data analysis algorithms that use the kernel trick. Examples of kernel machine include support vector machines (SVMs) and KPCA. Kernel trick is a method that generates a Gram matrix (inner product matrix or kernel matrix) in a higher dimensional feature space while bypassing explicit mapping of the data set into the feature space. For the kernel matrix to be an inner product matrix in the feature space, the kernel matrix calculated from the kernel trick should be positive semidefinite, which is defined as follows [14].

Definition 1 (Positive Semidefinite) *A symmetric matrix* $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ *is positive semidefinite, if* $\boldsymbol{x}^\top \boldsymbol{K}\boldsymbol{x} \geq 0$, *for any nonzero vector* $\boldsymbol{x} \in \mathbb{R}^n$.

So the kernel matrix should be carefully obtained. Usually judiciously chosen kernel functions are used to generate a positive semidefinite kernel matrix[2]. Note that kernels in data analysis and those in other area like statistics (kernel density estimation) are used differently and their properties are different. For more details, see [14, 32].

The doubly centered geodesic distance matrix $\boldsymbol{K}$ in Isomap is of the form

$$\boldsymbol{K} = -\frac{1}{2}\boldsymbol{H}\boldsymbol{D}^2\boldsymbol{H}, \tag{2.1}$$

where $\boldsymbol{D}^2 = [D_{ij}^2]$ means the element-wise square of the geodesic distance matrix $\boldsymbol{D} = [D_{ij}]$, $\boldsymbol{H}$ is the centering matrix, given by

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{N}\boldsymbol{e}_N\boldsymbol{e}_N^\top \tag{2.2}$$

for $\boldsymbol{e}_N = [1 \ldots 1]^\top \in \mathbb{R}^N$. Note that Eq. (2.1) is a kernel matrix, which is obtained by a distance matrix rather than a kernel function.

4. Isomap vs. KPCA

Understanding the relationship between MDS and PCA is essential in understanding the relationship between Isomap and KPCA. Given a data set $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N]$, the Euclidean distance D_{ij} from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ is

$$D_{ij}^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top(\boldsymbol{x}_i - \boldsymbol{x}_j). \tag{2.3}$$

Let the inner product matrix be $\boldsymbol{B}$, where

$$\boldsymbol{B}_{ij} = \boldsymbol{x}_i^\top\boldsymbol{x}_j. \tag{2.4}$$

[2]A definition of kernel function is given in Chapter V.

Then, with eigen-decomposition, we can get

$$\begin{aligned}\boldsymbol{B} &= \boldsymbol{X}^\top \boldsymbol{X} \\ &= \boldsymbol{V}_\mathrm{M} \boldsymbol{\Lambda}_\mathrm{M} \boldsymbol{V}_\mathrm{M}^\top,\end{aligned} \tag{2.5}$$

where $\boldsymbol{V}_\mathrm{M}$ and $\boldsymbol{\Lambda}_\mathrm{M}$ are the eigenvector and the eigenvalue matrix of $\boldsymbol{B}$, respectively. Finally, the projected data from MDS is obtained by

$$\boldsymbol{Y}_\mathrm{M} = \boldsymbol{\Lambda}_\mathrm{M}^{1/2} \boldsymbol{V}_\mathrm{M}^\top. \tag{2.6}$$

For PCA, the covariance matrix can be decomposed as follows:

$$\boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{V}_\mathrm{P} \boldsymbol{\Lambda}_\mathrm{P} \boldsymbol{V}_\mathrm{P}^\top, \tag{2.7}$$

where $\boldsymbol{V}_\mathrm{P}$ and $\boldsymbol{\Lambda}_\mathrm{P}$ are the eigenvector and the eigenvalue matrix of the covariance matrix $\boldsymbol{X}\boldsymbol{X}^\top$, respectively. The projected data from PCA is given by

$$\boldsymbol{Y}_\mathrm{P} = \boldsymbol{V}_\mathrm{P}^\top \boldsymbol{X}. \tag{2.8}$$

With $\boldsymbol{V}_\mathrm{P} = \boldsymbol{X}\boldsymbol{V}_\mathrm{M}$, the relation between $\boldsymbol{Y}_\mathrm{P}$ and $\boldsymbol{Y}_\mathrm{M}$ becomes:

$$\begin{aligned}\boldsymbol{Y}_\mathrm{P} &= \boldsymbol{V}_\mathrm{P}^\top \boldsymbol{X} \\ &= (\boldsymbol{X}\boldsymbol{V}_\mathrm{M})^\top \boldsymbol{X} \\ &= \boldsymbol{V}_\mathrm{M}^\top \boldsymbol{B} \\ &= \boldsymbol{\Lambda}_\mathrm{M}^{1/2} \boldsymbol{Y}_\mathrm{M}.\end{aligned} \tag{2.9}$$

That is, in the case of Euclidean distance, PCA and MDS are equivalent to each other, except for the ambiguity in scale.

Note that manifold learning is also known as nonlinear dimensionality reduction. More features provide more information, which gives potentially higher accuracy.

Unfortunately, however, more features make it harder to analyze, because of the curse of dimensionality. As a solution, we can start with as many potentially useful features as possible, and then reduce the number of features. There are two approaches to reduce the number of features: (1) feature selection methods select the salient features by some criteria and (2) feature extraction methods obtain a reduced set of features by a transformation of all features. Manifold learning belongs to the second approach.

B. Data Integration

In many research areas such as automatic target recognition (ATR) and target tracking, data integration has been an important issue to achieve improved accuracy than that based on a single source of information because one sensor might not be good enough to provide unambiguous information. Some data integration algorithms have been proposed (see [16]) such as Bayesian inference [17], evidence theory [18, 19], clustering algorithms and neural networks [20]. Kernel-based integration methods have also been used for integration [21].

According to [33], there are three basic alternatives in data integration: (1) direct integration of measurements, (2) integration of the extracted features, or (3) decision-level integration (or high-level inferences). Each of these approaches uses different integration techniques. If the data set consists of multiple heterogeneous measurements (for example, visual data and audio data), then the direct integration approach cannot be applied. Otherwise, classical estimation methods such as Kalman filtering can be applied [34]. Pattern recognition techniques such as neural networks, clustering algorithms, or template methods are feature-level integration methods. The decision-level integration approach which combines determinations based on each measurement includes weighted decision methods, Bayesian inference, and evidence theory.

Recently, a general approach, α-integration, was proposed by [35] for stochastic model integration of multiple positive measures. It can be applied to any of the three levels: homogeneous (positive) measurements, features, or classification results. α-integration is a one-parameter family of integration, where the parameter α determines the characteristics of integration. Given a number of stochastic models in the form of probability distributions, it finds the optimal integration of the sources in the sense of minimizing α-divergence. Many artificial neural network models for stochastic problems such as the mixture (or product) of experts model [36, 37] can be considered as special cases of α-integration. In this dissertation, α-integration is adopted as a general integration method for multiple manifolds. Although simple integration methods are also tested in this dissertation, they can all be considered as a special case of α-integration.

1. Review of α-Integration

Here I provide a brief overview of α-integration, more details on which can be found in [38, 35].

Let us consider two positive measures of random variable x, denoted by $m_1(x) > 0$ and $m_2(x) > 0$, satisfying

$$\int m_i(x)\, dx > 0,$$

for $i = 1, 2$. The simplest integration of these two positive measures is to compute the arithmetic mean

$$\widetilde{m}_{\mathrm{a}}(x) = \frac{1}{2}\left\{m_1(x) + m_2(x)\right\},$$

or the geometric mean

$$\widetilde{m}_{\mathrm{g}}(x) = \sqrt{m_1(x)m_2(x)}.$$

Fig. 8 shows an example of α-mean with two sources.

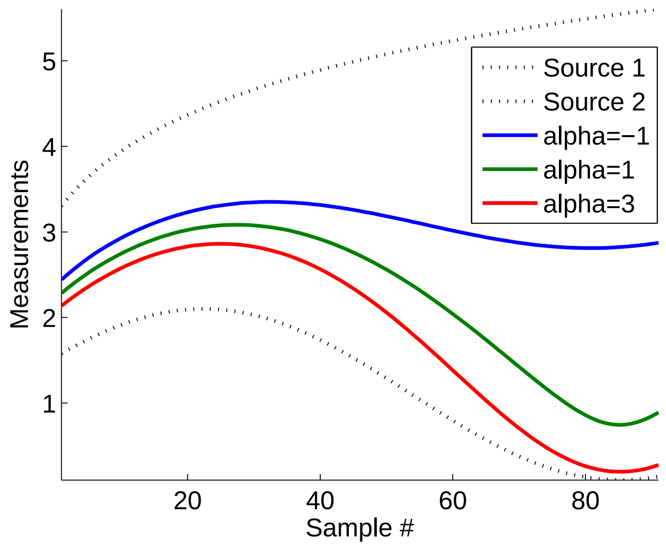


Fig. 8. An example of α-mean of two sources with different α values. Two black dotted curves are sources and the three solid curves are the α-mean.

α-mean [35] is a one-parameter family of means, which is defined by

$$\widetilde{m}_{\alpha}(x) = f_{\alpha}^{-1}\left(\frac{1}{2}\left\{f_{\alpha}(m_1(x)) + f_{\alpha}(m_2(x))\right\}\right), \tag{2.10}$$

where $f_{\alpha}(\cdot)$ is a differentiable monotone function given by

$$f_{\alpha}(z) = \begin{cases} z^{\frac{1-\alpha}{2}}, & \alpha \neq 1, \\ \log z, & \alpha = 1. \end{cases} \tag{2.11}$$

The function $f_{\alpha}(\cdot)$ in Eq. (2.11) is the only function which leads the α-mean to be *linear scale free* for $c > 0$, i.e., the α-mean of $c\,m_1(x)$ and $c\,m_2(x)$ is $c\,\widetilde{m}_\alpha(x)$ [39, 38],

$$c\,\widetilde{m}_\alpha(x) = f_\alpha^{-1}\left(\frac{1}{2}\{f_\alpha(c\,m_1(x)) + f_\alpha(c\,m_2(x))\}\right).$$

α-mean includes various means as its special case:

- $\alpha = -1$: arithmetic mean
- $\alpha = 1$: geometric mean
- $\alpha = 3$: harmonic mean
- $\alpha = \infty$: $\widetilde{m}_\infty(x) = \min(m_1(x), m_2(x))$
- $\alpha = -\infty$: $\widetilde{m}_{-\infty}(x) = \max(m_1(x), m_2(x))$

The value of the parameter α (which is usually specified in advance) reflects the characteristics of the integration. As α increases, the α-mean resorts more to the smaller of $m_1(x)$ or $m_2(x)$, while as α decreases, the larger of the two is considered more as in Fig. 9 [35].

α-mean can be generalized to the weighted α-mixture of M positive measures $m_1(x), \ldots, m_M(x)$ with weights w_i, which is referred to as *α-integration* of $m_1(x), \ldots, m_M(x)$ with weights w_i [35].

Definition 2 (α-integration) *The α-integration equation of $m_i(x)$, $i = 1, \ldots, M$, with weights w_i is defined by*

$$\widetilde{m}(x) = f_\alpha^{-1}\left(\sum_{i=1}^{M} w_i f_\alpha(m_i(x))\right), \tag{2.12}$$

where $w_i > 0$ for $i = 1, \ldots, M$ and $\sum_{i=1}^{M} w_i = 1$.

What are described in the case of two positive measures above carries over to the case of M positive measures.

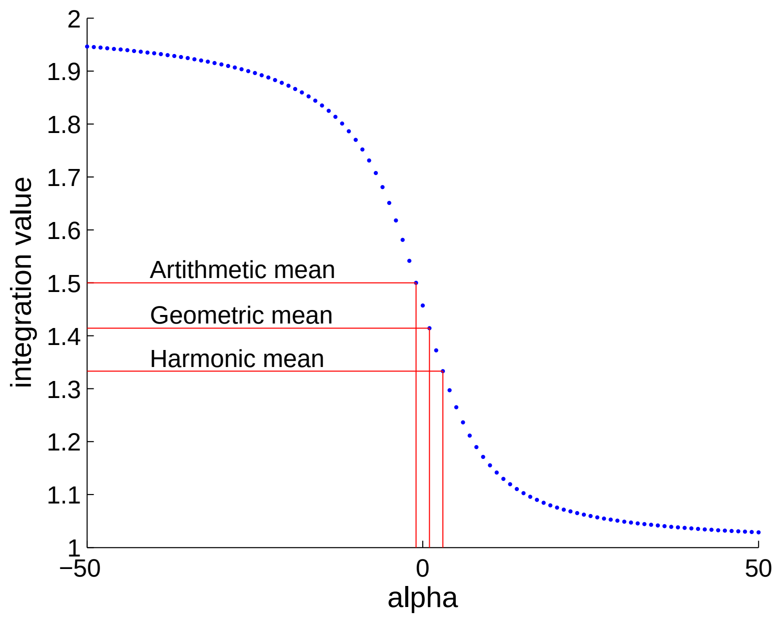


Fig. 9. α-integration includes various means as its special case according to the α value. Here, the values of two measures are 1 and 2, respectively. Note that the α-integration value monotonically decreases and converges to either 1 or 2.

Given M positive measures, $m_i(x)$, $i = 1, \dots, M$, the goal of integration is to seek their weighted average $\widetilde{m}(x)$ that is as close to each of the measures as possible, while how close two positive measures are is evaluated using divergence. It was shown in [35] that the α-integration $\widetilde{m}(x)$ is optimal in the sense that

$$\mathcal{J}_\alpha[\widetilde{m}(x)] = \sum_{i=1}^{M} w_i D_\alpha[m_i(x) \,\|\, \widetilde{m}(x)] \tag{2.13}$$

is minimized, where $D_\alpha[m_i(x) \,\|\, \widetilde{m}(x)]$ is the *α-divergence* of $\widetilde{m}(x)$ from the measures $m_i(x)$.

The α-divergence belongs to a family of convex divergence measures which is known as *Csiszár's f-divergence*, called sometimes also *Ali-Silvey divergence* [40, 41].

Definition 3 (f-divergence) *Csiszár's f-divergence is defined by*

$$D_f[m_1 \| m_2] = \int m_1(x)\, f\left(\frac{m_2(x)}{m_1(x)}\right) dx, \tag{2.14}$$

where $f(z)$ is a convex function, $f : [0, \infty) \mapsto (-\infty, \infty]$, which is continuous at 0, satisfying $f(1) = 0$ and $f'(1) = 0$.

When $f(z) = z - 1 - \log z$, the f-divergence becomes KL-divergence (a.k.a. I-divergence)

$$D_{\mathrm{KL}}[m_1 \| m_2] = \int m_1(x) \log \frac{m_1(x)}{m_2(x)} dx - \int (m_1(x) - m_2(x)) dx. \tag{2.15}$$

In the case where positive measures $m_1(x)$ and $m_2(x)$ are probability distributions satisfying $\int m_1(x)dx = 1$ and $\int m_2(x)dx = 1$ then Eq. (2.15) is simplified as

$$D_{\mathrm{KL}}[m_1 \| m_2] = \int m_1(x) \log \frac{m_1(x)}{m_2(x)} dx.$$

Definition 4 (α-divergence) *The α-divergence $D_\alpha[m_1 \| m_2]$ is derived from f-divergence*

$D_f[m_1 \| m_2]$, making use of $f(\cdot)$ given by

$$f(z) = \begin{cases} \frac{4}{1-\alpha^2}\left\{\frac{1-\alpha}{2} + \frac{1+\alpha}{2}z - z^{\frac{1+\alpha}{2}}\right\}, & \alpha \neq \pm 1, \\ z - 1 - \log z, & \alpha = -1, \\ z - 1 + z \log z, & \alpha = 1. \end{cases} \tag{2.16}$$

That is, for $\alpha \neq \pm 1$,

$$\begin{aligned} D_\alpha[m_1 \| m_2] \quad = \quad & \frac{4}{1-\alpha^2}\left\{\int \frac{1-\alpha}{2} m_1(x) + \frac{1+\alpha}{2} m_2(x)\right. \\ & \left. - m_1(x)^{\frac{1-\alpha}{2}} m_2(x)^{\frac{1+\alpha}{2}} dx\right\}. \end{aligned}$$

For $\alpha = -1$, $D_\alpha[m_1 \| m_2] = D_{\mathrm{KL}}[m_1 \| m_2]$ given in Eq. (2.15), and for $\alpha = 1$, $D_\alpha[m_1 \| m_2] = D_{\mathrm{KL}}[m_2 \| m_1]$.

Note that α and $\boldsymbol{w}$ are given and fixed (i.e., specified by the user). However, these values are unknown before we get a specific data set and understand it.

C. Summary

Manifold learning is an effective dimension reduction method and Isomap is a popular method and can be interpreted as a kernel machine. On the other hand, data integration is an important issue to achieve improved accuracy in pattern recognition and signal processing tasks. α-integration is a general approach for stochastic model integration. These manifold learning and data integration methods have been studied separately, even though there are some cases which both approaches could be applied to. In the next section, I will describe my work in both approaches that are to be considered together into manifold integration later.

CHAPTER III

THEORETICAL BASES FOR MANIFOLD INTEGRATION

As stated before, manifold learning and data integration are two theoretically key parts of manifold integration. In this chapter, I briefly review my prior work on both approaches that forms the basis for the development of the manifold integration concept and algorithms. Basically this chapter consists of two sections: manifold learning and data integration.

A. Manifold Learning

My prior work on manifold learning has been focused on kernel Isomap [28]. Here I present the algorithm and several applications.

1. Kernel Isomap

Since I use kernel Isomap as the main manifold learning algorithm later in this dissertation, in this section I will review kernel Isomap.

*Part of this chapter is reprinted with permission from "Gesture Recognition based on Manifold Learning" by H. Choi, B. Paulson and T. Hammond, 2008, LNCS 5342 pp. 247-256. Copyright 2009 by Springer.
*Part of this chapter is reprinted with permission from "Sketch Recognition based on Manifold Learning" by H. Choi and T. Hammond, 2008, Association for the Advanced of Artificial Intelligence (AAAI-08), Copyright 2008 by AAAI.
©2005 IEEE. Partially reprinted, with permission, from IEEE Int. Conf. on Development and Learning "Kernel Isomap on Noisy Manifold" H. Choi and S. Choi. For more information go to http://thesis.tamu.edu/forms/IEEE permission note.pdf.
©2010 IEEE. Partially reprinted, with permission, from IEEE Int. Conf. Acoustics, Speech and Signal Processing "Learning Alpha-Integration with Partially-Labeled Data" H. Choi, S. Choi, A. Katake and Y. Choe. For more information go to http://thesis.tamu.edu/forms/IEEE permission note.pdf.
©2010 IEEE. Partially reprinted, with permission, from IEEE Int. Conf. Acoustics, Speech and Signal Processing "Alpha-Integration of Multiple Evidence" H. Choi, A. Katake, S. Choi and Y. Choe. For more information go to http://thesis.tamu.edu/forms/IEEE permission note.pdf.

As stated in the previous chapter, Isomap can be interpreted as a kernel machine. However, the kernel matrix $\boldsymbol{K}$ from Isomap in Eq. (2.1), is not always positive semidefinite. In kernel machines, the kernel matrix should be positive semidefinite. The main idea for kernel Isomap is to make this $\boldsymbol{K}$ a Mercer kernel matrix (which is positive semidefinite) using a constant-shifting method, in order to relate it to KPCA such that the generalization property is preserved.

a. Kernel Isomap Algorithm

Given N objects with each object being represented by an m-dimensional vector $\boldsymbol{x}_i$, $i = 1, \ldots, N$, the kernel Isomap algorithm finds an implicit mapping which places N points in a low-dimensional space. In contrast to Isomap, kernel Isomap can project test data points onto a low-dimensional space using the kernel trick. Kernel Isomap mainly exploits the solution of the additive constant problem, the goal of which is to find an appropriate constant to be added to all dissimilarities (or distances), except for self-dissimilarities, that makes the kernel matrix positive semidefinite.

Given a distance matrix, we calculate Dijkstra's geodesic distances (shortest paths) $\boldsymbol{D}$ [42], and calculate the doubly centered kernel matrix as in Eq. (3.1).

$$\boldsymbol{K}(\boldsymbol{D}^2) = -\frac{1}{2}\boldsymbol{H}\boldsymbol{D}^2\boldsymbol{H}, \tag{3.1}$$

where $\boldsymbol{D}^2 = [D_{ij}^2]$ means the element-wise square of the geodesic distance matrix $\boldsymbol{D} = [D_{ij}]$, $\boldsymbol{H}$ is the centering matrix, given by

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{N}\boldsymbol{e}_N\boldsymbol{e}_N^\top, \tag{3.2}$$

for $\boldsymbol{e}_N = [1 \ldots 1]^\top \in \mathbb{R}^N$. Then, we make the kernel matrix positive semidefinite by adding a constant, c.

$$\widetilde{\boldsymbol{K}} = \boldsymbol{K}(\boldsymbol{D}^2) + 2c\boldsymbol{K}(\boldsymbol{D}) + \frac{1}{2}c^2\boldsymbol{H}, \tag{3.3}$$

where c is the largest eigenvalue of the matrix $\begin{bmatrix} \boldsymbol{0} & 2\boldsymbol{K}(\boldsymbol{D}^2) \\ -\boldsymbol{I} & -4\boldsymbol{K}(\boldsymbol{D}) \end{bmatrix}$ [43] and $\boldsymbol{K}(\boldsymbol{D}) = -\frac{1}{2}\boldsymbol{HDH}$ as shown in Eq. (3.1). Eq. (3.3) implies substituting $\widetilde{\boldsymbol{D}}$ for $\boldsymbol{D}$ in Eq. (3.1), given by

$$\widetilde{D}_{ij} = D_{ij} + c(1 - \delta_{ij}), \tag{3.4}$$

which makes the matrix $\boldsymbol{K}$ positive semidefinite. The term δ_{ij} is the Kronecker delta. Finally, projection mapping $\boldsymbol{Y}$ is given by Eq. (3.5) after eigen-decomposition, $\widetilde{\boldsymbol{K}} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$.

$$\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}}. \tag{3.5}$$

The matrix $\widetilde{\boldsymbol{K}}$ is a Mercer kernel matrix, so its (i, j)th element is represented by

$$\begin{aligned} \widetilde{K}_{ij} &= k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= \phi^\top(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j), \end{aligned} \tag{3.6}$$

where $\phi(\cdot)$ is a nonlinear mapping onto a feature space or a low-dimensional manifold. The coordinates in the feature space can be easily computed by projecting the centered data matrix onto the normalized eigenvectors of the sample covariance matrix in the feature space,

$$\boldsymbol{C} = \frac{1}{N}\left(\boldsymbol{\Phi H}\right)\left(\boldsymbol{\Phi H}\right)^\top, \tag{3.7}$$

where $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_N)]$. Using this mapping, novel data points can be projected on the same feature space as the training data points. See below for details.

b. Generalization Property

As in KPCA, we can project a test data point $\boldsymbol{t}_l$ in the low-dimensional space by

$$[\boldsymbol{y}_l]_i = \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{N} [\boldsymbol{v}_i]_j k(\boldsymbol{t}_l, \boldsymbol{x}_j), \tag{3.8}$$

where $[\cdot]_i$ represents the ith element of a vector and $\boldsymbol{v}_i$ is the ith eigenvector of $\widetilde{\boldsymbol{K}}$. The geodesic kernel for the test data point, $k(\boldsymbol{t}_l, \boldsymbol{x}_j)$, in Eq. (3.8), is constructed by the kernel matrix (Eq. (3.3)) for a set of training data points and geodesic distances, D_{lj}, between test data points $\boldsymbol{t}_l$ and all training data points $\boldsymbol{x}_j$, $j = 1, \ldots, N$. As in Eq. (3.4), D_{lj} is also modified by

$$\widetilde{D}_{lj} = D_{lj} + c. \tag{3.9}$$

Note that the geodesic distance $\widetilde{D}_{lj}$ in the feature space has a Euclidean representation. Hence, the following relation holds:

$$\begin{aligned} \widetilde{D}_{lj}^2 &= [\phi(\boldsymbol{t}_l) - \phi(\boldsymbol{x}_j)]^\top [\phi(\boldsymbol{t}_l) - \phi(\boldsymbol{x}_j)] \\ &= \phi^\top(\boldsymbol{t}_l)\phi(\boldsymbol{t}_l) + \phi^\top(\boldsymbol{x}_j)\phi(\boldsymbol{x}_j) - 2\phi^\top(\boldsymbol{t}_l)\phi(\boldsymbol{x}_j). \end{aligned} \tag{3.10}$$

Taking into account that the mappings on the feature space $\{\phi(\boldsymbol{x}_j)\}$ are centered, we have

$$\frac{1}{N} \sum_{j=1}^{N} \widetilde{D}_{lj}^2 = \phi^\top(\boldsymbol{t}_l)\phi(\boldsymbol{t}_l) + \frac{1}{N} \sum_{j=1}^{N} \phi^\top(\boldsymbol{x}_j)\phi(\boldsymbol{x}_j). \tag{3.11}$$

Then, it follows from Eq. (3.10) and Eq. (3.11) that the kernel for the test data point $\boldsymbol{t}_l$, is computed as

$$\begin{aligned} k(\boldsymbol{t}_l, \boldsymbol{x}_j) &= \phi^\top(\boldsymbol{t}_l)\phi(\boldsymbol{x}_j) \\ &= -\frac{1}{2}\left(\widetilde{D}_{lj}^2 - \phi^\top(\boldsymbol{t}_l)\phi(\boldsymbol{t}_l) - \phi^\top(\boldsymbol{x}_j)\phi(\boldsymbol{x}_j)\right) \\ &= -\frac{1}{2}\left(\widetilde{D}_{lj}^2 - \frac{1}{N}\sum_{i=1}^{N}\widetilde{D}_{li}^2 + \frac{1}{N}\sum_{i=1}^{N}\widetilde{K}_{ii} - \widetilde{K}_{jj}\right). \end{aligned} \tag{3.12}$$

For the detailed derivation and properties of kernel Isomap, see [10].

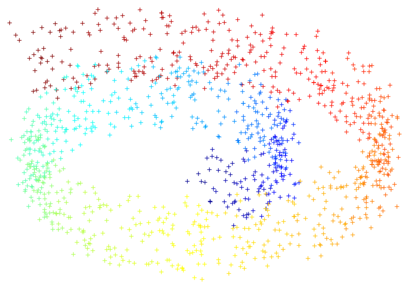
2. Applications

Kernel Isomap has been applied to many data sets showing successful performance in finding a smaller number of dominant factors out of a high-dimensional data set as well as demonstrating projection property for the test data. Here, I present some examples. See [10, 11, 23, 12, 44] for more experiments I have done.

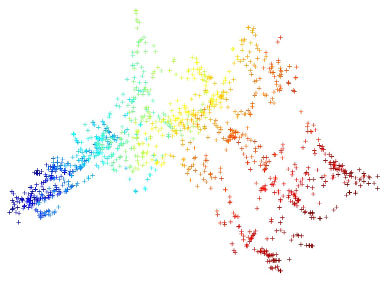
a. Noisy Swiss Roll Data

Noisy Swiss roll data was generated by adding isotropic Gaussian noise with zero mean and variance=0.25 to the original Swiss roll data that was used in Isomap (see Fig. 10 (a)). In the training phase, 1,200 data points were used to construct a neighborhood graph with a neighbor size of $k = 4$. As in Isomap, geodesic distances were computed by calculating shortest paths using Dijkstra's algorithm.

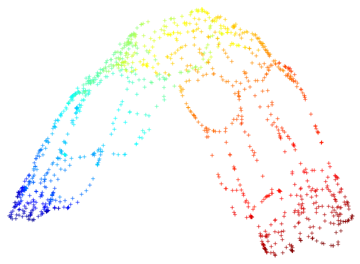
Representative embedding results (onto 3-dimensional feature space) for Isomap and kernel Isomap, are shown in Fig. 10 (b) and (c), respectively, where kernel Isomap is shown to provide a smoother embedded manifold than Isomap. The geodesic kernel matrix modified by a constant-shifting method led kernel Isomap to find a smooth embedded manifold.



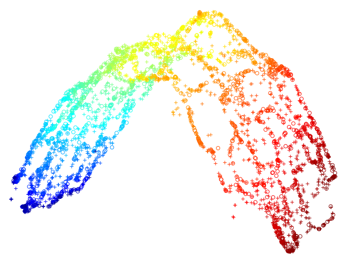
(a)



(b)



(c)



(d)

Fig. 10. Comparison of Isomap with kernel Isomap for the case of *noisy Swiss roll* data. (a) noisy Swiss roll data; (b) projection result by Isomap; (c) projection result by kernel Isomap; (d) projection of test data points using kernel Isomap [28].

The generalization property of the kernel Isomap algorithm is shown in Fig. 10 (d), where 3,000 test data points are correctly embedded while preserving local isometry. The modification by constant-shifting in kernel Isomap improves the embedding while preserving local isometry (see (c)) as well as allowing the projection of test data points onto the same feature space as the training data points (see (d)). Note that Isomap cannot be used for projections as in (d).

b. Handwritten Digits

I applied kernel Isomap to the United States Postal Service (USPS) data set. I used a portion of the USPS data set, which contained digits ‘7’ and ‘9’. Fig. 11 is the digit images projected on the recovered 2-dimensional space. I also applied kernel Isomap to information retrieval tasks. The result of retrieval is shown in Fig. 12. This result shows that kernel Isomap can be used in a retrieval system which searches data points similar to the query point referencing only the selected features.

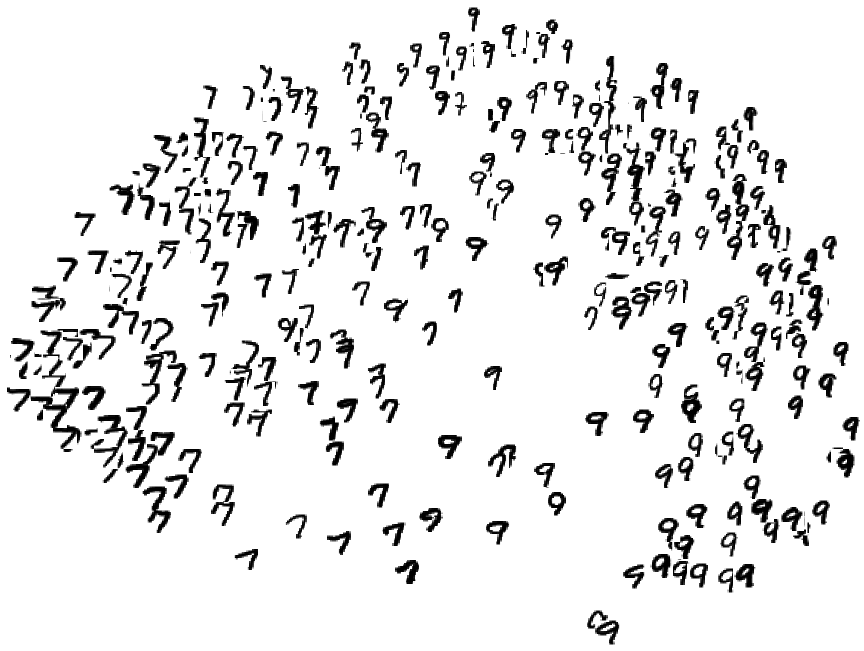
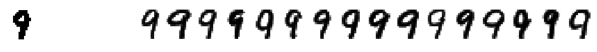


Fig. 11. Digit images in 2-dimensional space. The vertical axis means the height-width ratio and the horizontal axis means how curved the upper portion of digit is [11].

c. HRIR Data

It was recently shown in [45] that a low-dimensional manifold of the head-related impulse responses (HRIRs) can encode perceptual information related to the direction of the sound source. Locally linear embedding (LLE) [8] was applied to find



(a) (b)

Fig. 12. Query point and the retrieval results in USPS handwritten digits. (a) Query image, and (b) the Results which have the same degree of curve in the upper portion as that of the query image [11].

a nonlinear low-dimensional feature space of HRIRs [45]. In this experiment, I used public-domain CIPIC HRIR data set [46] and applied Isomap as well as kernel Isomap, comparing the embedding results of these two methods.

We mainly pay our attention to the HRIRs involving sound sources specified by different elevation angles (see Fig. 13). The database contains HRIRs sampled

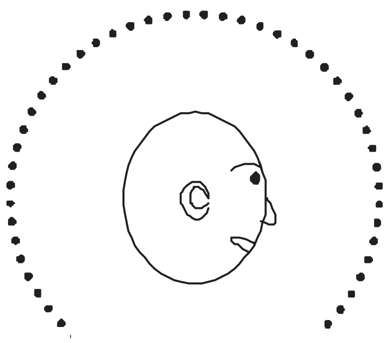


Fig. 13. HRIRs are measured for sound sources at different locations. Locations of sound sources vary according to different elevation angles in interaural-polar coordinates. Elevations are uniformly sampled in $360/64 = 5.625°$ steps from $-45°$ to $230.625°$.

at 1250 points around the head for 45 subjects. Azimuth is sampled from $-80°$ to $80°$ and elevation from $-45°$ to $230.625°$. Each HRIR is a 200-dimensional vector corresponding to a duration of about 4.5ms. The HRIR of the right ear for the 18th subject is shown in Fig. 14.

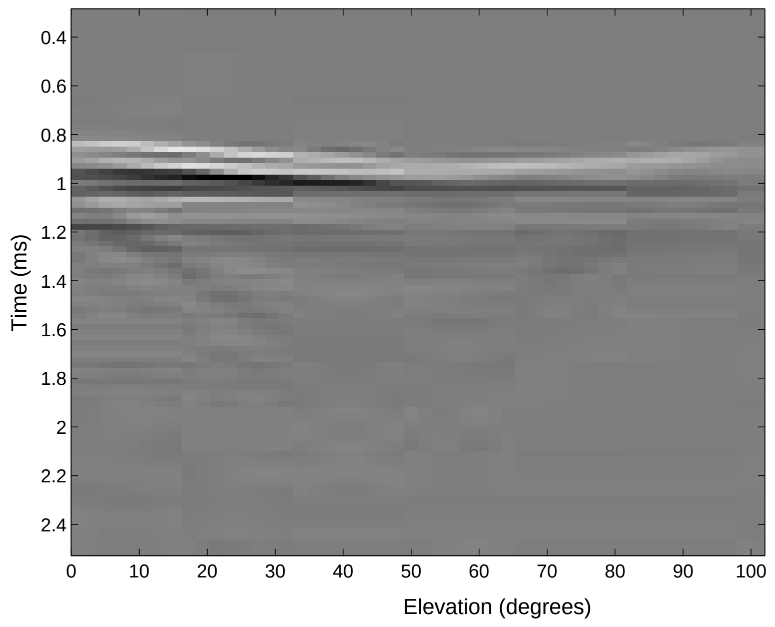


Fig. 14. A HRIR of the right ear for the case of zero azimuth.

Two-dimensional manifolds of HRIRs (with different elevation angles) are shown in Fig. 15 for Isomap and kernel Isomap, where kernel Isomap finds a smooth low-dimensional manifold that encodes perceptual information related to different elevation angles (location of sound), in contrast to Isomap where there are two curves due to noise effect. A one-dimensional manifold computed by kernel Isomap is shown in Fig. 16, where points projected onto the largest eigenvector of the geodesic kernel matrix $\widetilde{\boldsymbol{K}}$ used in kernel Isomap, are plotted with respect to elevation angles.

d. Spoken Letters Data

I applied Isomap and kernel Isomap to the 'Isolet Spoken Letters Data' which is available from 'University of California, Irvine' (UCI) Machine Learning Repository [47], that was also used recently in [48, 49]. I used a portion of Isolet DB, which contains utterances of 30 subjects who spoke the name of each letter of English alphabet twice. Thus the number of data points are 1,560 ($= 26 \times 2 \times 30$). Attributes

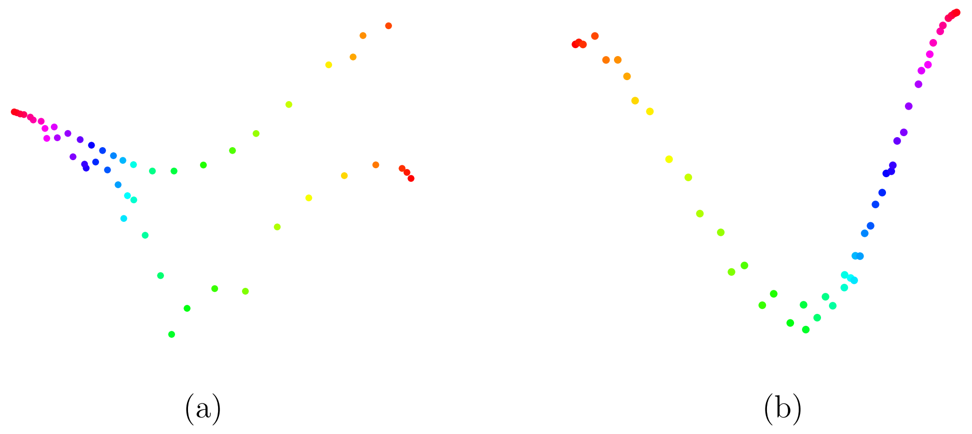


Fig. 15. Two-dimensional manifolds of HRIRs. (a) Isomap; (b) Kernel Isomap. Kernel Isomap finds a smooth low-dimensional manifold of HRIRs, in contrast to Isomap.

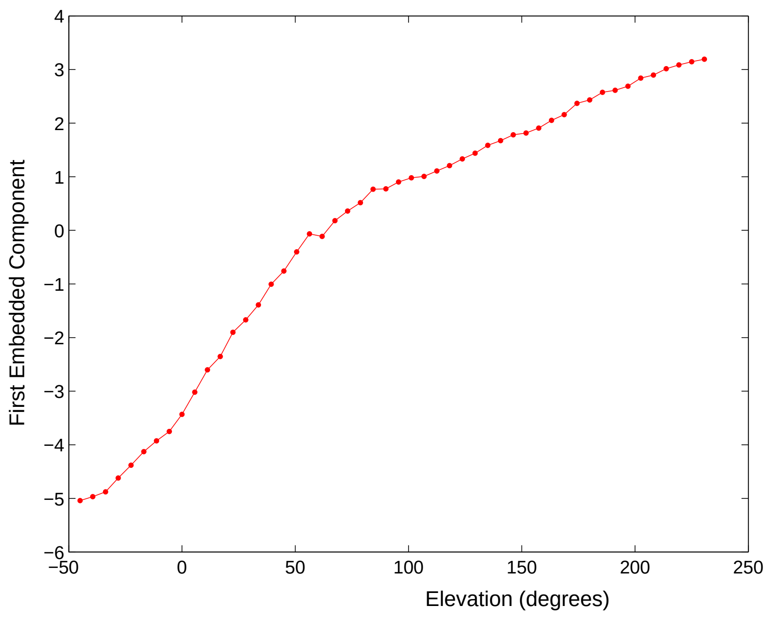


Fig. 16. One-dimensional manifold of HRIR computed by kernel Isomap.

(features) are 617, including spectral coefficients, contour features, sonorant features, pre-sonorant features, and post-sonorant features. These utterances are in a high-dimensional space, however, it is expected that distinctive phonetic dimensions are few. Two-dimensional manifolds of Isolet data found by Isomap and kernel Isomap, are shown in Figs. 17 and 18, where one can see that kernel Isomap shows slightly better cluster structure, compared to Isomap. Even though kernel Isomap is not able to discriminate clearly every English letters in a two-dimensional manifold, it still shows better performance over Isomap.

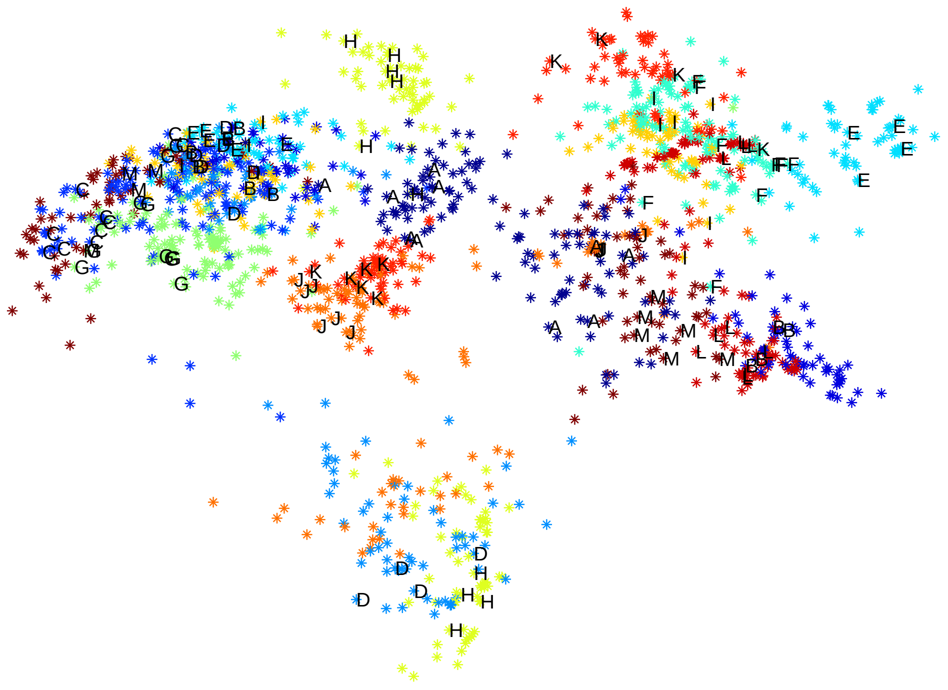


Fig. 17. Isomap result on Isolet data. Two-dimensional manifold of spoken letters data by Isomap.

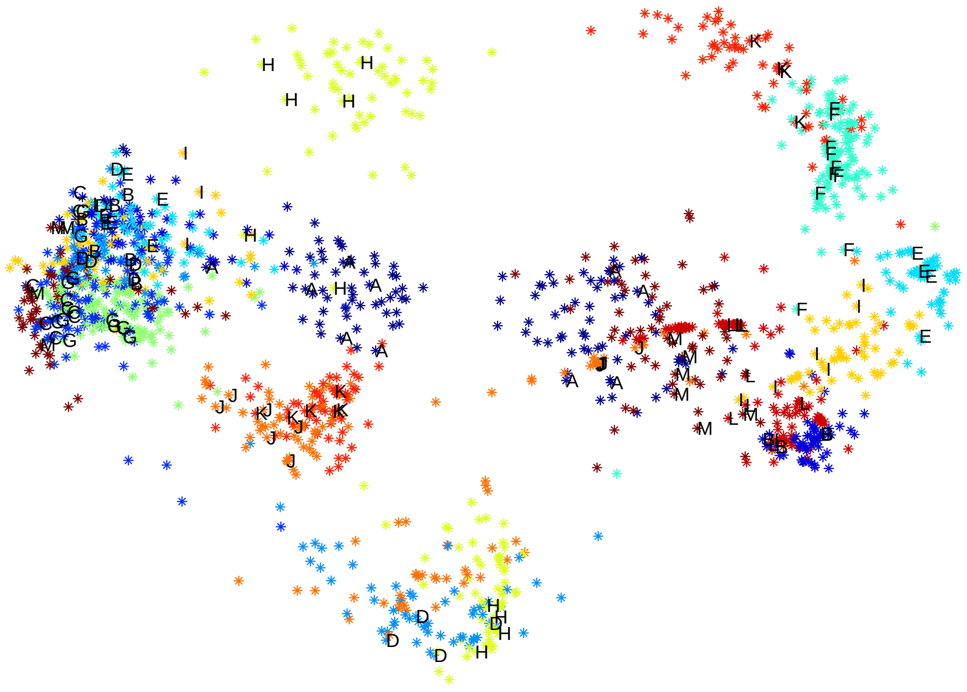


Fig. 18. Kernel Isomap result on Isolet data. Two-dimensional manifolds of spoken letters data by kernel Isomap.

e. Sketch Data

Current feature-based sketch recognition systems use human-chosen features to perform recognition. Effective features for classification can also be automatically learned and chosen by the computer. In other recognition domains, such as face recognition, manifold learning methods have been found to be good nonlinear feature extractors. Few manifold learning algorithms, however, have been applied to sketch recognition. Here, I developed a new algorithm for multi-stroke sketch recognition, which is based on kernel Isomap.

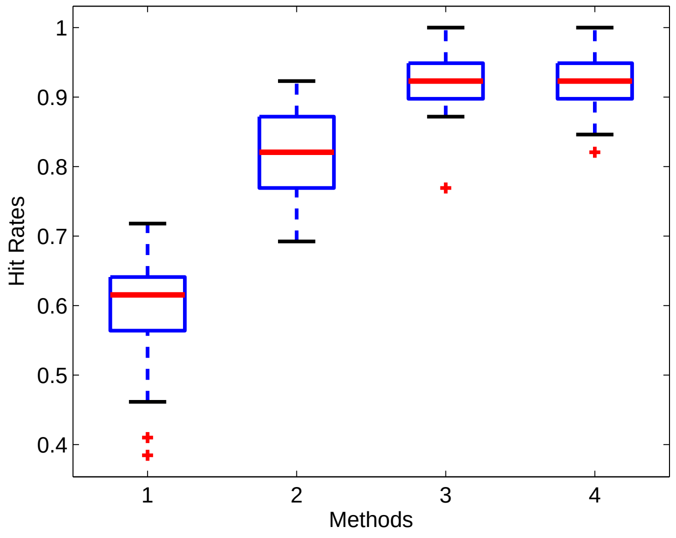


Fig. 19. Sketch recognition results. The classification accuracies of three methods for mathematical symbols (‘+’, ‘−’, ‘×’, ‘/’, ‘=’, ‘sin’, ‘cos’ and ‘tan’): (Method 1) Rubine’s method (Method 2) $1 recognizer (Method 3) kernel Isomap and (Method 4) weighted kernel Isomap. The average hit rates were 60.46%, 82.41%, 92.77% and 92.41%, respectively [12]. The box plot shows the lower quartile, median, and upper quartile values and the Whisker plot shows the most extreme values within 1.5 times the interquartile range from the ends of each box. The red ‘+’ marks are outliers.

I applied kernel Isomap to 8 different mathematical symbols (‘+’, ‘−’, ‘×’, ‘/’,

'=', 'sin', 'cos' and 'tan'), drawn by 10 subjects, where each class of each person contained 5 characters, each of which was drawn in one stroke. In order to show the advantages of using kernel Isomap, I compared the proposed method with the algorithm in [50] and the $1 recognizer in [51]. For more robust results, I executed 10-fold cross validation 50 times. Fig. 19 compares the classification hit rates by the four approaches: (1) Rubine's method, (2) $1 recognizer (3) kernel Isomap and (4) weighted kernel Isomap. Here kernel Isomap shows the best results. In this figure, kernel Isomap has better performance in classification than other methods. This approach can be applied directly to any other gesture recognition.

3. Summary

In sum, I presented the kernel Isomap algorithm where I constructed a geodesic Mercer kernel matrix through a constant-shifting method. Kernel Isomap was explicitly related to kernel PCA, providing the generalization property such that test data points were able to be embedded in the associated low-dimensional space by a geodesic kernel mapping. Numerical experiments with several data sets such as noisy Swiss roll data, USPS handwritten digits, HRIR, Isolet spoken letters, and sketch data verified the useful properties of kernel Isomap.

B. Data Integration

For data integration, I use a general framework called α-integration, proposed by Amari [35]. So, in this section, I will briefly review how I applied α-integration to other algorithms so that those algorithms become generalized. Also, I expand α-integration by using learning algorithms of the parameters in the integration.

1. α-Integration on Evidence Theory

In order to test the utility of α-integration in data integration, I applied α-integration to evidence theory (or Dempster-Shafer theory). Two approaches were evaluated: (1) the D-S combination rule interpreted as a special case of α-integration and generalized, and (2) the previous averaging methods for ET generalized. Note that both generalization approaches can be applied together at the same time. Here, I briefly describe evidence theory and show how to apply α-integration to evidence theory. See [18, 19, 52] for details on evidence theory.

a. Evidence Theory

Evidence theory (ET) is a mathematically well defined theory for handling conflicts between different bodies of evidence. It is conceptually the same as Bayesian theory except that it uses epistemic (subjective) uncertainty [53]. The advantages of ET include its flexibility in theory and easy implementability. As the first evidence theory, Dempster and Shafer proposed the D-S combination rule [18, 19]. Here, I give a brief review of it. For details, see [52] and references therein.

Let Θ be a set of hypotheses, and m be a basic belief assignment (BBA) which is a function from a subset of Θ to $[0, 1]$ with the following properties.

$$\begin{aligned} m(\phi) &= 0, \\ \sum_{A \subseteq \Theta} m(A) &= 1. \end{aligned} \tag{3.13}$$

When two bodies of evidence m_1 and m_2 are given, the D-S combination rule for $\tilde{m}(A)$ is defined by

$$\tilde{m}(A) = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{1 - K}, \tag{3.14}$$

where

$$K = \sum_{B\cap C=\phi} m_1(B)m_2(C). \tag{3.15}$$

Here, K indicates basic probability associated with conflict. This can be easily expanded to more than two bodies of evidence.

As pointed out in [54], in some cases the D-S combination rule is against our intuitive reasoning. For example, when only one evidence has 0 belief but all others have 1 belief, still the combination is 0. To overcome this weakness, ET has been improved in some directions, and the averaging approach is known to be better than others [52].

In [54], Murphy proposed an averaging rule to avoid nonintuitive combination in ET. When there are N bodies of evidence, Murphy's rule first calculates the average of each hypothesis for the evidence and applies the D-S combination rule with the averages $N-1$ times. That is, Eq. (3.14) can be modified as follows.

$$\tilde{m}(A) = \frac{\sum_{B\cap C=A} \bar{m}(B)\bar{m}(C)}{1-\bar{K}}, \tag{3.16}$$

where

$$\bar{K} = \sum_{B\cap C=\phi} \bar{m}(B)\bar{m}(C). \tag{3.17}$$

Here, $\bar{m}(B)$ and $\bar{m}(C)$ are the averages of evidence for B and C, respectively. Here, all bodies of evidence have the same importance with the same weight in calculating the average, which is not always the case. Instead of a simple averaging rule, some other researchers have tried a weighted sum of bodies of evidence [55, 56].

b. α-Integration on D-S Combination Rule

I interpreted the combination rule as a kind of averaging and proposed a generalized method which can avoid the problems in the original D-S combination rule has.

The D-S combination rule in Eq. (3.14) can be rewritten by

$$\tilde{m}(A) = \frac{\sum_{B\cap C=A} m_{12}(BC)^2}{1-K}, \tag{3.18}$$

where

$$m_{12}(BC) = \sqrt{m_1(B)m_2(C)}. \tag{3.19}$$

This is the geometric mean of $m_1(B)$ and $m_2(C)$.

Now using the fact that the geometric mean is a special case of α-integration with $\alpha = 1$, we generalize it as follows.

$$\tilde{m}(A) = \frac{\sum_{B\cap C=A} m_{\alpha,12}(BC)^2}{1-K}, \tag{3.20}$$

where

$$m_{\alpha,12}(BC) = f_\alpha^{-1}\left[f_\alpha(m_1(B)) + f_\alpha(m_2(C))\right]. \tag{3.21}$$

Eq. (3.20) with Eq. (3.21) is a more generalized combination rule, which can avoid the nonintuitive problem stated above by choosing the α value carefully.

c. α-Integration on Averaging Methods

Several averaging rules have been proposed to overcome the problem of the D-S combination rule. I also generalized the previous averaging methods with α-integration. By applying α-integration, the previous averaging methods which are arithmetic means with different weights can be replaced with

$$\bar{m}_\alpha(A_i) = f_\alpha^{-1}\left(\sum_{j=1} w_j f_\alpha(m_j(A_i))\right), \tag{3.22}$$

where w_j are obtained as in the previous averaging methods. Even though in the experiments below I used one example of the averaging rule, probabilistic weight method in [56], the approach can be applied to any other averaging methods.

After calculating the α-integration of all the hypotheses, I applied the D-S combination rule $N-1$ times as other averaging methods do. With the new averaging rule in Eq. (3.22), the combination rule in Eq. (3.16) is modified as follows.

$$\tilde{m}_\alpha(A) = \frac{\sum_{B\cap C=A} \bar{m}_\alpha(B)\bar{m}_\alpha(C)}{1-\bar{K}_\alpha}, \tag{3.23}$$

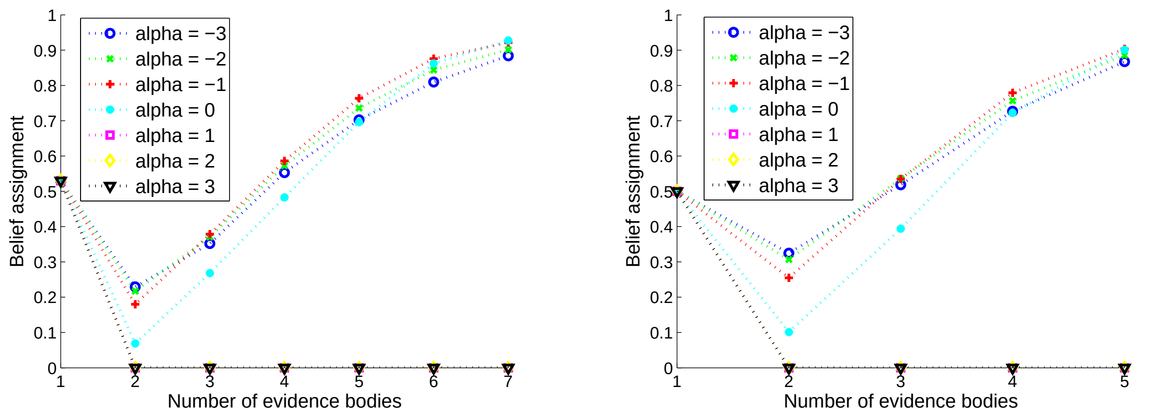
where

$$\bar{K}_\alpha = \sum_{B\cap C=\phi} \bar{m}_\alpha(B)\bar{m}_\alpha(C). \tag{3.24}$$

d. Experiments

In order to show the useful properties of the generalized versions, I carried out experiments with two different data sets used in two previously published papers: (a) the data set in [57] and (b) the data set in [55]. Both data sets are for target recognition systems where there is one true target (for both cases, hypothesis A is the true target) with multiple evidence. I tested my proposed methods with different α values. Note that the D-S combination rule and the previous averaging methods are a special case with specific α values. For the comparison between the previous averaging methods, see [56].

Application of α-Integration on D-S combination rule: In Fig. 20, we can see that the original D-S combination rule which is a special case of the proposed method with $\alpha = 1$ is not the best way to combine the bodies of evidence. Rather, when α is around -1, the combination result is more desirable. Note that the results for hypothesis A with $\alpha = 1, 2, 3$ are zero after evidence 2 no matter how high other BBAs are, because it ignores all the conflicting evidence which can be interpreted as an AND operation as mentioned in [52]. Since α-integration is a monotonically



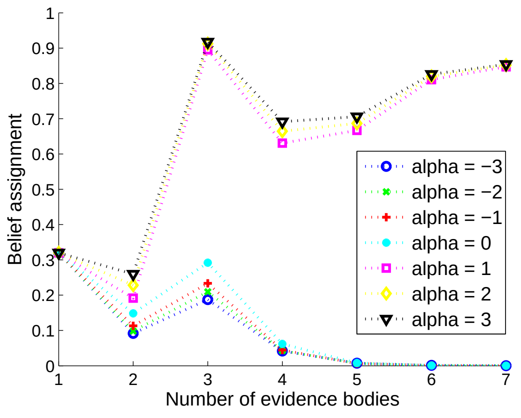
(a) (b)

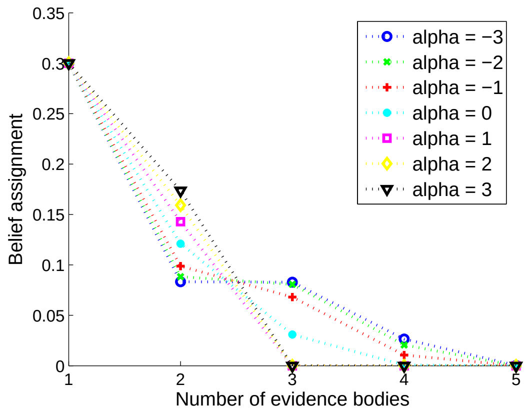
Fig. 20. The D-S combination rule is generalized with several α values. The belief assignments of the hypothesis A with different α values are shown for (a) Xu's data [57] and (b) Yong's data [55].

decreasing function, once the result of a certain α value is the minimum, then the bigger α values have the same minimum results as shown in Fig. 20. On the other hand, in the cases of $\alpha < 1$, the results are not against our intuitive reasoning.

Application of α-Integration on averaging methods: When an averaging rule is generalized with α-integration, the belief assignments of the hypothesis A are similar to Fig. 20. Fig. 21 shows the belief value of the hypothesis C which is expected to decrease as the bodies of evidence are added. In Fig. 21 (a), smaller values for α than -1 are generally better and in Fig. 21 (b), roughly speaking, $\alpha = 0$ seems better than others. That is, the best α value depends on each situation and we can wisely choose the value to get a more desirable result from multiple evidence. Also, note that even though I used only integer values for α, any real number would be possible.

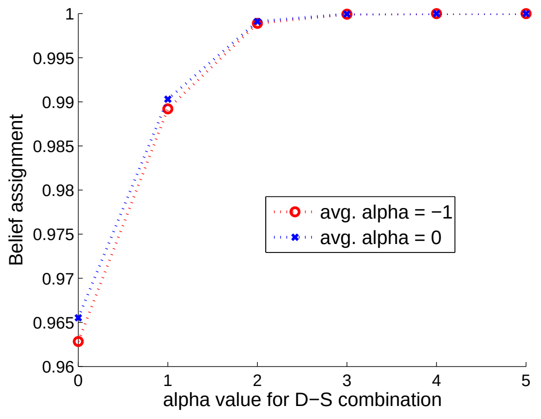
Note that we can apply α-integration to both the D-S combination rule and an averaging rule. In Fig. 22, to make the figures clearer, I tested two α values for the averaging rules and six α values for the D-S combination rule. Fig. 22 shows

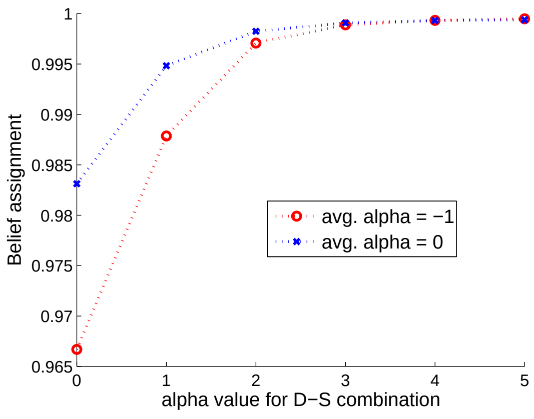




(a) (b)

Fig. 21. One averaging rule for ET is generalized with several α values. The belief assignments of the hypothesis C with different α values for the averaging rule are shown for (a) Xu's data [57] and (b) Yong's data [55].





(a) (b)

Fig. 22. ET generalized by α-integration. The converged belief assignments of the hypothesis A from several α to both the D-S combination rule and the averaging rule for (a) Xu's data and (b) Yong's data.

the converged belief assignment after combining all the evidence. We can see with different α values, the results are changing. That is, this generalized method gives us more room to improve the performance by fitting the α value. Note that the previous averaging method is a special case with $\alpha = -1$ (arithmetic mean) for the averaging rule and $\alpha = 1$ (geometric mean) for the combination rule. With different α values, we could get better results.

See [58] for details and experimental results that confirm the effectiveness of the algorithm.

2. α-Integration on Linear Discriminant Analysis

As another application of α-integration, I generalized linear discriminant analysis (LDA) [59] by applying α-integration to calculate the between-scatter and within-scatter matrices in the LDA formulation. The ways to calculate these scatter matrices can be interpreted as a special averaging method. In this section, I also discuss the problems with LDA that is the reason why we need to generalize LDA.

a. Linear Discriminant Analysis

Since there is an abundance of published work on LDA, here it is suffice to describe the objective function of LDA. See [4, 60, 61, 62, 63] for the detail and the related work.

For the case of two classes, assume that there are n d-dimensional samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ with n_1 samples in class C_1 and n_2 samples in class C_2. The class mean vectors, $\boldsymbol{m}_i$, and global mean vector, $\boldsymbol{m}$, are calculated by

$$\boldsymbol{m}_i = \frac{1}{n_i}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}, \text{ for } i = 1,2, \tag{3.25}$$

$$\boldsymbol{m} = \frac{1}{n}\sum_{i}^{n}\boldsymbol{x}_i. \tag{3.26}$$

The projection of $\boldsymbol{x}$ is obtained by

$$y = \boldsymbol{w}^\top\boldsymbol{x}, \tag{3.27}$$

where $\boldsymbol{w}$ is a direction vector for projection. Then the objective function is basically the ratio of the variance between the classes to the variance within the classes on the projected space, and defined by

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{\sigma^2_{\text{between}}}{\sigma^2_{\text{within}}} = \frac{\boldsymbol{w}^\top\boldsymbol{S}_{\text{B}}\boldsymbol{w}}{\boldsymbol{w}^\top\boldsymbol{S}_{\text{W}}\boldsymbol{w}}, \tag{3.28}$$

where

$$\boldsymbol{S}_{\text{B}} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^\top, \tag{3.29}$$

$$\boldsymbol{S}_{\text{W}} = \boldsymbol{S}_1 + \boldsymbol{S}_2, \tag{3.30}$$

$$\boldsymbol{S}_i = \sum_{\boldsymbol{x}\in C_i}(\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^\top.$$

For more than two classes, say c classes, the objective function is slightly changed as follows.

$$\boldsymbol{J}(\boldsymbol{W}) = \frac{\boldsymbol{W}^\top\boldsymbol{S}_{\text{B}}\boldsymbol{W}}{\boldsymbol{W}^\top\boldsymbol{S}_{\text{W}}\boldsymbol{W}}, \tag{3.31}$$

where

$$\boldsymbol{S}_{\mathrm{W}} = \sum_{i}^{c} \boldsymbol{S}_i, \tag{3.32}$$

$$\boldsymbol{S}_{\mathrm{B}} = \sum_{i}^{c} n_i(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^{\top}, \tag{3.33}$$

where $\boldsymbol{S}_{\mathrm{W}}$ and $\boldsymbol{S}_{\mathrm{B}}$ use arithmetic mean of $\boldsymbol{S}_i$ and $(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^{\top}$, respectively.

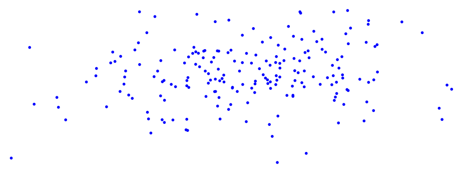
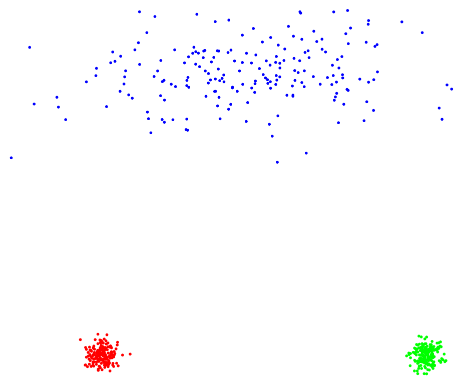



Fig. 23. Toy data set with three classes for LDA. The three classes have different scales of variance. Especially the blue class has a much greater variance than the other two classes. If we try to reduce the within-scatter on the projected space as the original LDA does, the scatterness in the upper class (blue) would affect to the projection more than that of the other two classes.

When the scales of $\boldsymbol{S}_i$ are different in each class as in Fig. 23, the objective function in Eq. (3.31) considers the bigger covariance matrix more seriously on the projected space so that the variance of the blue class will play the major role in LDA as shown in Fig. 25 (a), which is not always the best projection. We want to reduce the effect of the variance of the bigger classes than other classes for better classification performance in lower-dimensional space.

Likewise, when $\boldsymbol{m}_i$ are not equidistant from $\boldsymbol{m}$ as in Fig. 24, the objective function in Eq. (3.31) considers the longer distance more seriously on the projected space.

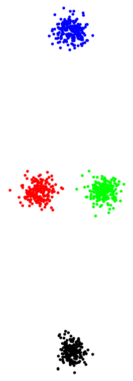


Fig. 24. Toy data sets with four classes for LDA. All classes have different distances to the global mean and the difference is large. The green and the red classes (the two middle ones) are very close to the global mean but the blue and the black ones are very far from the mean. If we try to preserve (or maximize) the distance between classes in a one-dimensional space, the distance of classes far away from the center would affect to the projection more than the other two classes near the center. Note that all four classes have the same variance.

In such cases, regardless of the classification ability on the projected space, LDA tries to preserve the distance on the space as shown in Fig. 26 (a), which is not the best way for projection. We want to separate all the classes as much as possible on the projected space rather than preserving the distances.

Since the goal of LDA is to extract more informative projected space for classification instead of preserving the distance on the projected space, we need to put more weight on the smaller distances or "within-scatter". This is what α-integration can do. Therefore, both scatter matrices, $\boldsymbol{S}_{\mathrm{W}}$ and $\boldsymbol{S}_{\mathrm{B}}$, can be generalized with α-integration.

b. α-Integration on the Scatter Matrices

I generalized the within-scatter matrix $\boldsymbol{S}_{\mathrm{W}}$ and the between-scatter matrix $\boldsymbol{S}_{\mathrm{B}}$ for multi-classes cases with α-integration. As applied in evidence theory, α-integration can be applied simply and the generalized equations are as follows.

$$\boldsymbol{S}_{\mathrm{W}} = f_\alpha^{-1}\left(\sum_i^c f_\alpha(\boldsymbol{S}_i)\right), \tag{3.34}$$

$$\boldsymbol{S}_{\mathrm{B}} = f_\alpha^{-1}\left(\sum_i^c n_i f_\alpha\left((\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^\top\right)\right), \tag{3.35}$$

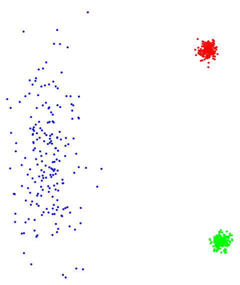
where $f_\alpha(\cdot)$ is the α-function in Eq. (2.11) element-wisely applied to a matrix. The original LDA is a special case of the generalized method with $\alpha = -1$ for both integration.

Note that α-integration works with positive sources and the scatter matrices might have negative values. We can add a constant value to the elements of the scatter matrices so that all the values become positive and then subtract the constant from the integrated matrix. In the current work, I made the constant value by negating the minimum value of the sources. We may need a more careful estimation of this constant which affects the performance.

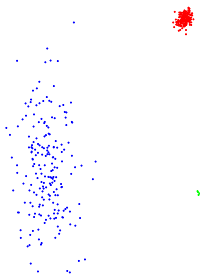
c. Experiments

Fig. 25 shows projected spaces of the data in Fig. 23. The project direction of the data set rotates as the α value increases. Even though the plots are shown in 2-dimensional space, the first component (horizontal axis) is more important in pursuing the effectiveness of dimensionality reduction. In the first component, the generalized LDA (b-f) has room for improvement especially with $\alpha = 1$. In the original LDA in (a), however, the red and the green classes are completely overlapped in the first component. In this data set, each class mean is equidistant to the global mean so that the generalized integration of the within-scatter matrices affects the projection more dominantly than the one based on the between-scatter matrices.

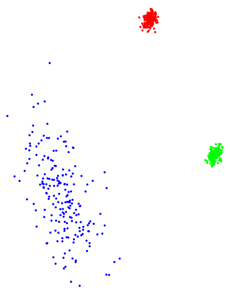
Fig. 26 shows the projected spaces of the data in Fig. 24. As in Fig. 25, the



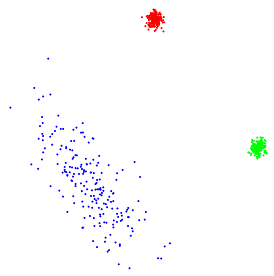
(a) $\alpha = -1$



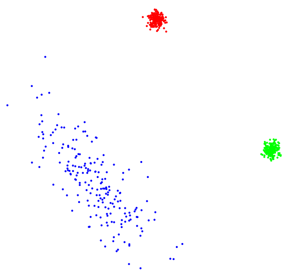
(b) $\alpha = 0$



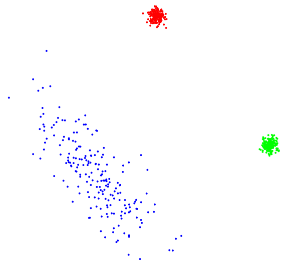
(c) $\alpha = 1$



(d) $\alpha = 2$



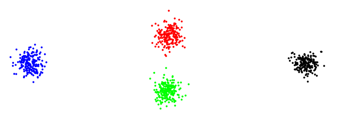
(e) $\alpha = 3$

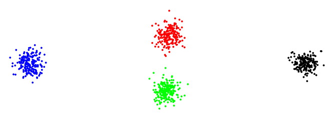


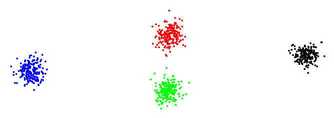
(f) $\alpha = 4$

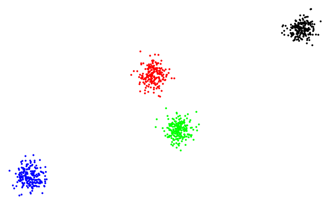
Fig. 25. Original LDA vs. generalized LDA on the three classes data. (a) Conventional LDA where the red and the green classes are completely overlapped in the first component space (horizontal axis). (b-f) Generalized LDA by α-integration in the scatter matrices with $\alpha = 0, 1, 2, 3, 4$. In (c) with $\alpha = 1$, the red and the green classes are well separated in the first component space as well as the blue one.

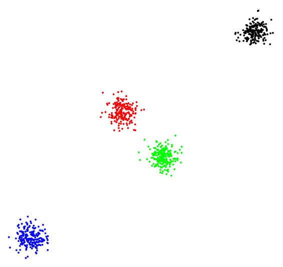
horizontal axis is more important. The projection direction rotates along with the α value. The generalized LDA method has better clustering results especially with $\alpha = 4$ while the conventional LDA in (a) has totally overlapping classes. In this data set, with the isotropic variance, the generalized integration of the between-scatter matrices affects the projection more dominantly than the one based on the within-scatter matrices.


(a) $\alpha = -1$


(b) $\alpha = 0$


(c) $\alpha = 1$


(d) $\alpha = 2$


(e) $\alpha = 3$

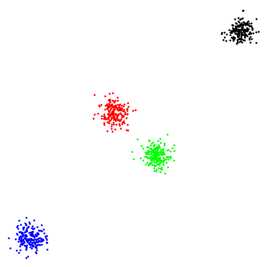

(f) $\alpha = 4$

Fig. 26. Original LDA vs. generalized LDA on the four classes data. (a) Conventional LDA where the red and the green classes are completely overlapped in the first component space (horizontal axis). (b-f) Generalized LDA by α-integration in the scatter matrices with $\alpha = 0, 1, 2, 3, 4$. When α is 4, the red and the green classes are well separated in the first component space as well as the blue and the black ones.

From the two experiments, we can see that the optimal α value depends on each data set. The within-scatter and the between-scatter matrices determine which value would be better for α. That is, we cannot determine the value in advance and should learn the value from each data set. Also, conventional LDA does not always find the best projection for all kinds of data sets.

3. Learning α-Integration

There is an unresolved critical issue with α-integration. In most existing works [64, 35, 65, 66] including the description in the previous sections, the value of α as well as the weight vector $\boldsymbol{w}$ is given in advance rather than learned. Even though the theory generalizes some specific stochastic models into the α-family, in practice, we have to specify the α value when we use this integration with a specific data set. It means that we have to decide which model is going to be used in advance. For example, if we fix α to 1 in advance, we get to use geometric mean from the exponential family which is a special case of α-integration from α-family. Then there is no actual benefit in generalizing an arbitrary stochastic model. If we can find out the α value automatically, the size of the model that is considered gets larger than that of a specific stochastic model. Instead of one model specified by α, the integration gets more accurate in terms of α-divergence since it searches over all the models.

To overcome these issues, I proposed new algorithms to learn α-integration from data when the sources and only a few integrated target values are available. There are two kinds of parameters: α and $\boldsymbol{w}$. Given a couple of training data points, I first define an objective function with respect to α and $\boldsymbol{w}$ and then derive two algorithms to learn the parameters based on gradient descent. The update procedure consists of two parts: (1) α-integration and (2) parameter updating. These parts are executed iteratively because parameter update equations include the α-integration equation in themselves.

The problem that I consider is as follows. Given M positive measurements, $m_i(x)$, where $i = 1, \cdots, M$ and $m_i(x) > 0$ for all x, our task is to determine an α-integration $\widetilde{m}(x)$ when target values for $\widetilde{m}(x)$ are partially available. We assume that either α or $\boldsymbol{w}$ is known in advance. In other words, given w_i's (or fixed in advance), we learn the parameter α such that the optimal α-integration $\widetilde{m}(x)$ is as close to partially available target values as possible. If α is given, then we learn parameters w_i's under the same criterion.

Optimal α-integration has the form

$$\widetilde{m}(x) = \begin{cases} \left\{\sum_i w_i m_i(x)^{\frac{1-\alpha}{2}}\right\}^{\frac{2}{1-\alpha}}, & \alpha \neq 1, \\ \exp\left\{\sum_i w_i \log m_i(x)\right\}, & \alpha = 1, \end{cases} \tag{3.36}$$

which is derived by applying calculus of variation to solve $\frac{\partial \mathcal{J}_\alpha[\widetilde{m}(x)]}{\partial \widetilde{m}(x)} = 0$ for $\widetilde{m}(x)$, where $\mathcal{J}_\alpha[\widetilde{m}(x)]$ is of the form of Eq. (2.13).

a. Learning α

Given M measures $m_i(x_k)$ where $i = 1, ..., M$ and $k = 1, ..., N$, let S_N be the number of targets ($S_N \ll N$). With true target values t_j (the integrated values) where $j = 1, \cdots, S_N$, the objective function for α, $\mathcal{J}(\alpha)$ is defined as

$$\mathcal{J}(\alpha) = \sum_{j=1}^{S_N} (t_j - \widetilde{m}(x_j))^2, \tag{3.37}$$

which is expected to be minimized. Then, we take a derivative of Eq. (3.37), as follows.

$$\frac{\partial \mathcal{J}(\alpha)}{\partial \alpha} = -2 \sum_j \left(t_j - \widetilde{m}(x_j)\right) \frac{\partial \widetilde{m}(x_j)}{\partial \alpha}, \tag{3.38}$$

where

$$\begin{aligned}\frac{\partial \widetilde{m}(x)}{\partial \alpha} &= \frac{2\widetilde{m}(x)}{1-\alpha}\left\{\frac{\log(\sum_i w_i f_\alpha(m_i(x)))}{1-\alpha} + \frac{\sum_i w_i \frac{\partial f_\alpha(m_i(x))}{\partial \alpha}}{\sum_i w_i f_\alpha(m_i(x))}\right\},\\ \frac{\partial f_\alpha(u)}{\partial \alpha} &= -\frac{1}{2}\log(u)u^{\frac{1-\alpha}{2}}.\end{aligned}$$

Finally, we can use gradient descent to update α given by

$$\Delta\alpha = -\eta_\alpha \frac{\partial \mathcal{J}(\alpha)}{\partial \alpha}, \tag{3.39}$$

where η_α is the learning rate for α. Note that since $\widetilde{m}(x)$ is a monotonically decreasing function with respect to α, Eq. (3.37) is a convex function and Eq. (3.39) always converges to the global optimizer.

b. Learning $\boldsymbol{w}$

In order to learn $\boldsymbol{w}$, we can use Eq. (3.37) as an objective function for $\boldsymbol{w}$ to get a gradient algorithm based on the derivative of the objective function with respect to $\boldsymbol{w}$. Each element of the gradient vector $\frac{\partial \mathcal{J}(\boldsymbol{w})}{\partial \boldsymbol{w}}$ is obtained by

$$\frac{\partial \mathcal{J}(\boldsymbol{w})}{\partial w_i} = -2\sum_j (t_j - \widetilde{m}(x_j))\frac{\partial \widetilde{m}(x_j)}{\partial w_i}, \tag{3.40}$$

where

$$\frac{\partial \widetilde{m}(x)}{\partial w_i} = \begin{cases} \frac{2}{1-\alpha}\left(\frac{\widetilde{m}(x) f_\alpha((m_i(x))}{\sum_k w_k f_\alpha(m_k(x))}\right), & \alpha \neq 1 \\ \widetilde{m}(x)\log m_i(x), & \alpha = 1 \end{cases}.$$

Then, the update rule is given by

$$\Delta\boldsymbol{w} = -\eta_w \frac{\partial(\mathcal{J}\boldsymbol{w})}{\partial \boldsymbol{w}}, \tag{3.41}$$

where η_w is the learning rate for $\boldsymbol{w}$.

However, for $\boldsymbol{w}$, as long as the α value is around 1, we can have an algebraic solution which is approximately the same as the solution from Eq. (3.41). We can slightly modify the objective function in Eq. (3.37) to look like

$$\mathcal{J}_2(\boldsymbol{w}) = \sum_{j=1}^{S_N} (f_\alpha(t_j) - \sum_i w_i f_\alpha(m_i(x_j))^2. \tag{3.42}$$

Note that this objective function does not have $\widetilde{m}(x)$ in the equation in contrast to $\mathcal{J}$, so $\boldsymbol{w}$ can be obtained without any iterative interaction with the update rule for $\widetilde{m}(x)$. Actually, $\mathcal{J}_2$ is a variation of $\mathcal{J}$ where f was taken off, where f has nothing to do with $\boldsymbol{w}$. Then we use the least square method for optimization as below (pseudo-inverse)

$$\boldsymbol{w} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1}\boldsymbol{\Phi} f_\alpha(\boldsymbol{t}), \tag{3.43}$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix} f_\alpha(m_1(\boldsymbol{s}_t)) \\ f_\alpha(m_2(\boldsymbol{s}_t)) \\ \cdots \\ f_\alpha(m_M(\boldsymbol{s}_t)) \end{bmatrix}.$$

Note that usually the gradient method needs a series of updates until it converges, whereas the least square method needs just one step of calculation.

c. Experiments

In order to show the effectiveness of the proposed algorithm, I carried out experiments with two different data sets: (a) a synthetic data set with two curves and a few true integrated values and (b) monthly average temperatures of multiple cities from www.cityrating.com. See [67] for more detail.

Synthetic data set: Given 2 curves and only 5 target values from the true integration, I tried to find the true integration curve which has an optimal α or $\boldsymbol{w}$ for the target values when one of the parameters is given. In Fig. 27, I used one weight vector, [0.4 0.6] with different true α values, 3 and -2 (unknown to the algorithm). The α learning procedure is based on two black solid curves and 5 points randomly selected from the true curve with the fixed weights. The learning trajectory is shown in Fig. 28. In Fig. 27, the blue dotted curves are the estimated curve $\widetilde{m}(x)$ and the sum of the squared errors between the estimated curve and the true curve are 9.85e-11 and 9.91e-11 for (a) and (b), respectively. With different weight vectors like [0.9 0.1], the proposed algorithm worked perfectly too (data not shown).

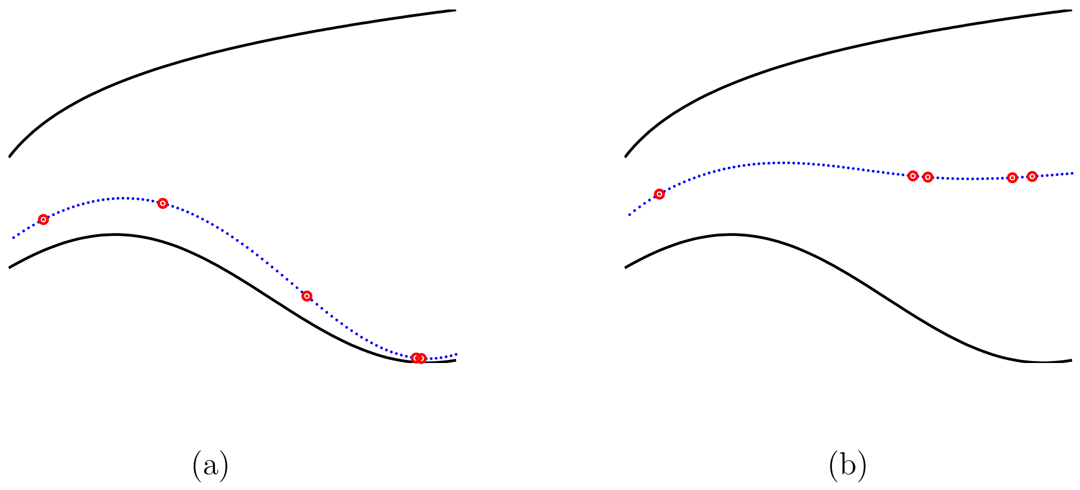


Fig. 27. Estimated curves by learning α-integration. The true curves were generated with different α, but with the same weight vector [0.6 0.4]. The true α values are 3 and -2 for (a) and (b), respectively. Two black curves in each figure are m_i and the blue dot curve is the estimated one $\widetilde{m}(x)$ for each case. Five red dots are target values.

Given α, the proposed algorithm found out $\boldsymbol{w}$ perfectly with any given hidden true weights. The squared errors in the estimated curve and in the estimated weight vector are almost 0 as shown in Table I. In this data set, I did not assume any noise added to the true values.

City temperature data set: To test the proposed method in a real world

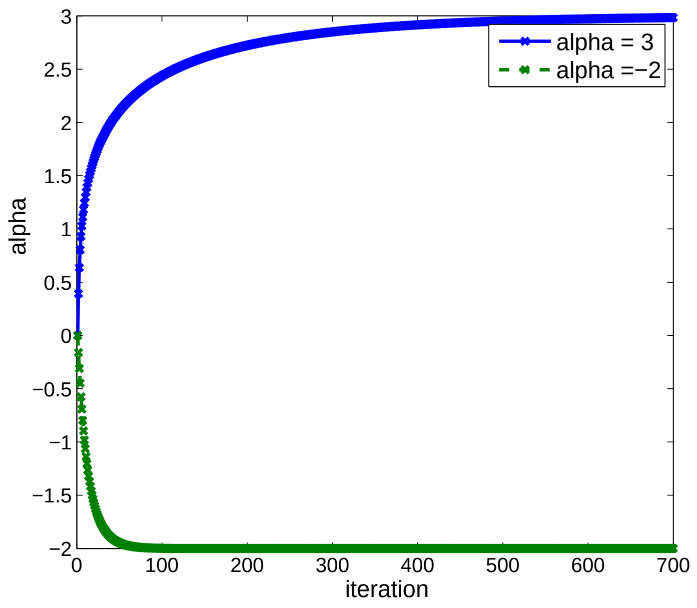


Fig. 28. Trajectories of α values in learning. The true α values were 3 and -2, respectively. Both curves starts from the initial value 0 and are converging to 3 and -2 at around 700 iterations, respectively. The speed of convergence depends on the learning rate, which was 0.05 in this case.

Table I. Squared errors in learning $\boldsymbol{w}$ ($\times 10^{-29}$).

α	-2	-1	0	1	2	3
$\mathcal{J}$	0.725	0.875	3.116	0.017	1.505	0.932
$\boldsymbol{w}$	0.019	0.024	0.135	0.001	0.228	0.217

task, I used a monthly average temperature data from several cities in the U.S as shown in Table II. First, I used three cities (New York, Chicago, and Houston) as sources and estimated the temperature of Atlanta. Second, two cities (San Antonio and Boston) were used as sources and New York as the target. I assumed that all temperature information is correct, so I used the same weights for all source cities. For both cases, I used temperatures from 10 randomly selected months to learn α and tested with the other 2 months' temperature.

Table II. Monthly average temperature data (F^o). Adapted from www.cityrating.com.

City	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Chicago	21.0	25.4	37.2	48.6	58.9	68.6	73.2	71.7	64.4	52.8	40.0	26.6
Boston	28.6	30.3	38.6	48.1	58.2	67.7	73.5	71.9	64.8	54.8	45.3	33.6
NY	31.5	33.6	42.4	52.5	62.7	71.6	76.8	75.5	68.2	57.5	47.6	36.6
Atlanta	41.0	44.8	53.5	61.5	69.2	76.0	78.8	78.1	72.7	62.3	53.1	44.5
Houston	50.4	53.9	60.6	68.3	74.5	80.4	82.6	82.3	78.2	69.6	61.0	53.5
SA	49.3	53.5	61.7	69.3	75.5	82.2	85.0	84.9	79.3	70.2	60.4	52.2

For the first case, Atlanta is roughly equidistant to three cities. Here we have two cooler cities (New York and Chicago) and one warmer city (Houston) than Atlanta. So we can expect α to be low to move the estimation to get closer to the higher value (Houston's temperature) because we have only one warmer city. Fig. 29 (a) shows boxplot of α values for 50 experiments and the average α value is -11.62 (variance 0.54).

In the second case, New York is much closer to Boston than San Antonio. So we can expect the α value to be high. Fig. 29 (b) shows boxplot of α values for 50

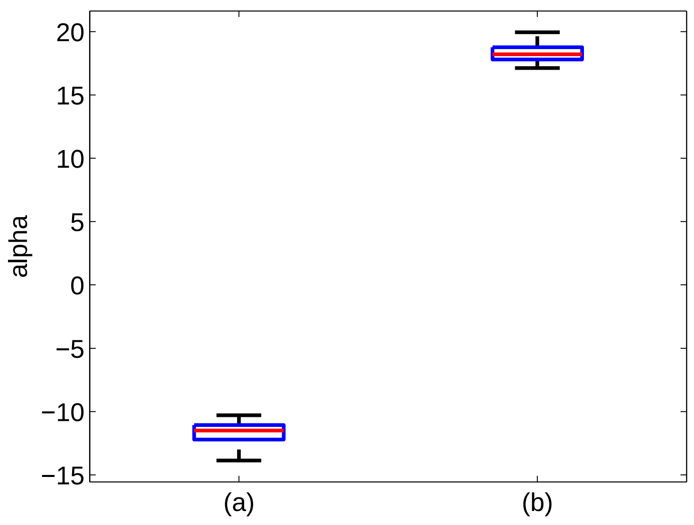


Fig. 29. Boxplot of α values for city temperatures from 50 random experiments. (a) α values for estimating Atlanta's temperature from the three other cities: Chicago, New York and Houston (b) α values for estimating New York's temperature from the two other cities: Boston and San Antonio.

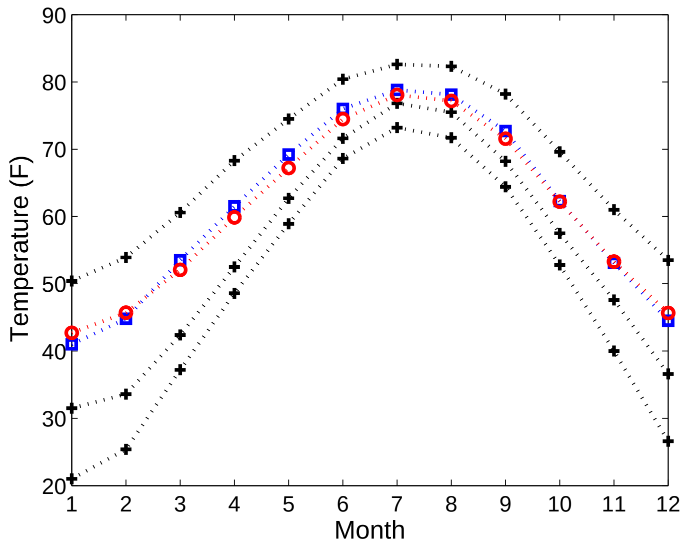


Fig. 30. Average temperatures (F^o) for 1 year. The black (cross) lines are New York, Chicago and Houston. The blue (square) line is the true Atlanta temperature. The red (circle) line is the estimated temperature.

experiments and the average is 18.37 (variance 0.67). With this kind of an example, the α value takes on a concrete meaning, based on temperature-based geometry. According to this α value, we can guess how close the city is to some other city in terms of the temperature. A high α value means that the city is close to the cooler city, and a low α value means the city is close to the warmer city.

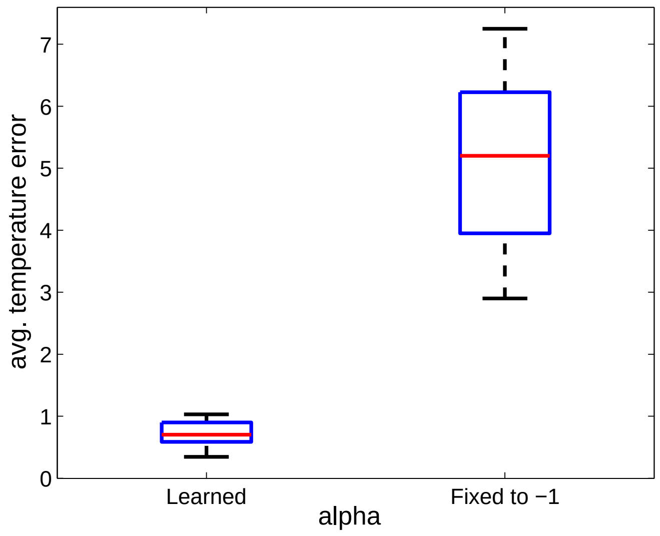


Fig. 31. Errors in New York's temperature (F^o) estimation from the two other cities: Boston and San Antonio. The boxplots are from 50 random experiments. In the 'alpha' axis, 'Learned' means the estimated α and 'Fixed to -1' means the simple arithmetic average.

Fig. 30 shows the true temperature of Atlanta and estimated temperature with $\alpha = -11.62$ learned above. Fig. 31 shows a boxplot of errors in the estimated temperature of New York with $\alpha = 18.37$ learned above or $\alpha = -1$ fixed in advance. There are many causes to affect the temperature of a city. Each month of each city might have different causes to the temperature. That can be the reason of the error we cannot overcome simply by averaging even with a very carefully learned α value. However, the proposed method is better than the simple arithmetic (or geometric or harmonic) average. It theoretically achieves the minimum error among all linear scale-free averaging methods.

d. Discussion

The α-family includes some stochastic models such as the exponential family and the mixture family. So, learning α can be seen as finding the best family out of all possible stochastic families in the α-family. In that sense, the proposed algorithm tries to find the best stochastic family model and the best distribution in the model, iteratively. That is, when we learn α, it finds a better model (a set of distributions) than the current model for the current integration and sources. Then α-integration gives us the best distribution out of the current model. As a consequence, α-integration with learning the value of α finds out the best average out of all distributions in α-family.

From a geometrical view point, if we define the distance between any two points in the set, it implies one corresponding metric, which defines the manifold where the points lie on. Here, learning α means defining the manifold of the probability distributions (or nonnegative measurements). When we initialize α, we assume one manifold and when α changes, the shape of the manifold we assume changes. So, α-integration with learning the value of α gives us the best integration with the metric of the manifold which is defined by the optimized α value. The α-integration and the manifold shape are determined iteratively. These two interpretations are very similar because α-integration originated from information geometry [38].

In addition, as we saw in the previous section, α can take on a concrete meaning depending on the data set. In the temperature experiments, α has some temperature-based geometrical meaning. Likewise, with a specific data set, we can try to interpret the optimized α value after learning.

Here, the parameters α and $\boldsymbol{w}$ are learned separately in α-integration. So, we need further works to learn α and $\boldsymbol{w}$ at the same time.

C. Summary

In this Chapter, I summarized what I have done so far as the theoretical bases for manifold integration. I have studied manifold learning and data integration and developed algorithms for both. Kernel Isomap I proposed is an effective manifold learning algorithm with desirable properties and has been proved effective with many real-world data sets. As for data integration, I proposed some algorithms equipped with α-integration and expanded α-integration. In the next Chapter, I will show how to consider together manifold learning and data integration, leading to manifold integration.

CHAPTER IV

MANIFOLD INTEGRATION

In this dissertation, I develop a new concept, *manifold integration*, a data integration method using the structure of data expressed by multiple manifolds. In this chapter, I describe what manifold integration is (section A) and review some algorithms that are related to manifold integration (section B). Also, in order to achieve manifold integration, I propose three manifold integration algorithms: (1) *kernel oriented discriminant analysis* (KODA) (section C), (2) *random walk on multiple manifolds* (RAMS) (section D), (3) *manifold α-integration* (MAI) (section E). All the proposed algorithm are tested with synthetic and real world data sets.

A. What is Manifold Integration?

Manifold integration (MI) is a concept combining manifold learning and data integration as shown in Fig. 32. Contrary to conventional manifold learning algorithms, MI considers the relationship between data sets (or different manifolds). Also, contrary to typical data integration algorithms, MI considers structural information in each data set (or on each manifold).

One assumption in manifold learning is that all the data points are connected. This assumption makes conventional manifold learning unable to work with multiple manifolds. I start from another assumption that two types of dissimilarities from

©2008 IEEE. Partially reprinted, with permission, from IEEE Int. Conf. on Pattern Recognition "Kernel Oriented Discriminant Analysis for Speaker-Independent Phoneme Spaces" H. Choi, R. Gutierrez-Osuna, S. Choi and Y. Choe. For more information go to http://thesis.tamu.edu/forms/IEEE permission note.pdf.
*Part of this chapter is reprinted with permission from "Manifold Integration with Markov Random Walks" by H. Choi, S. Choi and Y. Choe, 2008, Association for the Advanced of Artificial Intelligence (AAAI-08), Copyright 2008 by AAAI.

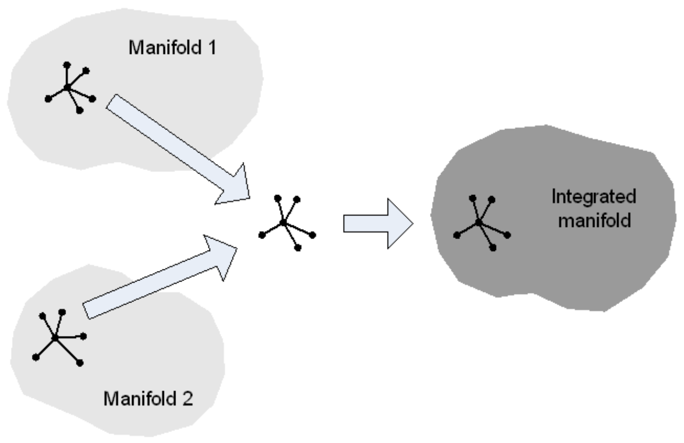


Fig. 32. Overview of manifold integration. Data integration uses statistical relation between two sets and manifold learning uses geometric relation between data points in each set. MI uses both relations.

two different measurements are measured independently. From this, I calculate an integrated transition probability from multiple transition probabilities converted from multiple dissimilarities, and then obtain the statistical distance between data points from the integrated transition probability. This statistical distance is a nonlinear sum of the multiple distances. This approach can be applied to the case of more than two manifolds.

However, the assumption may not generally hold and it makes the integration method to be the product of multiple transition probabilities, which is a special way to integrate data sets. Motivated by advances in manifold learning and data integration, I generalize the approach that makes use of α-integration on the transition probabilities or distances between data points. For the manifold learning part, I use the kernel Isomap algorithm to utilize its projection property. I show that my method includes as its special case previous methods such as statistical distance, kernel-based data fusion or mixture of random walks, by analyzing the compromised distances on the integrated manifold. MI can be considered as generalized kernel integration and can be applied to sensorimotor integration as well as multimodal sensory integration. The proposed MI approach is summarized in Table III.

Table III. Sketch of the MI approach.

1. Obtain dissimilarity matrices, $\boldsymbol{E}_{\mathrm{D}}^{(i)}$, from the ith measurement.
2. Calculate geodesic distance matrices, $\boldsymbol{G}_{\mathrm{D}}^{(i)}$, from $\boldsymbol{E}_{\mathrm{D}}^{(i)}$.
3. Integrate $\boldsymbol{G}_{\mathrm{D}}^{(i)}$ into $\boldsymbol{G}_{\mathrm{D}}^{*}$ with data integration.
4. Find the low-dimensional space from $\boldsymbol{G}_{\mathrm{D}}^{*}$ with manifold learning.
5. Project the measurements onto the low-dimensional space.

However, the current MI approach only works with specific types of data that satisfy the constraints summarized in Table IV. Otherwise, MI will not work as expected. The first constraint is the same constraint as for manifold learning. The second one means we should know which point in the ith measurement corresponds to which point on the jth manifold for all i and j. Tensor network theory also needs this constraint [68]. Finally, MI will not work well if some manifolds are too different topologically from the others. That is, neighbors of the kth point on the ith manifold should be also neighbors of the corresponding point on the jth manifold for all i and j. Note that these constraints are for the current MI approach and we can make these constraints loose in the future.

To make the constraints easier to understand, I will give some examples. As a wrong example for MI, we can think about face images and speech signals for personal identification, though they can be combined in a different way for personal identification. They can be dense enough and the personal identity can connect the facial image to the corresponding speech signal. But, the two manifolds based on the two features probably have different topologies which means there is no homeomorphism

Table IV. Constraints of MI.

1. Data points should be dense enough and connected on each manifold.
2. Each point on one manifold should correspond to one counter part in the other manifolds.
3. All the manifolds should share the same topology (topologically isomorphic).

between the two manifolds. That is, two persons with similar looking faces might have totally different voices.

A good example of MI is two sound signals from the left and the right ears. Another good example is phonemes from multiple speakers, since phonemes that are similar in one speaker would be generally similar in others too. As shown in the next Chapter, sensory information and motor commands are also a good example. Even though the data sets are heterogeneous, sensory and motor maps can be dense enough and can be connected, each point on one map corresponding to one point on the other map and they have topologically similar maps.

B. Related Work

So far, few approaches have been proposed to merge multiple dissimilarity matrices. A simple method to make one dissimilarity matrix is to combine them through substraction or division [69]. Alternatively, we can use the reproducing kernel Krein space (RKKS) instead of reproducing kernel Hilbert space (RKHS) as in [70]. RKKS uses negative eigenvalues in the kernel matrix, while conventional kernel methods use positive semidefinite kernel matrix. Recently, Abdi suggested an algorithm, DISTATIS, based on linear sum of manifolds through principal component analysis (PCA) [71]. Also, oriented PCA (OPCA) [72] was proposed to integrate disconnected subspace.

More importantly, kernel integration (or kernel fusion) is more closely related to MI [21].

In this section, I briefly describe those algorithms to compare them to my algorithms presented in the next sections.

1. Using Reproducing Kernel Krein Space

Most kernel methods use positive semidefinite kernel matrix to guarantee that the dissimilarity matrix is the Euclidean representation in the embedded manifold. In these methods, any negative eigenvalue for the kernel matrix means error or noise. However, as in Isomap, usually the kernel matrix from dissimilarity of points does not satisfy the semi-positiveness. Moreover, as stated in [69], negative eigenvalues might indicate important features.

Ong showed that non-positive kernels are meaningful just as positive kernels and suggested using RKKS to generalize reproducing kernel Hilbert space (RKHS) for both positive semidefinite kernel and negative kernel [70]. With two RKHSs, $\mathcal{H}_+$ and $\mathcal{H}_-$, if a Krein space $\mathcal{K}$ is on RKKS with $\mathcal{K} = \mathcal{H}_+ \ominus \mathcal{H}_-$ defined by $\langle f, g\rangle_{\mathcal{K}} = \langle f_+, g_+\rangle_{\mathcal{H}_+} - \langle f_-, g_-\rangle_{\mathcal{H}_-}$, where $\langle , \rangle_{\mathcal{K}}$, $\langle , \rangle_{\mathcal{H}_+}$ and $\langle , \rangle_{\mathcal{H}_-}$ represent inner products in $\mathcal{K}$, $\mathcal{H}_+$ and $\mathcal{H}_-$, respectively, and $f \in \mathcal{K}$, $f_+ \in \mathcal{H}_+$, $f_- \in \mathcal{H}_-$ as do g, g_+ and g_-, then there exists two positive kernels k_+ and k_- such that

$$k = k_+ - k_-. \tag{4.1}$$

From this, if we have two manifolds that are orthogonal, then we can merge them using Eq. (4.1). From two dissimilarity matrices, we can calculate two kernel matrices and then apply Eq. (4.1). However, this assumption about orthogonality between two manifolds is not satisfied in the general case, so when we apply this approach, there is some distortion in the compromised manifold. Moreover, this approach is only for two dissimilarity measures. To apply this method to more than two dissimilarity measures, we must apply it iteratively. However, it is unnatural and the order of merger should be determined properly.

2. DISTATIS

Abdi calculated kernel matrices $\boldsymbol{S}^{(k)}, k = 1, \cdots, C$ for each dissimilarity matrix $\boldsymbol{D}^{(k)}$, as in kernel Isomap [71].

$$\begin{aligned} \widetilde{\boldsymbol{S}}^{(k)} &= -\frac{1}{2}\boldsymbol{H}\boldsymbol{D}^{(k)2}\boldsymbol{H}, && (4.2) \\ \boldsymbol{S}^{(k)} &= \lambda_1^{-1}\widetilde{\boldsymbol{S}}^{(k)}, && (4.3) \end{aligned}$$

where λ_1 is the first eigenvalue of $\widetilde{\boldsymbol{S}}^{(k)}$, $\boldsymbol{D}^{(k)2} = [D_{ij}^{(k)2}]$ means the element-wise square of the distance matrix $\boldsymbol{D}^{(k)} = [D_{ij}^{(k)}]$, and $\boldsymbol{H}$ is the centering matrix, given by Eq. (2.2). Then, each kernel matrix is converted into one vector $\boldsymbol{s}_i$, to produce a matrix $\boldsymbol{X} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_T]$, which corresponds to a matrix of manifolds using my terminology. The principal components are then calculated using the inner product of $\boldsymbol{X}$ as in MDS or Isomap. The first eigenvector corresponding to the largest eigenvalue is the compromised matrix, which serves as the target manifold to project the data set. Actually, this final compromised manifold is expressed using the kernel matrix, $\boldsymbol{S}_+$.

$$\boldsymbol{S}_+ = \sum_{k=1}^{C} \alpha^{(k)}\boldsymbol{S}^{(k)}, \tag{4.4}$$

where α is the first eigenvector of $\boldsymbol{N}^{-\frac{1}{2}}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{N}^{-\frac{1}{2}}$, and $\boldsymbol{N}$ is the diagonal matrix with diagonal terms of $\boldsymbol{X}^\top\boldsymbol{X}$.

The remaining part is the same as MDS or Isomap. Projection to this compromised manifold $\boldsymbol{Y}$ is executed by Eq. (4.5) after eigen-decomposition, $\boldsymbol{S}_+ = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$,

$$\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}}, \tag{4.5}$$

where columns of $\boldsymbol{V}$ and diagonal elements of $\boldsymbol{\Lambda}$ are eigenvectors and eigenvalues of $\boldsymbol{S}_+$, respectively.

Here, Abdi et al. tried to find the best space to project the data set. However, because they used just the first linear principal component of $\boldsymbol{X}$, they lose some information residing in the other components. Especially, when the principal component is nonlinear, it will lose more information.

3. Oriented PCA

OPCA [72] was proposed by Malayath [73] as a potential method to find such speaker-independent phoneme space. OPCA is an extension of principal component analysis (PCA). Like PCA, OPCA maximizes variance in directions defined as informative, but in addition also minimizes variance in directions considered to be noisy. In the original formulation in [73], OPCA was used to separate two phonemes and two speakers as shown in Fig. 33.

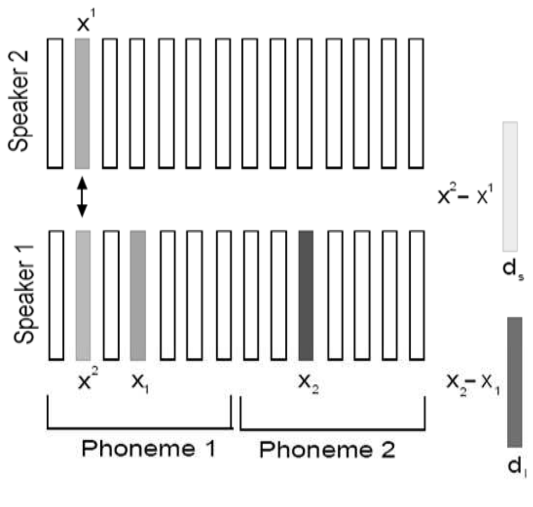


Fig. 33. Speech utterances contain two kinds of information. OPCA maximizes the linguistic information and minimizes the identity information for phoneme recognition. Adapted from [73].

Malayath defines a difference vector d_l to capture differences between two phonemes for the same speaker, and a difference vector d_s to capture differences between two speakers for the same phoneme. From these difference vectors, we can then estimate a covariance matrix for each of the two sources of information in the data (i.e., speaker-specific and linguistic) as:

$$\begin{aligned} \boldsymbol{R}_\mathrm{l} &= E\left[(\boldsymbol{d}_\mathrm{l} - \overline{\boldsymbol{d}}_\mathrm{l})(\boldsymbol{d}_\mathrm{l} - \overline{\boldsymbol{d}}_\mathrm{l})^\top\right], \\ \boldsymbol{R}_\mathrm{s} &= E\left[(\boldsymbol{d}_\mathrm{s} - \overline{\boldsymbol{d}}_\mathrm{s})(\boldsymbol{d}_\mathrm{s} - \overline{\boldsymbol{d}}_\mathrm{s})^\top\right], \end{aligned} \tag{4.6}$$

where $\overline{\boldsymbol{d}}_\mathrm{l}$ and $\overline{\boldsymbol{d}}_\mathrm{s}$ are the mean difference between phonemes and speakers, respectively, and $E\left[\cdot\right]$ is the expectation operator. Then, the objective function $\boldsymbol{J}_\mathrm{OPCA}(\boldsymbol{w})$ to be maximized can be written as follows:

$$\boldsymbol{J}_\mathrm{OPCA}(\boldsymbol{w}) = \frac{\text{Signal}}{\text{Noise}} = \frac{\boldsymbol{w}^\top \boldsymbol{R}_\mathrm{l} \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{R}_\mathrm{s} \boldsymbol{w}}, \tag{4.7}$$

where $\boldsymbol{w}$ are the basis vectors of the projected space. Note that, by maximizing $\boldsymbol{J}_\mathrm{OPCA}$, we also maximize the signal-to-noise ratio, i.e., the variance due to phonetic content relative to the variance due to speaker information. This equation is similar to the objective function in LDA [27], except that $\boldsymbol{R}_\mathrm{l}$ and $\boldsymbol{R}_\mathrm{s}$ are covariance matrices of phoneme-difference and speaker-difference, instead of between-class scatter and within-class scatter.

Note that OPCA assumes that all the phonemes from all the speakers are on one linear space, which is not always the case.

4. Kernel Fusion

Kernel fusion is a process to integrate separately tuned kernels to analyze multiple measurements including heterogeneous data sources in computer vision, bioinformatics, audio processing problems and so on. Lanckriet [21] used a weighted sum of kernel matrices for kernel-based data fusion. The integrated kernel from C kernel matrices is calculated as follows.

$$\boldsymbol{K}_{\text{fusion}} = \sum_{k=1}^{C} w_k \boldsymbol{K}^{(k)}, \tag{4.8}$$

where $\boldsymbol{K}^{(k)}$ is the kernel matrix from the kth measurement. Usually the kernel fusion methods focus on how to calculate the weight w_k, assuming that the fused kernel matrix can be obtained by weighted sum of kernel matrices, even though there are other ways to integrate the kernels like the product of kernels.

C. Kernel Oriented Discriminant Analysis

Here, I interpret OPCA as a linear multiple MI method considering one speaker's phoneme space as one manifold. OPCA is an extension of PCA. Like PCA, OPCA maximizes variance in directions defined informative, but in addition also minimizes variance in directions considered to be noisy. In the original formulation [73], OPCA was used to separate two phonemes of two speakers.

In this dissertation, I extend OPCA to the non-linear case by means of the kernel trick used in KPCA and KFD [13, 27]. This method, referred to as *kernel OPCA* (KOPCA), employs a geodesic-distance-based kernel similar to my previous work [10], but the technique can be easily extended to other kernel functions (e.g. polynomial, exponential, hyperbolic tangent functions), as long as they satisfy the Mercer kernel condition, i.e., positive semidefiniteness of the kernel matrix. As a second step, I propose a generalization of this algorithm, KOPCA, to multi-class discrimination problems, and will demonstrate its effectiveness on multiple phonemes. For this purpose, I use the classical linear discriminant analysis (LDA) approach to OPCA which leads to oriented discriminant analysis (ODA), and will apply the kernel trick to ODA to obtain a *kernel ODA* (KODA) method [74].

Although KODA does not perfectly fit into the MI concept described in the previous section, it can be considered as a special version of manifold integration since it integrates manifolds. We discuss this issue later with Fig. 36.

1. KODA Algorithm

Kernel ODA can be considered as a generalized version of kernel OPCA for multi-class problems. Here, to avoid clutter, I show the derivation of kernel OPCA in detail. Since the derivation of kernel ODA is very similar to that of kernel OPCA, I give just the objective function of kernel ODA.

In order to "kernelize" OPCA, the objective function in Eq. (4.7) is implemented as an inner product matrix. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{4N}]$ and $\boldsymbol{x}_i \in \mathcal{R}^{d\times 1}$, where

$$\boldsymbol{x}_i = \begin{cases} \text{Speaker A, Phoneme X} & \text{if } 1 \le i \le N \\ \text{Speaker B, Phoneme X} & \text{if } N+1 \le i \le 2N \\ \text{Speaker A, Phoneme Y} & \text{if } 2N+1 \le i \le 3N \\ \text{Speaker B, Phoneme Y} & \text{if } 3N+1 \le i \le 4N \end{cases}.$$

As in KFD [27], we transform $\boldsymbol{w}^\top \boldsymbol{R}_{\mathrm{l}} \boldsymbol{w}$ and $\boldsymbol{w}^\top \boldsymbol{R}_{\mathrm{s}} \boldsymbol{w}$ into $\boldsymbol{\alpha}^\top \boldsymbol{M} \boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^\top \boldsymbol{N} \boldsymbol{\alpha}$, respectively using the kernel trick as follows.

$$\boldsymbol{\alpha}^\top \boldsymbol{M} \boldsymbol{\alpha} = \boldsymbol{w}^\top \boldsymbol{R}_{\mathrm{l}}^{\Phi} \boldsymbol{w}, \tag{4.9}$$

$$\boldsymbol{\alpha}^\top \boldsymbol{N} \boldsymbol{\alpha} = \boldsymbol{w}^\top \boldsymbol{R}_{\mathrm{s}}^{\Phi} \boldsymbol{w}, \tag{4.10}$$

where $\boldsymbol{R}_{\mathrm{l}}^{\Phi}$ and $\boldsymbol{R}_{\mathrm{s}}^{\Phi}$ correspond to $\boldsymbol{R}_{\mathrm{l}}$ and $\boldsymbol{R}_{\mathrm{s}}$ in nonlinear feature space obtained by applying the nonlinear mapping Φ to the original data points $\boldsymbol{X}$. Once these $\boldsymbol{M}$ and $\boldsymbol{N}$ matrices are defined, the remaining part of the method is the same as in KFD

[27]. I describe the details of the case $\boldsymbol{R}_{\text{l}}$ below ($\boldsymbol{R}_{\text{s}}$ follows a similar derivation).

First, I express the objective function in terms of the input data $\boldsymbol{X}$ instead of the difference vectors $\boldsymbol{d}_{\text{l}}$ and $\boldsymbol{d}_{\text{s}}$ using $\boldsymbol{d}_{\text{l}} = [\boldsymbol{x}_{2N+1} - \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{4N} - \boldsymbol{x}_{2N}]$, which represents the difference between phonemes. Then the covariance matrix[1] is given by

$$\begin{aligned}\boldsymbol{R}_{\text{l}} &= E\left[(\boldsymbol{d}_{\text{l}} - \overline{\boldsymbol{d}_{\text{l}}})(\boldsymbol{d}_{\text{l}} - \overline{\boldsymbol{d}_{\text{l}}})^\top\right] \\ &= \frac{1}{2N}\sum_i^{2N}\left((\boldsymbol{d}_{\text{l}}^i - \overline{\boldsymbol{d}_{\text{l}}^i})(\boldsymbol{d}_{\text{l}}^i - \overline{\boldsymbol{d}_{\text{l}}^i})^\top\right) \\ &= \frac{1}{2N}\sum_i^{2N}(\boldsymbol{x}_{2N+i} - \boldsymbol{x}_i - \overline{\boldsymbol{x}_{2N+i} - \boldsymbol{x}_i}) \cdot (\boldsymbol{x}_{2N+i} - \boldsymbol{x}_i - \overline{\boldsymbol{x}_{2N+i} - \boldsymbol{x}_i})^\top, \quad (4.11)\end{aligned}$$

and

$$\boldsymbol{w} = \sum_i^{4N}\alpha_i\boldsymbol{x}_i. \quad (4.12)$$

Now, $\boldsymbol{w}^\top\boldsymbol{R}_{\text{l}}\boldsymbol{w}$ is given by

$$\begin{aligned}\boldsymbol{w}^\top\boldsymbol{R}_{\text{l}}\boldsymbol{w} &= \sum_i^{4N}\alpha_i\boldsymbol{x}_i^\top\frac{1}{2N}\sum_k^{2N}\left((\boldsymbol{d}_{\text{l}}^k - \overline{\boldsymbol{d}_{\text{l}}^k})(\boldsymbol{d}_{\text{l}}^k - \overline{\boldsymbol{d}_{\text{l}}^k})^\top\right)\sum_j^{4N}\alpha_j\boldsymbol{x}_j \\ &= \frac{1}{2N}\sum_k^{2N}\left(\sum_i^{4N}\alpha_i\boldsymbol{x}_i^\top(\boldsymbol{d}_{\text{l}}^k - \overline{\boldsymbol{d}_{\text{l}}^k})(\boldsymbol{d}_{\text{l}}^k - \overline{\boldsymbol{d}_{\text{l}}^k})^\top\sum_i^{4N}\alpha_i\boldsymbol{x}_i\right). \quad (4.13)\end{aligned}$$

Let $\boldsymbol{H}^{ik} = \boldsymbol{x}_i^\top(\boldsymbol{d}_{\text{l}}^k - \overline{\boldsymbol{d}_{\text{l}}^k})$. Now, with $\boldsymbol{K}_{i,j} = \boldsymbol{x}_i{}^\top\boldsymbol{x}_j$,

$$\boldsymbol{H}^{ik} = \boldsymbol{K}_{i,2N+k} - \boldsymbol{K}_{i,k} - \frac{1}{2N}\sum_m^{2N}\left(\boldsymbol{K}_{i,2N+m} - \boldsymbol{K}_{i,m}\right). \quad (4.14)$$

[1]We can also define the correlation. Then, consequently, $\boldsymbol{M}$ and $\boldsymbol{N}$ change slightly with respect to Eqs. (4.16) and (4.17), respectively. In my experiments, the use of this correlation showed a slightly better performance than the covariance-based algorithm.

Finally,

$$\begin{aligned}
\boldsymbol{w}^\top \boldsymbol{R}_l \boldsymbol{w} &= \frac{1}{2N}\sum_{k}^{2N}\left(\sum_{i,j}^{4N}\alpha_i\alpha_j \boldsymbol{H}^{ik}\boldsymbol{H}^{jk}\right) \\
&= \frac{1}{2N}\sum_{i,j}^{4N}\alpha_i\alpha_j\sum_{k}^{2N}\boldsymbol{H}^{ik}\boldsymbol{H}^{jk} \\
&= \frac{1}{2N}\sum_{i,j}^{4N}\alpha_i\alpha_j(\boldsymbol{H}\boldsymbol{H}^\top)_{ij} \\
&= \frac{1}{2N}\boldsymbol{\alpha}^\top\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{\alpha}. \qquad (4.15)
\end{aligned}$$

Therefore, we obtain $\boldsymbol{M}$ by

$$\boldsymbol{M} = \frac{1}{2N}\boldsymbol{H}\boldsymbol{H}^\top. \qquad (4.16)$$

Likewise, in case of the denominator $\boldsymbol{w}^\top\boldsymbol{R}_s\boldsymbol{w}$ in Eq. (4.7), with $\boldsymbol{d}_s = [\boldsymbol{x}_{N+1} - \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{2N} - \boldsymbol{x}_N, \boldsymbol{x}_{3N+1} - \boldsymbol{x}_{2N+1}, \cdots, \boldsymbol{x}_{4N} - \boldsymbol{x}_{3N}]$, we can derive $\boldsymbol{N}$ as

$$\boldsymbol{N} = \frac{1}{2N}\boldsymbol{G}\boldsymbol{G}^\top + \mu\boldsymbol{I}, \qquad (4.17)$$

where

$$\begin{aligned}
\boldsymbol{G} &= [\boldsymbol{G}_1\boldsymbol{G}_2], \qquad (4.18) \\
\boldsymbol{G}_{1,ik} &= \boldsymbol{K}_{i,N+k} - \boldsymbol{K}_{i,k} \\
&\quad -\frac{1}{2N}\sum_{m=1}^{N}\left(\boldsymbol{K}_{i,N+m} + \boldsymbol{K}_{i,3N+m} - \boldsymbol{K}_{i,m} - \boldsymbol{K}_{i,2N+m}\right), \qquad (4.19) \\
\boldsymbol{G}_{2,ik} &= \boldsymbol{K}_{i,3N+k} - \boldsymbol{K}_{i,2N+k} \\
&\quad -\frac{1}{2N}\sum_{m=1}^{N}\left(\boldsymbol{K}_{i,N+m} + \boldsymbol{K}_{i,3N+m} - \boldsymbol{K}_{i,m} - \boldsymbol{K}_{i,2N+m}\right). \qquad (4.20)
\end{aligned}$$

Note that, for regularization purposes, I add a multiple of the identity matrix, $\mu\boldsymbol{I}$ to $\boldsymbol{N}$ in Eq. (4.17), where μ is a small number to make $\boldsymbol{N}$ positive definite [27]. In my experience, this regularization term makes the algorithm more stable.

Given a novel test data $\boldsymbol{x}_t$, the projected point $\boldsymbol{y}_t$ can be calculated as

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{w}^\top \boldsymbol{x}_t & (4.21) \\ &= \sum_i^{4N} \alpha_i \boldsymbol{K}_{\mathrm{T},it}, & (4.22) \end{aligned}$$

where $\boldsymbol{K}_\mathrm{T}$ is a kernel matrix for test data as in Eq. (3.12).

Several Mercer kernel functions (i.e. polynomial, exponential, or hyperbolic tangent function) can be used for the kernel matrices $\boldsymbol{K}$ and $\boldsymbol{K}_\mathrm{T}$. Here, I use a kernel matrix based on the geodesic distance. As discussed in previous work [10], this approach has the advantage that it can find a nonlinear structure of data set without critical parameters affecting the performance.

In kernel OPCA, I considered an example with two phonemes and two speakers. I extend the solution to problems with more than two phonemes. This extension leads to a new method, which I term kernel ODA. I first define ODA in terms of the covariance matrix, then derive the kernel ODA solution. With only two speakers, the correlation matrix $\boldsymbol{R}_\mathrm{s}$ remains the same as before. Only the correlation matrix $\boldsymbol{R}_\mathrm{l}$ must be adjusted to handle more than two phonemes. My solution is to use the between-scatter matrix $\boldsymbol{R}_\mathrm{L}$ in the LDA solution. With this minor adjustment, the final objective function of ODA becomes

$$\boldsymbol{J}_{\mathrm{ODA}}(\boldsymbol{w}) = \frac{\boldsymbol{w}^\top \boldsymbol{R}_\mathrm{L} \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{R}_\mathrm{s} \boldsymbol{w}}, \tag{4.23}$$

where $\boldsymbol{R}_\mathrm{s}$ is given in Eq. (4.6) and $\boldsymbol{R}_\mathrm{L}$ is given by

$$\boldsymbol{R}_\mathrm{L} = \sum_i^C n_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^\top, \tag{4.24}$$

where C is the number of classes (phonemes), n_i is the number of phonemes in the ith class, $\boldsymbol{\mu}_i$ is the average of the ith class, and $\boldsymbol{\mu}$ is the average of all data. To kernelize it as in Eqs. (4.9) and (4.10), $\boldsymbol{M}$ is obtained from the KFD solution [27] instead of Eq. (4.16), and $\boldsymbol{N}$ is obtained from the kernel OPCA solution in Eq. (4.17). Projections of novel test data are obtained with Eq. (4.21). To calculate $\boldsymbol{M}$ and $\boldsymbol{N}$, I again use the kernel matrices $\widetilde{\boldsymbol{K}}$ and $\widetilde{\boldsymbol{K}_{\mathrm{T}}}$ from kernel Isomap [10].

2. Experiments

I validated the proposed methods through a series of experiments using the CMU ARCTIC speech database [75]. As performance measures, we applied quadratic classifiers to the projection of the test data and measured the Bhattacharyya distance of the class-conditional distributions [76].

Two speakers and two phonemes: The CMU ARCTIC database is a phonetically balanced corpus from US speakers, which was designed for unit selection speech synthesis research. The database includes US English male ('bdl', 'rms') and female ('slt', 'clb') speakers. For a representative example, I extracted two phonemes ('AH' and 'IH') for each speaker, and used Mel frequency cepstral coefficients (MFCCs) as the feature vectors. I used two speakers ('bdl', 'slt') for training and two speakers ('rms', 'clb') for testing. I used 300 samples per phoneme of each speaker for training, and 400 samples for testing, for a total of 1,200 training samples and 1,600 test samples. Note that since the phonemes were extracted from real sentences, two samples from the same speaker and the same phoneme class 'AH' may have significantly different MFCCs as a result of coarticulatory effects.

Fig. 34 shows the subspaces for linear OPCA and kernel OPCA. Even though both scatterplots show speaker-independent subspaces, the kernel OPCA solution appears to provide increased separability. Indeed, I measured the Bhattacharyya distance [76] between the two clusters of phonemes and compared the classification rate based on quadratic classifier. The results, summarized in Table V, indicate that

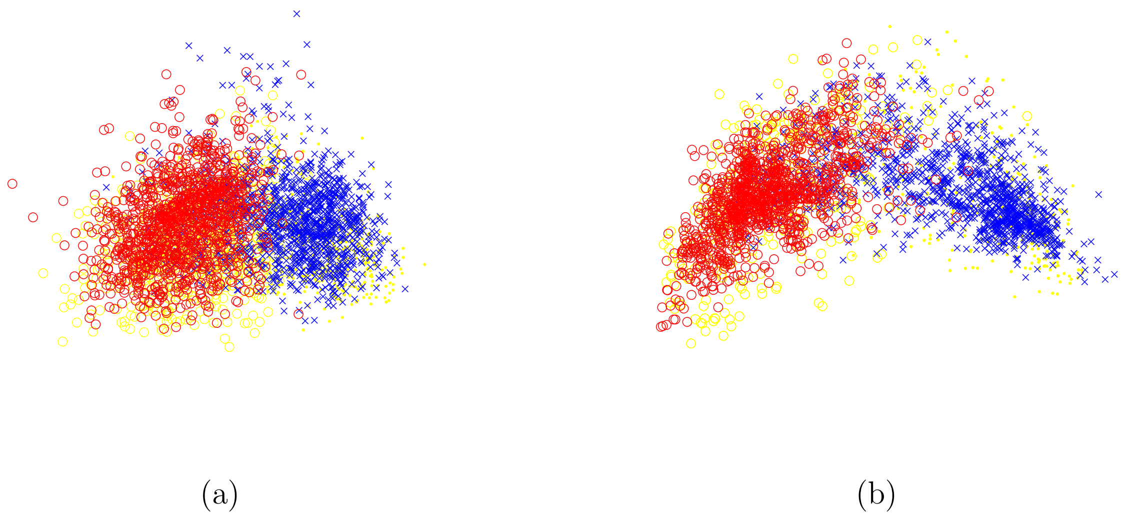


Fig. 34. Subspaces for two speakers and two phonemes. (a) linear OPCA and (b) kernel OPCA. Blue crosses correspond to the phoneme ‘AH,’ whereas red circles correspond to the phoneme ‘IH’. Yellow circles denote the training data.

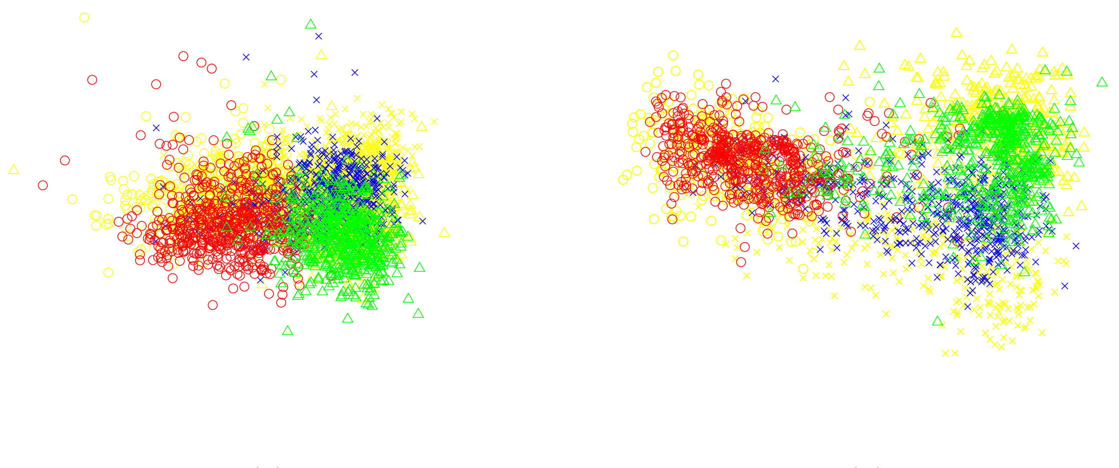
kernel OPCA provides better phoneme discrimination than linear OPCA using either measure. Paired T-test indicates that the difference in classification performance between both methods is statistically significant (p=0.0406; n=24).

Two speakers and multiple phonemes: The CMU ARCTIC database is also used for these experiments. In this case, I extracted three phonemes (‘AH’, ‘IH’ and ‘OW’) for each speaker and used MFCCs as the encoding vector. As in the previous experiment, I used two speakers (‘bdl’, ‘slt’) for training and two speakers (‘rms’, ‘clb’) for testing, resulting in 300 samples per phoneme of each speaker for training, and 300 samples for testing, for a total of 1,800 training samples and 1,800 test samples.

Fig. 35 shows the resulting subspaces for linear ODA and kernel ODA. Kernel ODA provides more scattered clusters than linear ODA (both within and between classes), in agreement with kernel OPCA in the previous experiments. Classification

Table V. Measurement of the performance for two speakers and two phonemes (MFCCs) on OPCA and KOPCA.

Methods	B-dist Train	B-dist Test	Hit Rate on test data
OPCA + Cov	14.71	14.70	85.88%
OPCA + Cor	14.52	14.53	87.58%
KOPCA + Cov	24.11	23.80	88.13%
KOPCA + Cor	24.33	23.91	89.54%



(a) (b)

Fig. 35. Subspaces for two speakers and three phonemes. (a) linear ODA and (b) kernel ODA. The blue crosses are 'AH', the red circles are 'IH', and the green triangles are 'OW'.

rates using a quadratic classifier were 72.93% (linear ODA) and 78.17% (kernel ODA). When KFD is optimized with some kernel functions (here RBF was the best) and parameters, it has 74.86% hit rate with the same classifier as before, which means just phoneme information is not enough to find a speaker-independent space.

3. Discussion

In this section, I proposed a two-pronged generalization of oriented PCA. First, I found a nonlinear subspace by means of the kernel trick, which led to kernel OPCA. Second, I extended kernel OPCA to problems with more than two classes, which led to linear ODA and kernel ODA. Experimental results on the CMU ARCTIC corpus showed that the proposed methods, kernel OPCA and kernel ODA, provide better separability than their linear counterparts (OPCA and linear ODA) in finding a speaker-independent phoneme space, as measured by classification rates and the Bhattacharyya distance.

These algorithms can be viewed as nonlinear manifold-learning strategies for problems where data points exist on several clustered manifolds corresponding to their classes. These algorithms were tested with relatively small data sets in the speech domain. Additional work is required to determine the extent to which these results will hold when applied to a larger data set of speakers and the entire phonetic space.

KODA can integrate multiple manifolds assuming that all the manifolds are submanifolds of the same unknown manifold and that they are connected as in Fig. 36. The method also maximizes the separability between different classes. In practice, however, multiple manifolds might not be connected on one manifold as submanifolds. To overcome such a strong assumption in KODA, as shown in Fig. 32, we need to obtain all the structural information from each separate manifold and then integrate

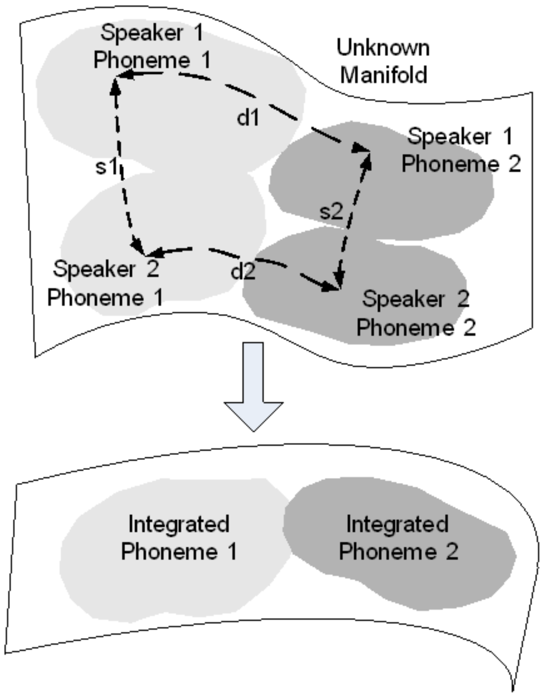


Fig. 36. Overview of KODA in the case of two speakers' two phonemes. KODA minimizes the distance between the same phonemes (s1 and s2), and maximizes the distance between different phonemes (d1 and d2). These distances are obtained based on the assumption that these submanifolds are on one unknown manifold.

all the information while not connecting the manifolds geometrically.

D. Random Walks on Multiple Manifolds

In this section, I suggest a new algorithm, *random walk on multiple manifolds* (RAMS), utilizing the fact that the distance in Euclidean space can be translated into transition probability [77]. Sometimes, it is natural to assume that two dissimilarities are measured independently. With this assumption, we can calculate one transition probability from multiple transition probabilities, and then obtain a statistical distance from the integrated transition probability. This statistical distance is a nonlinear sum of the multiple distances, contrary to DISTATIS [71]. RAMS can be applied to the case of more than two manifolds, contrary to the reproducing kernel Krein space (RKKS) method which uses the sign of the eigenvalues [70]. Moreover, since RAMS uses kernel Isomap after getting one integrated dissimilarity matrix, RAMS inherits the projection property from kernel Isomap [10].

1. RAMS Algorithm

The connection between random walk and manifold has been mentioned in many other papers [78, 15, 79]. Let $\boldsymbol{G}$ be a weighted graph with N nodes $\boldsymbol{V}$ and edges $\boldsymbol{E}$, representing a manifold. Then, the distance between two nodes on the kth manifold, $D_{ij}^{(k)}$, can be transformed into probability $P_{ij}^{(k)}$ which is the transition probability from the ith node to the jth node on the kth manifold, which can be given by

$$P_{ij}^{(k)} = \frac{1}{Z_i^{(k)}} e^{-\frac{D_{ij}^{(k)2}}{\sigma^{(k)2}}}, \tag{4.25}$$

where $Z_i^{(k)}$ is a normalization term so that the sum of transition probabilities from the ith node to all its neighbors on the kth manifold is 1, and $\sigma^{(k)}$ is a parameter representing the standard deviation.

Given C dissimilarity matrices, $\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}, \cdots, \boldsymbol{D}^{(C)}$, we can get C probability matrices, $\boldsymbol{P}^{(1)}, \boldsymbol{P}^{(2)}, \cdots, \boldsymbol{P}^{(C)}$. I assume that these dissimilarities are measured independently. Note that we are not assuming that these dissimilarities make up orthogonal manifolds. Based on these assumptions, the compromised probability matrix $\boldsymbol{P}^*$ is calculated as

$$P^*_{ij} = \frac{1}{Z^*_i} \prod_{k=1}^{C} P^{(k)}_{ij}, \tag{4.26}$$

where Z^*_i is a normalization term given by $Z^*_i = \sum_j P^*_{ij}$. Eq. (4.26) represents the probability for transition from the ith node to the jth node on the target (or integrated) manifold. From $\boldsymbol{P}^*$, I reconstruct the compromised dissimilarity $\boldsymbol{D}^*$ again.

$$D^*_{ij} = \sigma^* \sqrt{-\log(P^*_{ij})}, \tag{4.27}$$

which is the *statistical distance* from all the individual manifolds. Here, the parameters are given by

$$\sigma^{(k)} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} D^{(k)}_{ij}, \tag{4.28}$$

$$\sigma^* = \frac{1}{C} \sum_{k=1}^{C} \sigma^{(k)}. \tag{4.29}$$

Fig. 37 sketches how RAMS handle multiple sources.

After getting $\boldsymbol{D}^*$, the rest is the same as kernel Isomap [10]. I calculate the kernel matrix from the compromised dissimilarity matrix.

$$\boldsymbol{K} = -\frac{1}{2} \boldsymbol{H} \boldsymbol{D}^{*2} \boldsymbol{H}. \tag{4.30}$$

As in kernel Isomap, I make the kernel matrix positive semidefinite by adding a

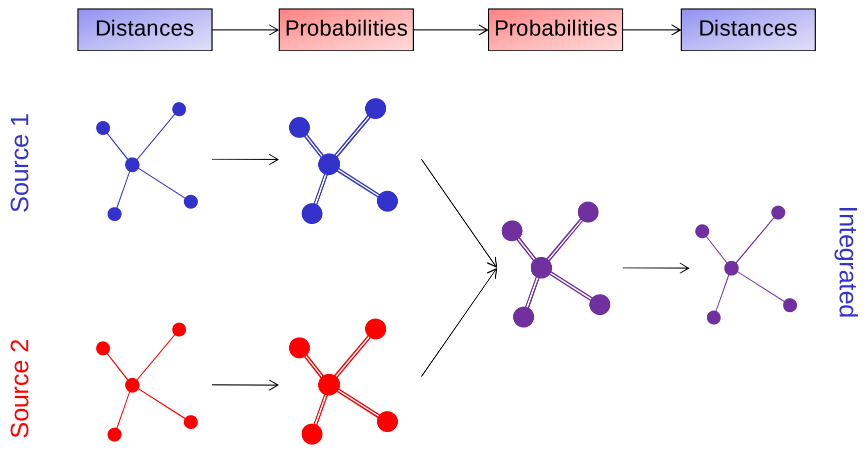


Fig. 37. Conversion between distance and transition probability in RAMS.

constant, c.

$$\widetilde{\boldsymbol{K}} = \boldsymbol{K}(\boldsymbol{D}^{*2}) + 2c\boldsymbol{K}(\boldsymbol{D}^{*}) + \frac{1}{2}c^2\boldsymbol{H}, \tag{4.31}$$

where c is the largest eigenvalue of the matrix

$$\begin{bmatrix} \mathbf{0} & 2\boldsymbol{K}(\boldsymbol{D}^{*2}) \\ -\boldsymbol{I} & -4\boldsymbol{K}(\boldsymbol{D}^{*}) \end{bmatrix}. \tag{4.32}$$

Eq. (4.31) implies substituting $\widetilde{\boldsymbol{D}}^{*}$ for $\boldsymbol{D}^{*}$ in Eq. (4.30), which is given by

$$\widetilde{D}^{*}_{ij} = D^{*}_{ij} + c(1 - \delta_{ij}), \tag{4.33}$$

which makes the matrix $\boldsymbol{K}$ to be positive semidefinite. The term δ_{ij} is the Kronecker delta. Finally, projection mapping $\boldsymbol{Y}$ is given by Eq. (4.34) after eigen-decomposition, $\widetilde{\boldsymbol{K}} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\top}$.

$$\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}}. \tag{4.34}$$

2. Relation to Previous Work

In RKKS, in the case of two distance matrices, the compromised distance matrix $\boldsymbol{D}^*$ is obtained implicitly from Eq. (4.1) and is given by

$$D^*_{ij} = \sqrt{D^{(1)2}_{ij} - D^{(2)2}_{ij}}. \tag{4.35}$$

It is not easy to compare Eq. (4.35) directly to DISTATIS or RAMS because it uses negative eigenvalues even though $\boldsymbol{D}^{*2}$ is a linear sum of $\boldsymbol{D}^{(1)2}$ and $\boldsymbol{D}^{(2)2}$ as in DISTATIS.

From the RKKS theory, however, if $D^{(1)}$ is orthogonal to $D^{(2)}$, which means two distances are uncorrelated, RKKS works, otherwise it has some distortion in the compromised manifold. Generally, two measurements will be correlated and this approach has some distortion on the final manifold. On the other hand, RAMS does not assume that two measurements are orthogonal. One assumption in RAMS is that two distances are measured independently, which is generally true. So, even with correlated distance matrices, RAMS works well.

In DISTATIS, the compromised distance matrix, $\boldsymbol{D}^*$, is obtained implicitly from Eq. (4.4) and given by

$$D^*_{ij} = \sqrt{\sum_{k=1}^{C} \alpha^{(k)} \frac{D^{(k)2}_{ij}}{\lambda^{(k)}_1}}. \tag{4.36}$$

Eq. (4.36) shows that $\boldsymbol{D}^{*2}$ is a linear sum of each squared distance matrix as in the RKKS method, where $\alpha^{(k)}$ gives the weight of the normalized distance matrix. In other words, it represents the importance of the individual manifold.

On the other hand, in RAMS, before adding the constant c in Eq. (4.33), the compromised distance matrix $\boldsymbol{D}^*$ is obtained explicitly as in Eq. (4.27) and is given by

$$D^*_{ij} = \frac{1}{C}\sum_{t=1}^{C}\sigma^{(t)}\sqrt{\log Z^*_i + \sum_{k=1}^{C}\log Z^{(k)}_i + \frac{D^{(k)2}_{ij}}{\sigma^{(k)2}}}. \tag{4.37}$$

In Eq. (4.37), the distance is calculated in a statistical way, where some variables such as $\log Z^*_i + \sum_{k=1}^{C}\log Z^{(k)}_i$ reflect how much related the ith point is to others on the compromised manifold. To compare with Eq. (4.36), if the normalization terms are 1, which means data points are already distributed in a well normalized form, then $\boldsymbol{D}^{*2}$ is a linear sum of each squared distance matrix as in Eq. (4.36).

RAMS is generally better than DISTATIS, especially when the compromised manifold is nonlinear. For example, let A, B, and C be three points with two distance matrices and the distance between A and C is longer than the distance between B and C in both distance matrices. However, if A and B on each individual manifold are located on a perpendicular line from the compromised manifold, then the distance between A and C will be the same as that between B and C on the compromised manifold represented by $\boldsymbol{D}^*$. But in RAMS, if one distance is longer than the other in both distance matrices, the former will be definitely longer than the latter on the compromised manifold. The reason for this difference is that DISTATIS uses a linearly compromised manifold that discards the other components except for the first principal component, whereas RAMS uses statistical distance as in Eq. (4.27).

Usually, the manifold made by a distance matrix is curved in high dimensional space and then the first components of the manifolds is not enough to contain the proper information. This is the reason why DISTATIS has some problems with nonlinear manifolds. However, RAMS does not depend on the linear structure of distance matrices but depends on the statistical structure. This makes RAMS robust even with nonlinear manifolds.

3. Projection Property

RAMS has the projection property which involves the embedding of test data points in the associated low-dimensional space. In other words, generalization property means the ability to determine a point $\boldsymbol{y}_l$ embedded in the low-dimensional space, given a test data point $\boldsymbol{x}_l$. The generalization property naturally emerges from the fact that $\widetilde{\boldsymbol{K}}$ (geodesic kernel with the constant-shifting employed in kernel Isomap) is a Mercer kernel.

In this section, I describe how to get the distance from test point $\boldsymbol{x}_l$ to other training points $\boldsymbol{x}_j$ on the compromised manifold, and how to map the test point on the compromised manifold. Let the distance between the test point $\boldsymbol{x}_l$ and a training point $\boldsymbol{x}_j$ on the kth manifold be $D^{(k)}_{\mathrm{T}:lj}$, then it can be transformed into probability $P^{(k)}_{\mathrm{T}:lj}$ which is given by

$$P^{(k)}_{\mathrm{T}:lj} = \frac{1}{Z^{(k)}_{\mathrm{T}l}} e^{-\frac{D^{(k)2}_{\mathrm{T}:lj}}{\sigma^{(k)2}}}, \tag{4.38}$$

where $Z^{(k)}_{\mathrm{T}l}$ is a normalization term and $\sigma^{(k)}$ is from Eq. (4.28), calculated from the training data set. Given C dissimilarity matrices, $\boldsymbol{D}^{(1)}_{\mathrm{T}}, \boldsymbol{D}^{(2)}_{\mathrm{T}}, \cdots, \boldsymbol{D}^{(C)}_{\mathrm{T}}$, as in the training phase, we can get C probability matrices, $\boldsymbol{P}^{(1)}_{\mathrm{T}}, \boldsymbol{P}^{(2)}_{\mathrm{T}}, \cdots, \boldsymbol{P}^{(C)}_{\mathrm{T}}$. With the same assumption, the compromised probability matrix, $\boldsymbol{P}^{*}_{\mathrm{T}}$, is calculated by

$$P^{*}_{\mathrm{T}:lj} = \frac{1}{Z^{*}_{\mathrm{T}l}} \prod_{k=1}^{C} P^{(k)}_{\mathrm{T}:lj}, \tag{4.39}$$

where $Z^{*}_{\mathrm{T}l}$ is a normalization term. From $\boldsymbol{P}^{*}_{\mathrm{T}}$, we reconstruct the compromised dissimilarity $\boldsymbol{D}^{*}_{\mathrm{T}}$ again,

$$D^{*}_{\mathrm{T}:lj} = \sigma^{*} \sqrt{-\log(P^{*}_{\mathrm{T}:lj})}. \tag{4.40}$$

The remaining part after this is exactly the same as kernel Isomap. That is, the embedding of a test data point $\boldsymbol{x}_l$ in the low-dimensional space is done by

$$\boldsymbol{y}_{li} = \frac{1}{\sqrt{\Lambda_{ii}}} \sum_{j=1}^{N} \boldsymbol{V}_{ij} k(\boldsymbol{x}_l, \boldsymbol{x}_j), \tag{4.41}$$

where the kernel for the test data point $\boldsymbol{x}_l$, $k(\boldsymbol{x}_l, \boldsymbol{x}_j)$, is computed as

$$k(\boldsymbol{x}_l, \boldsymbol{x}_j) = -\frac{1}{2}\left(\widetilde{D}^{*2}_{\mathrm{T}:lj} - \frac{1}{N}\sum_{i=1}^{N} \widetilde{D}^{*2}_{\mathrm{T}:li} + \frac{1}{N}\sum_{i=1}^{N} \widetilde{K}_{ii} - \widetilde{K}_{jj} \right), \tag{4.42}$$

where $\widetilde{\boldsymbol{K}}$ is from Eq. (4.31) and $\widetilde{D}^{*}_{\mathrm{T}:lj}$ is calculated by

$$\widetilde{D}^{*}_{\mathrm{T}:lj} = D^{*}_{\mathrm{T}:lj} + c, \tag{4.43}$$

where c is same as in Eq. (4.33). See [10] for the details.

4. Experiments

In order to show the useful behavior of the proposed method, I carried out experiments with three different data sets: (a) disc data set made of 100 discs with different colors and sizes; (2) head-related impulse response (HRIR) data [46]; and (3) face data [71].

Discs: I made an artificial data set to show the differences between the three methods: (1) RKKS, (2) DISTATIS, and (3) RAMS. Fig. 38 shows 100 discs, where the horizontal axis represents color and the vertical axis represents size. Actually, 100 points were generated on a 10×10 lattice with added random noise in location.

Let $\boldsymbol{X} \in \mathbb{R}^{2\times 100}$ be the discs' locations. The first row $\boldsymbol{X}_1$ and the second row $\boldsymbol{X}_2$ are the coordinates for color and size, respectively. From this disc data set, I made 3 pairs of measurements: (1) Orthogonal and linear case, (2) correlated but still linear case and (3) orthogonal but nonlinear case. Each case is calculated by the following equations.

(1) Orthogonal and linear: Each distance matrix is obtained by only color or

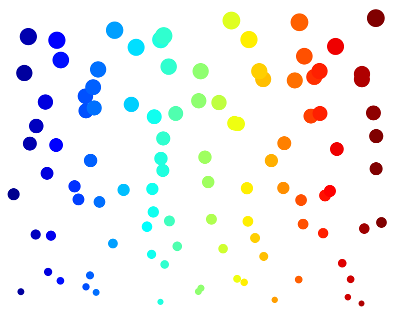


Fig. 38. Disc data set in 2D.

size, respectively,

$$D_{ij}^{(k)} = dist(X_{k,i}, X_{k,j}),$$

where $k = 1, 2$, and $dist(x, y)$ is the Euclidean distance function between two points x and y.

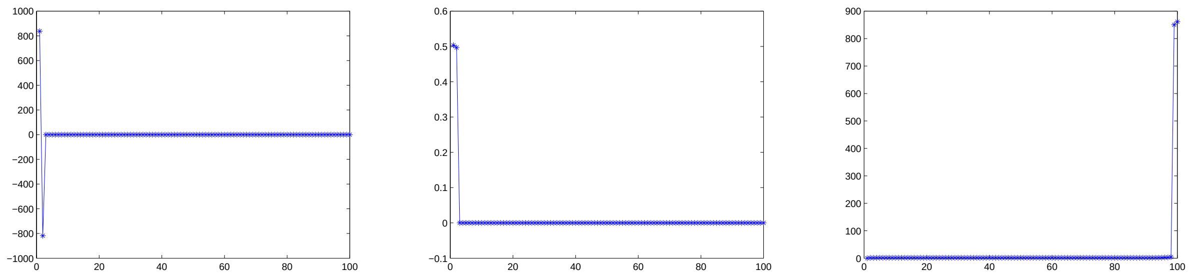
(2) Nonorthogonal and linear: one distance matrix is based on only color, and the other is based on both color and size;

$$D_{ij}^{(k)} = dist(X_{1:k,i}, X_{1:k,j}).$$

(3) Orthogonal and nonlinear: Each distance matrix is obtained by only color or size, respectively, and squared.

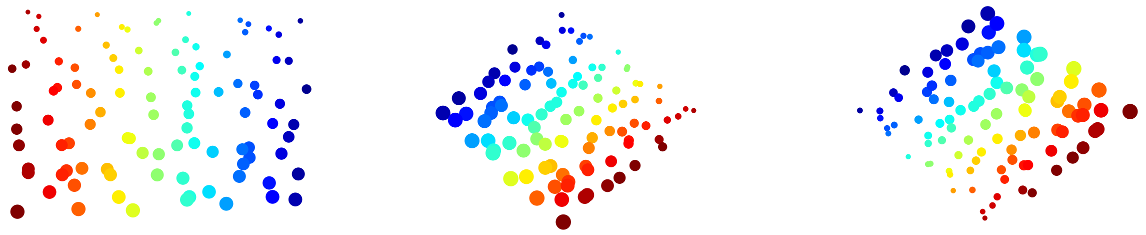
$$D_{ij}^{(k)} = dist(X_{k,i}, X_{k,j})^2,$$

Orthogonal and linear case: Fig. 39 shows the eigenvalues for each method. All three methods found 2 dominant eigenvalues. RKKS uses positive and negative values. Fig. 40 shows projection results for each method. Even though they are rotated, all three methods had no problem in finding the relative locations of the discs. In this case, two distance matrices are uncorrelated and each distance is obtained



(a) (b) (c)

Fig. 39. Eigenvalues of discs in the case of orthogonal and linear distances. (a) RKKS, (b) DISTATIS and (c) RAMS.



(a) (b) (c)

Fig. 40. Projections of discs for the orthogonal and linear case. (a) RKKS, (b) DISTATIS and (c) RAMS.

linearly. So, RKKS and DISTATIS also work well as well as RAMS.

Nonorthogonal or nonlinear cases: When the distances are measured nonorthogonaly or nonlinearly, RKKS and DISTATIS failed to find true locations, even though they are able to estimate along one coordinate, either size or color, but not both. Fig. 41 is the result of correlated but linear case. As we expected, RKKS found just one coordinate (size) and failed to find the color axis. On the other hand, RAMS still found proper coordinates as DISTATIS also did.

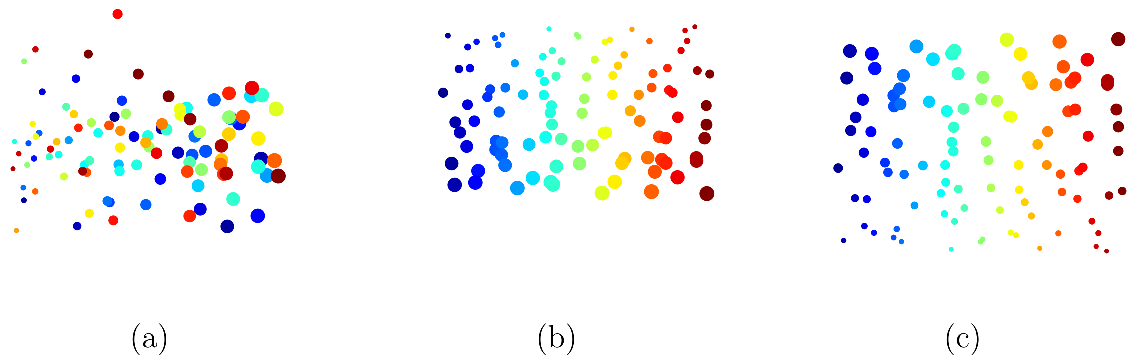


Fig. 41. Projections of discs for the nonorthogonal and linear case. (a) RKKS, (b) DISTATIS and (c) RAMS. RKKS finds the size axis but fails to find the color axis, whereas RAMS and DISTATIS find both successfully.

Fig. 42 is the result of nonlinear but orthogonal case. In contrast to the linear and nonorthogonal case, DISTATIS found the size axis but failed to find the color axis, whereas RAMS found proper coordinates as RKKS also did.

Head-Related Impulse Responses (HRIR): In this experiment, I used the public-domain CIPIC HRTF data set. Detailed description of the data can be found in [46]. It was recently shown in [45] that the low-dimensional manifold of HRIRs could encode perceptual information related to the direction of sound source. However, there has been no attempt to use both the left and the right HRIRs at the same time. I applied kernel Isomap to both left and right, and then tried to merge

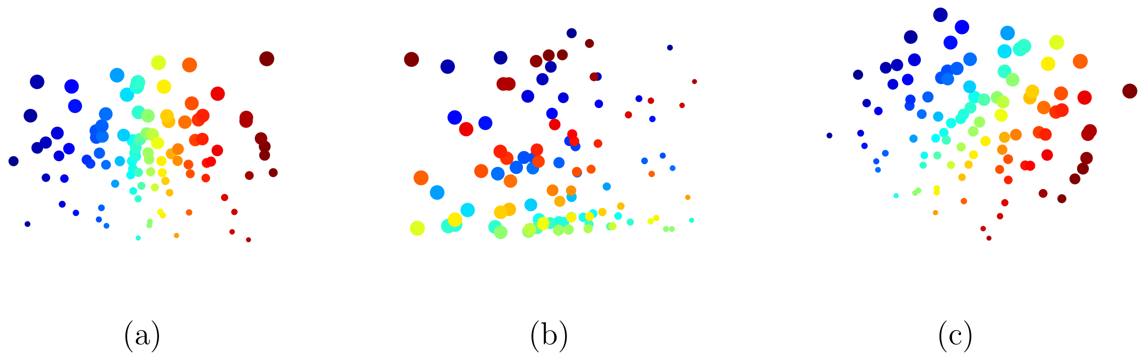


Fig. 42. Projections of discs for the orthogonal and nonlinear case. (a) RKKS, (b) DISTATIS and (c) RAMS. DISTATIS finds the size axis but fails to find the color axis, whereas RAMS and RKKS find both successfully.

the two dissimilarity matrices into one with the three methods (RKKS, DISTATIS, and RAMS).

Two-dimensional manifolds of HRIRs are shown in Fig. 43 for kernel Isomap, where each ear has similar information, with a little distortion. Fig. 44 shows the

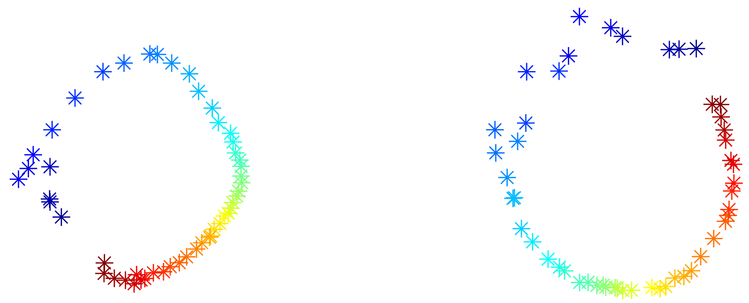


Fig. 43. Two-dimensional manifolds of HRIRs. (left) Left HRIR; (right) Right HRIR. Each ear has similar information with a little distortion. This is the result from the 10th subject.

eigenvalues of three methods, where RAMS and DISTATIS found two dominant eigenvalues of HRIRs because the manifold of each ear's HRIR is almost linear (see [10]), whereas RKKS failed because these two pieces of information from the two ears are strongly correlated. Fig. 45 shows the embedded manifolds of the two methods and confirms the result. RKKS did not work here, so the results are not shown.

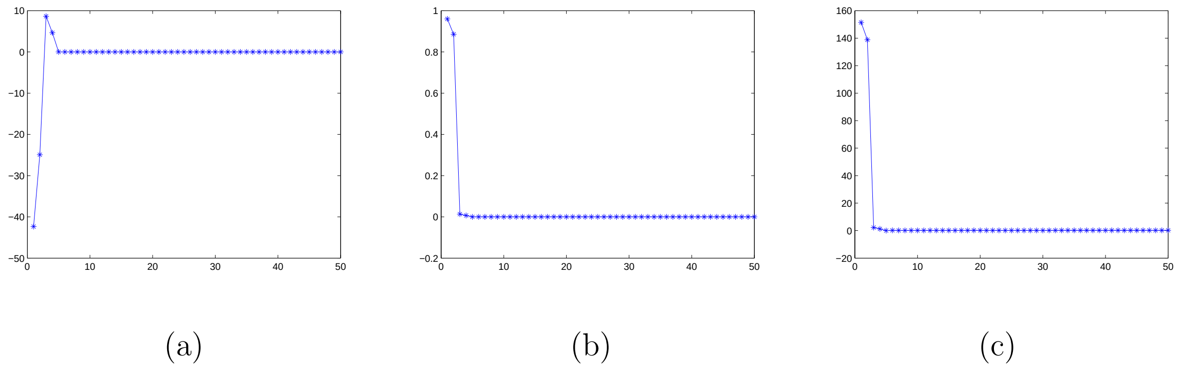


Fig. 44. Eigenvalues of two ears' HRIRs. (a) RKKS, (b) DISTATIS and (c) RAMS.

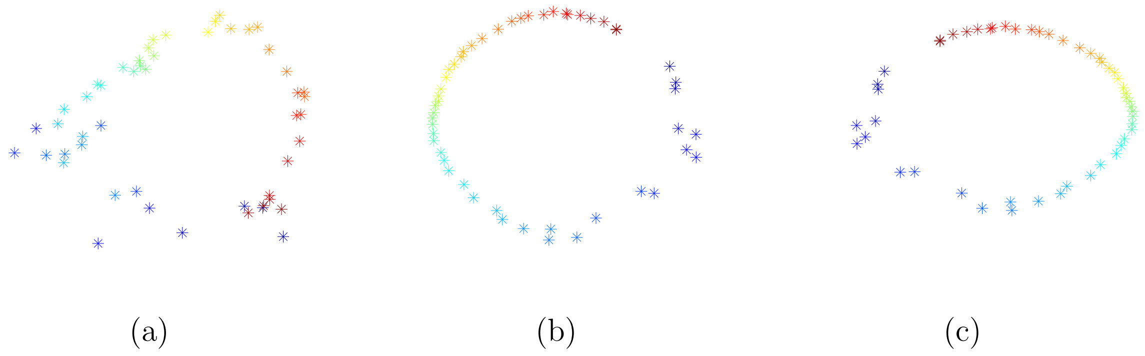
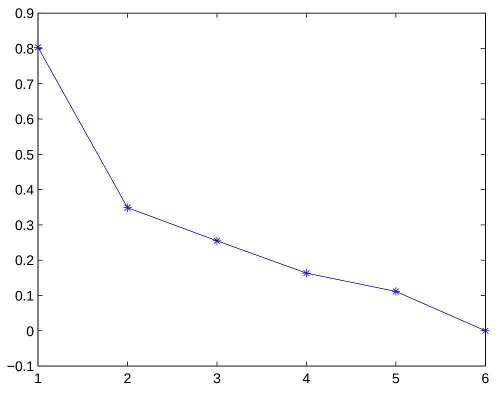


Fig. 45. Projected results on compromised 2-dimensional manifold. (a) RKKS, (b) DISTATIS and (c) RAMS. DISTATIS and RAMS work well.

Faces: I used the face data set from [71], to compare RAMS with DISTATIS since RKKS does not work with more than two dissimilarity matrices. Abdi made four distance matrices and found a 2-dimensional space to represent 6 face images.

Fig. 46 shows the eigenvalues and Table VI compares the 2 top eigenvalues from the two methods, DISTATIS and RAMS, showing that RAMS is more efficient. Fig. 47 shows the 2-dimensional spaces from the two methods. Even though they have



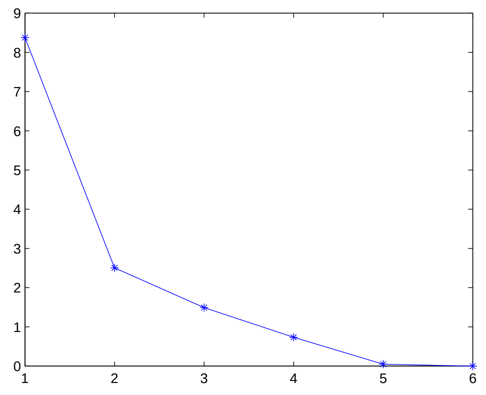


Fig. 46. Eigenvalues of faces. (a) DISTATIS and (b) RAMS.

Table VI. Efficiency of eigenvalues of DISTATIS and RAMS.

Methods	First Eigenvalue	Second Eigenvalue
DISTATIS	47.75%	20.74%
RAMS	63.67%	19.04%

similar results, RAMS produces a little more spread map, because RAMS uses all information whereas DISTATIS uses only the first component of the compromised manifold and discards the rest.

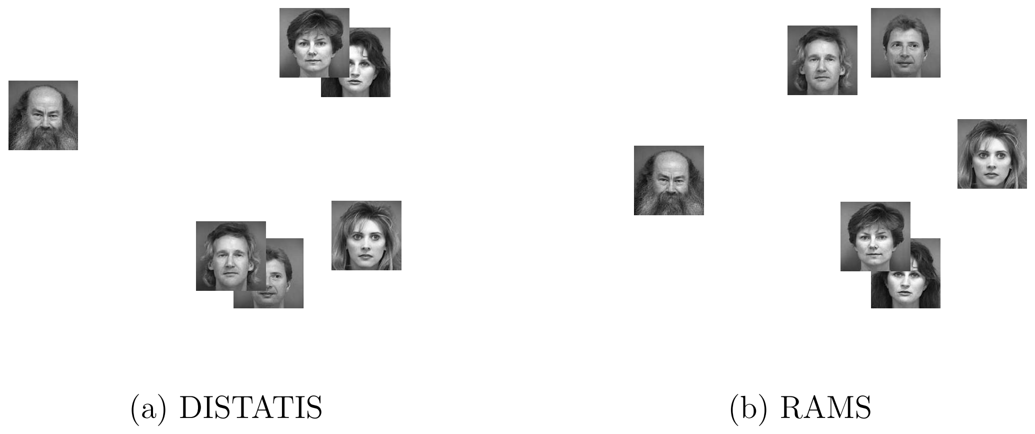

(a) DISTATIS (b) RAMS

Fig. 47. Projections of the faces. RAMS has a more scattered result than DISTATIS.

Projection property: To prove the projection property of RAMS, I just checked with 3 cases of discs above. I used 100 training data points as in Fig. 38, and 100 test data points generated in the same way as the training points. In Fig. 48 (a), blue crosses are training data points and colored and different sized discs are test data points in the original space. I projected the test data points into the compromised manifold of the training data based on the two distance matrices. Fig. 48 (b),(c) and (d) show the result of projection in RAMS in all three cases. As expected, RAMS successfully projected the test data points on the compromised manifold in all three cases.

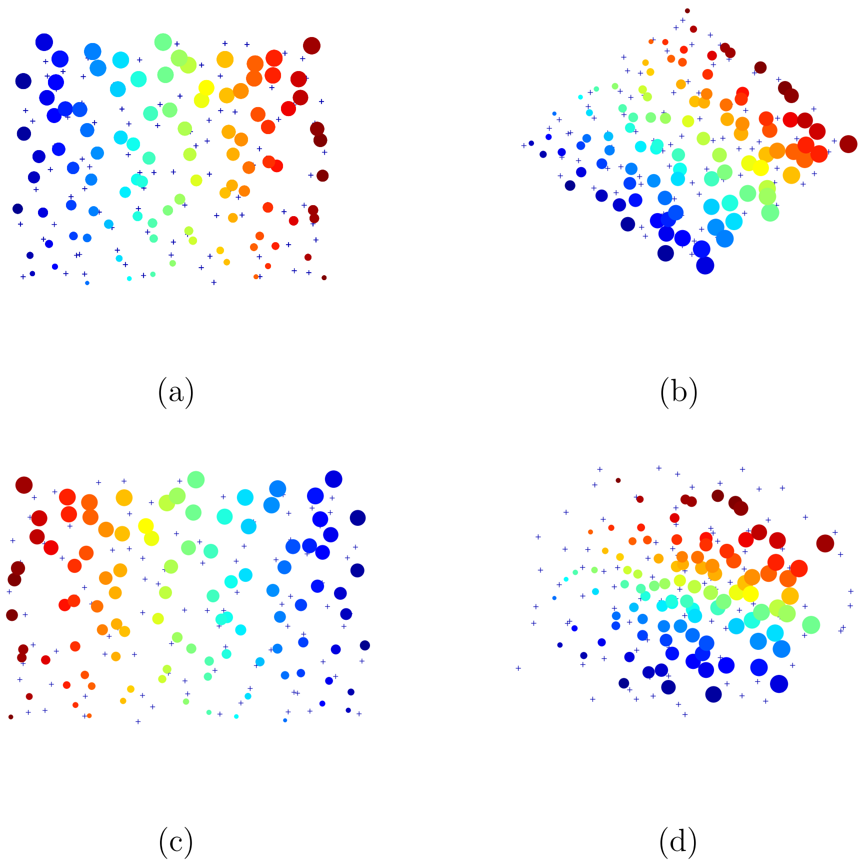


Fig. 48. Projection property of RAMS. Training data set (blue crosses) and test data set (colored and differently sized discs) are shown. Test of the projection property of RAMS with (a) orthogonal and linear, (b) nonorthogonal and linear, (c) orthogonal and nonlinear, and (d) nonorthogonal and nonlinear cases show good mapping along both the color and the size axes.

5. Discussion

In RKKS, even though two measurements are obtained independently, if their values are affected by each other, then the final manifold is distorted because RKKS assumes that the two Hilbert spaces are orthogonal. However, in RAMS, even though the measurements are related to each other, it gives a proper mapping because the two dissimilarity matrices are measured in a statistically independent way. Another interesting issue is that in RKKS, it is not straightforward to extend the number of measurements to more than 2, while RAMS can be naturally extended.

DISTATIS uses the principal component of the manifolds to project data sets into the compromised manifold. That results in loss of information that could be found in the other components. When the distances are measured nonlinearly, such as in squareness, DISTATIS fails to find a proper mapping, because nonlinear manifold does not guarantee that one linear component of manifolds can contain most of the information. However, in RAMS, the final manifold is not a linear combination but rather a probabilistic combination, so it is robust even with nonlinear manifolds.

More interestingly, RKKS and DISTATIS are not concerned with the projection property, while RAMS is. So, RAMS can be considered as the generalization of RKKS and DISTATIS. An interesting direction for future work is to compare with other data fusion theories and extend this algorithm.

However, since statistical independence is still a strong assumption, I propose a more generalized integration method in the next section.

E. Manifold α-Integration

In this section, I propose a new MI algorithm, *manifold α-integration* (MAI) that makes use of α-integration proposed by [35], applying it on the transition probabilities (or distances) from one data point to the others in RAMS. Statistical independence in RAMS may be too strong an assumption for some data sets. In the RAMS algorithm, this assumption is exploited by using multiplication of the transition probabilities. This multiplication is a squared geometric mean of the probabilities and geometric mean is a special case of α-integration with $\alpha = 1$. So, α-integration can generalize RAMS with different α values, which leads to MAI.

I show that this method includes as its special case other existing methods such as statistical distance in RAMS, kernel-based data fusion method [21] or mixture of random walks [80] by analyzing the compromised distances on the integrated manifold. Moreover, since MAI uses kernel Isomap after getting one dissimilarity matrix, MAI inherits the desirable projection property from kernel Isomap [10], which is necessary for classification tasks.

1. MAI Algorithm

Let $\boldsymbol{G}$ be a weighted graph with N nodes, to represent a manifold. Then, the distance between two nodes on the kth manifold, $D_{ij}^{(k)}$, can be transformed into probability $P_{ij}^{(k)}$, the transition probability from the ith node to the jth node on the kth manifold. We simply use the Gaussian kernel which is given by

$$P_{ij}^{(k)} = \frac{1}{Z_i^{(k)}} e^{-\frac{D_{ij}^{(k)2}}{\sigma^{(k)2}}}, \tag{4.44}$$

where $Z_i^{(k)}$ is a normalization term so that the sum of transition probabilities from the ith node to all other nodes on the kth manifold is 1, and $\sigma^{(k)}$ is a parameter representing the standard deviation. Given C dissimilarity matrices, $\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}, \cdots, \boldsymbol{D}^{(C)}$, we can get C probability matrices, $\boldsymbol{P}^{(1)}, \boldsymbol{P}^{(2)}, \cdots, \boldsymbol{P}^{(C)}$.

There are two approaches in using α-integration on multiple manifolds: (1) using transition probability matrices and (2) using distance matrices. First, the transition probability from the ith node to the jth node on the kth manifold is given by Eq. (4.44). So, given $\boldsymbol{P}^{(1)}, \boldsymbol{P}^{(2)}, \cdots, \boldsymbol{P}^{(C)}$, with α-integration, the compromised probability is given by

$$P_{\alpha,ij} = \frac{1}{Z_{\alpha,i}} f_\alpha^{-1}\left(\sum_{k=1}^{C} w_k f_\alpha(P_{ij}^{(k)})\right), \tag{4.45}$$

where $Z_{\alpha,i}$ is a normalization term. From the compromised probability $\boldsymbol{P}_\alpha$, we can reconstruct the compromised dissimilarity $\boldsymbol{D}_{\alpha\mathrm{p}}$ as follows.

$$D_{\alpha\mathrm{p},ij} = \sigma^* \sqrt{-\log(P_{\alpha,ij})}, \tag{4.46}$$

where σ^* is the average of $\sigma^{(k)}, k = 1, ..., C$.

By performing α-integration on probabilities, we can have a generalized integration of the probabilities. For example, with $\alpha = 1$, the integration result is similar to Bayesian inference (or the product of experts) where the likelihood and the prior distributions are multiplied. Or, with $\alpha = -1$, the integration result is similar to the mixture of experts or Gaussian mixture model.

Once $D_{\alpha\mathrm{p},ij}$ is obtained, I use kernel Isomap [10]. Given a distance matrix, $\boldsymbol{D}_{\alpha\mathrm{p}}$, I substitute $\widetilde{\boldsymbol{D}}_{\alpha\mathrm{p}}$ for $\boldsymbol{D}_{\alpha\mathrm{p}}$, which is given by

$$\widetilde{D}_{\alpha\mathrm{p},ij} = D_{\alpha\mathrm{p},ij} + c(1 - \delta_{ij}), \tag{4.47}$$

where δ_{ij} is the Kronecker delta. Here, c is the solution of constant-shifting method [43] to make the doubly centered kernel matrix positive semidefinite given by

$$\widetilde{\boldsymbol{K}} = -\frac{1}{2}\boldsymbol{H}\widetilde{\boldsymbol{D}}^2_{\alpha\mathrm{p}}\boldsymbol{H}. \tag{4.48}$$

Here, $\widetilde{\boldsymbol{D}}^2_{\alpha\mathrm{p}}$ is the element-wise square of $\widetilde{\boldsymbol{D}}_{\alpha\mathrm{p}}$ and $\boldsymbol{H}$ is a centering matrix as in Eq.

(2.2). Finally, after eigen-decomposition,

$$\widetilde{\boldsymbol{K}} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\top}, \tag{4.49}$$

projection mapping $\boldsymbol{Y}$ is given by

$$\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{1}{2}}. \tag{4.50}$$

A more interesting approach is to apply α-integration to the distance matrices directly. Given C dissimilarity matrices, $\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}, \cdots, \boldsymbol{D}^{(C)}$, we can reconstruct the compromised dissimilarity $\boldsymbol{D}_{\alpha\mathrm{d}}$ directly by α-integration without considering the transition probability. It is given by

$$D_{\alpha\mathrm{d},ij} = f_{\alpha}^{-1}\left(\sum_{k=1}^{C} w_k f_{\alpha}(D_{ij}^{(k)})\right). \tag{4.51}$$

Note that Eq. (4.45) is an α-integration of the probabilities, while Eq. (4.51) is an α-integration of the distances. So, with Eq. (4.45), the compromised distance in Eq. (4.46) is different from that of Eq. (4.51). I derive two slightly different manifold integration methods from these two different integration approaches, and call the two versions $\mathrm{MAI_p}$ and $\mathrm{MAI_d}$, respectively. After getting $\boldsymbol{D}_{\alpha}$ (either $\boldsymbol{D}_{\alpha\mathrm{p}}$ or $\boldsymbol{D}_{\alpha\mathrm{d}}$), the rest is the same as kernel Isomap [10], by which my method inherits the dimensionality reduction property.

MAI also has the projection property which involves the projection of novel data points onto the associated low-dimensional space. In other words, with the projection property it is able to determine the location of a point $\boldsymbol{y}_l$ embedded in the low-dimensional space, given a test data point $\boldsymbol{x}_l$. I do not derive the equations for the projection here. The derivations for the projection property is the almost same as in RAMS.

Note that $\boldsymbol{D}_{\alpha\mathrm{d}}$ might not be a true distance in mathematical terms. For example, suppose that we have three points. Then, let the three distances for two measurements be $d_i^{(k)}$, with $k = 1, 2$ and $i = 1, 2, 3$ as in Table VII. As shown in the table, when the $\alpha = \infty$, the α-integration results do not satisfy the distance property (triangle inequality) since $d_1 + d_2 < d_3$. This is another point of using kernel Isomap that makes the integration of multiple distances mathematically distance implicitly by making the kernel matrix positive semidefinite.

Table VII. A counter example shows that the integration of multiple distances might not be a mathematical distance.

Distance	d_1	d_2	d_3
$d^{(1)}$	1	10	10
$d^{(2)}$	10	1	10
Integration with $\alpha = \infty$	1	1	10

2. Comparison with Existing Integration Approaches

I analyze $\mathrm{MAI_p}$ and $\mathrm{MAI_d}$, comparing them to previous methods [77, 21, 80] even though some of them are not immediately about MI.

Case 1: In *random walk on multiple manifolds* (RAMS) [77], the compromised transition probability matrix $\boldsymbol{P}^*$ is simply given by multiplication of the source probabilities. Approximately, this is a special case of $\mathrm{MAI_p}$ in Eq. (4.45) with $\alpha = 1$ and uniform weights for all manifolds:

$$\begin{aligned} P_{1,ij} &= \frac{1}{Z_{1,i}} f_1^{-1}\left(\sum_{k=1}^{C} \frac{1}{C} f_1(P_{ij}^{(k)})\right) \\ &= \frac{1}{Z_{1,i}} \exp\left(\frac{1}{C}\sum_{k=1}^{C} \log P_{ij}^{(k)}\right) \\ &= \frac{1}{Z_{1,i}} \left(\prod_{k=1}^{C} P_{ij}^{(k)}\right)^{\frac{1}{C}} \\ &= \frac{1}{Z_{1',i}} (P_{ij}^{*})^{\frac{1}{C}}, \end{aligned} \tag{4.52}$$

where $Z_{1',i}$ is a normalization term. Then, the compromised distance in Eq. (4.46) is reconstructed by

$$\begin{aligned} D_{1\text{p},ij} &= \sigma^{*}\sqrt{-\log\left(\frac{1}{Z_{1',i}}(P_{ij}^{*})^{\frac{1}{C}}\right)} \\ &= \sigma^{*}\sqrt{\log Z_{1',i} - \frac{1}{C}\log P_{ij}^{*}}, \end{aligned} \tag{4.53}$$

which is almost the same as the compromised distance in RAMS except for the normalization term and $\frac{1}{C}$. That is, RAMS is approximately a special case of MAI_p with $\alpha = 1$ and uniform weights. If we relax the assumption on the weights so that the sum of weights is not 1 but $\sum_{k=1}^{C} w_k = C$, then MAI_p leads to an exactly the same result as RAMS.

Now, we can check the case when α-integration is applied to the distance matrices directly (MAI_d). When $\alpha = -3$ and the weights are given by $\frac{1}{\sigma^{(k)2}}$, the compromised distance matrix of MAI_d in Eq. (4.51) is given by

$$\begin{aligned} D_{-3\text{d},ij} &= f_{-3}^{-1}\left(\sum_{k=1}^{C} \frac{1}{\sigma^{(k)2}} f_{-3}(D_{ij}^{(k)})\right) \\ &= \sqrt{\sum_{k=1}^{C} \frac{1}{\sigma^{(k)2}} (D_{ij}^{(k)})^2}, \end{aligned} \tag{4.54}$$

which is the same as the the compromised distance matrix $\boldsymbol{D}^*$ in RAMS except the normalization terms. Here, the weights $\frac{1}{\sigma^{(k)2}}$ can serve as normalization terms for different units across measurements.

Case 2: Lanckriet [21] used a weighted sum of kernel matrices for kernel-based data fusion. For the special case when $\alpha = -3$ and weight w_k for the kth manifold applied to $\mathrm{MAI_d}$, the corresponding kernel matrix $\boldsymbol{K}_{\mathrm{MAI_d}}$ is given by just the weighted average of the kernel matrices as follows:

$$\begin{aligned} \boldsymbol{K}_{\mathrm{MAI_d}} &= -\frac{1}{2}\boldsymbol{H}(\boldsymbol{D}_{-3\mathrm{d}})^2\boldsymbol{H} \\ &= -\frac{1}{2}\boldsymbol{H}\left(\sum_{k=1}^{C} w_k \boldsymbol{D}^{(k)2}\right)\boldsymbol{H} \\ &= \sum_{k=1}^{C} w_k \boldsymbol{K}^{(k)}, \end{aligned} \tag{4.55}$$

where

$$\boldsymbol{K}^{(k)} = -\frac{1}{2}\boldsymbol{H}\boldsymbol{D}^{(k)2}\boldsymbol{H}. \tag{4.56}$$

Notice that Eq. (4.55) is the kernel-based data fusion proposed in [21] which can now be seen as a special case of $\mathrm{MAI_d}$. It was shown that manipulating the distance matrix gives a better result than manipulating the kernel matrix directly [10, 77]. In other words, the integrated space of $\mathrm{MAI_d}$ is better than (or at least equal to) the kernel-based data fusion methods when the α value is carefully chosen. Also note that with w_k the same, Eq. (4.55) is equivalent to stacking multiple measurements in a single vector.

Case 3: Also, in [80], even though they did not discuss directly about MI, a mixture of random walks was used as an integration method. With $\alpha = -1$ and different weights w_k for the kth manifold, $\mathrm{MAI_p}$ has

$$\begin{aligned} P_{-1,ij} &= \frac{1}{Z_{-1,i}} f_{-1}^{-1}\left(\sum_{k=1}^{C} w_k f_{-1}(P_{ij}^{(k)})\right) \\ &= \frac{1}{Z_{-1,i}} \sum_{k}^{C} w_{ki} P_{ij}^{(k)}, \end{aligned} \tag{4.57}$$

which is a mixture of random walks.

In sum, I checked three previous methods for data integration and compared them with the two proposed approaches. The previous approaches all turn out to be a special case of the proposed method, though $\mathrm{MAI_p}$ is approximately the same as RAMS.

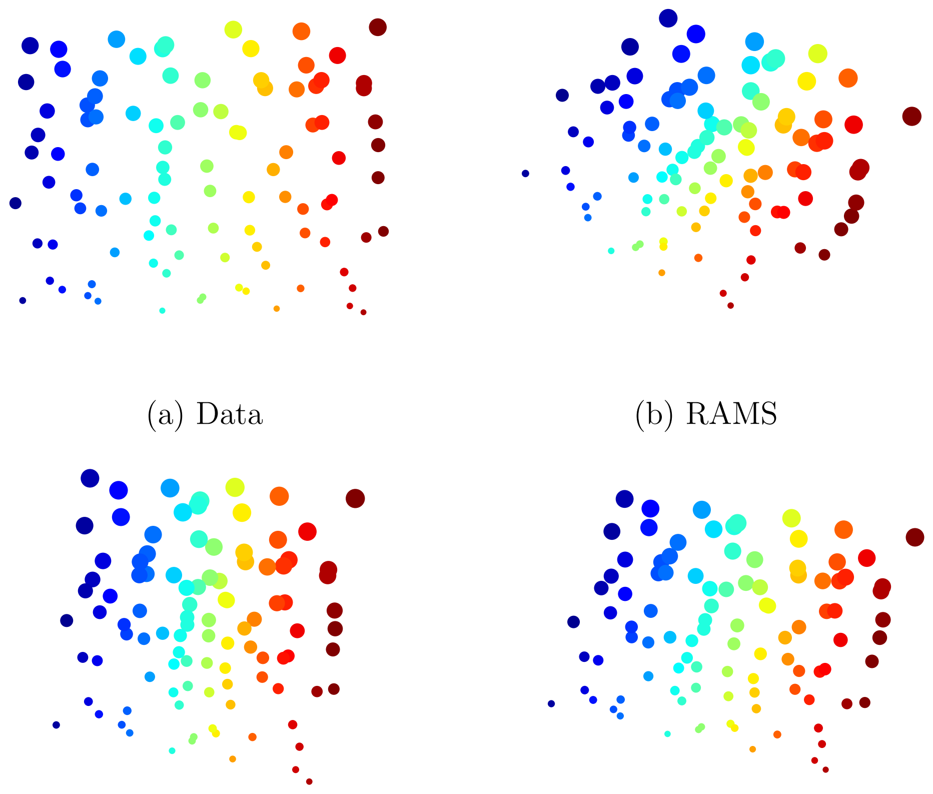
3. Experiments

In order to show the effectiveness of my method, I carried out experiments with four different data sets: (1) disc data set made of 100 discs with different colors and sizes [77]; (2) head-related impulse response (HRIR) data [46]; (3) the CMU ARCTIC speech database [75]; and (4) sensorimotor integration.

Disc data: I used an artificial disc data set [77] to show the differences between the three methods: (1) RAMS [77], (2) MAI on transition probability matrices ($\mathrm{MAI_p}$), and (3) MAI on distance matrices ($\mathrm{MAI_d}$). Let $\boldsymbol{X} \in \mathbb{R}^{2\times 100}$ be the discs' locations. The first row $\boldsymbol{X}_1$ and the second row $\boldsymbol{X}_2$ are the coordinates for color and size, respectively. From this disc data set, each distance matrix is obtained by only color or by size, and squared as follows.

$$\boldsymbol{D}_{ij}^{(k)} = dist(X_{k,i}, X_{k,j})^2.$$

Fig. 49 shows the data set and three integrated spaces from the three methods. If we use $\alpha = 1$ and $\alpha = -3$ for $\mathrm{MAI_p}$ and $\mathrm{MAI_d}$, respectively, then the results



(a) Data

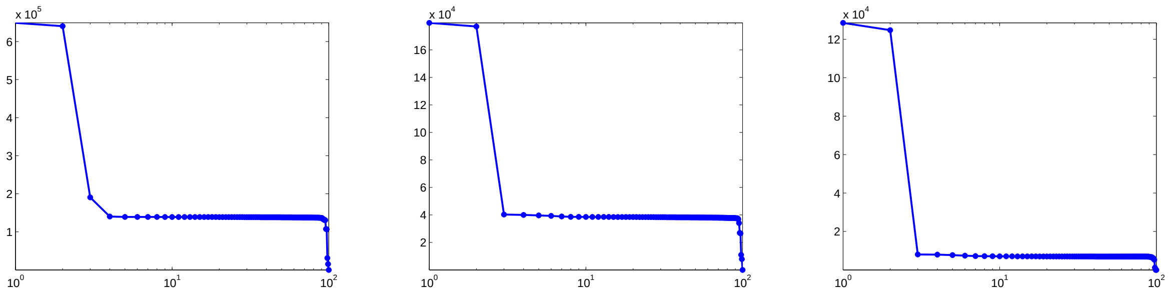
(b) RAMS

(c) $\mathrm{MAI_p}$

(d) $\mathrm{MAI_d}$

Fig. 49. RAMS and MAI on disc data sets. Disc data set (a) and three integrated spaces (b-d); (a) original data set, 100 discs, (b) RAMS, (c) $\mathrm{MAI_p}$ with $\alpha = 0.89$, and (d) $\mathrm{MAI_d}$ with $\alpha = -1.25$.

of MAI are the same as RAMS. Here, I chose the α values to be 0.89 and -1.25 for $\mathrm{MAI_p}$ and $\mathrm{MAI_d}$, respectively. Note that the integrated space from MAI have almost a square shape, which is supposed to be like that, while RAMS has a fat square even though it found a "properly" integrated space where color and size are two dominant coordinates. In addition, $\mathrm{MAI_d}$ found almost the same result as the original set, while $\mathrm{MAI_p}$ has a little denser dots around the center of the space. Even though $\mathrm{MAI_p}$ generalizes RAMS with the same transition probability matrices, the reconstructed distance matrix is not optimal in the α-integration sense, because the transition probability equation in Eq. (4.44) is combined into f_α, which is not a linear scale free function of distance any more. This can be why $\mathrm{MAI_p}$ has a little distortion in the integrated space, even though it is still better than RAMS.

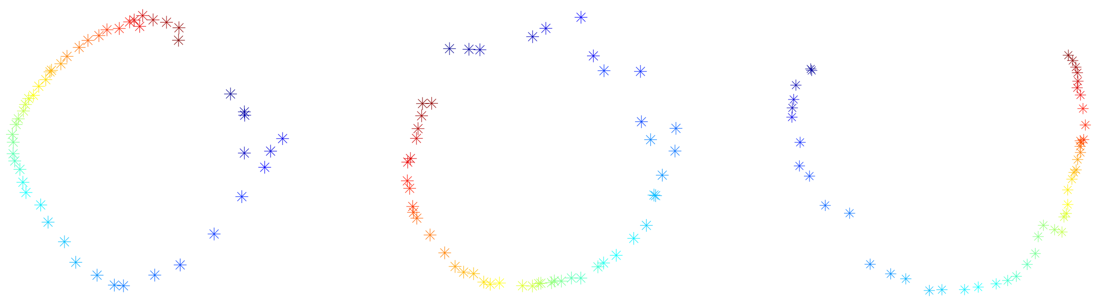


(a) RAMS (b) $\mathrm{MAI_p}$ (c) $\mathrm{MAI_d}$

Fig. 50. Eigenvalues of the kernel matrices in the three methods. (a) RAMS, (b) $\mathrm{MAI_p}$, and (c) $\mathrm{MAI_d}$. The x-axes (index for eigenvalues) are in log-scale to highlight the first few eigenvalues.

Moreover, in Fig. 50, all three methods (after making the kernel matrix positive semidefinite) have two dominant eigenvalues which are for the size and color. However, RAMS has one more eigenvalue higher than the other majority. $\mathrm{MAI_p}$ and $\mathrm{MAI_d}$ have only two dominant eigenvalues but the majority of eigenvalues in $\mathrm{MAI_p}$ has relatively much higher values than in $\mathrm{MAI_d}$. So, based on the eigenvalues, $\mathrm{MAI_d}$ has more efficient features than RAMS or $\mathrm{MAI_p}$.

HRIR data: In this experiment, I used public-domain CIPIC HRTF data set [46] as in the previous Chapter. First, I applied kernel Isomap to each ear's HRIR data to generate a 2-dimensional manifold for each ear. Then I applied $\mathrm{MAI_d}$ to integrate the two manifolds. We mainly pay attention to the HRIRs involving sound sources specified by different elevation angles.

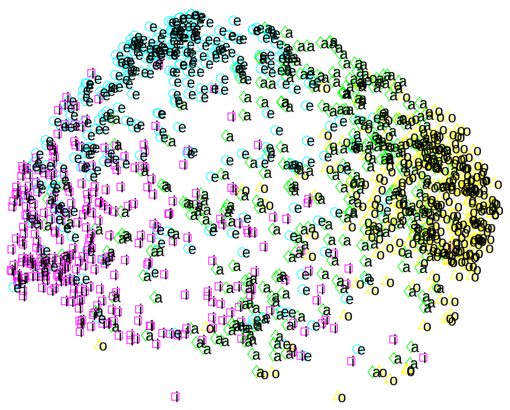


(a) Left only (b) Right only (c) Integrated

Fig. 51. Embedded manifolds of HRIRs. (a) left HRIR (b) right HRIR and (c) integrated HRIR. Even though the left HRIR is seriously distorted and the right one is also not smooth, the integrated space shows a very smooth, low error result, due to the use of both pieces of information.

Fig. 51 shows the performance of the method $\mathrm{MAI_d}$ with $\alpha = -0.5$ on 20th subjects in the data set. $\mathrm{MAI_d}$ was applied to the distance matrices of (a) and (b). Either (a) or (b) is not perfect for locating the sound source. The integrated result is better than the two results considered separately, as to where the sound source is. Note that the embedded manifolds in Fig. 51 have some ambiguities like up-down or front-back.

Speech data: I carried out numerical experiments with the CMU ARCTIC speech database [75] as used in the previous section in order to show an integrated manifold from multiple manifolds which leads to a speaker independent phoneme space and to show the benefit of the integrated space for phoneme classification.


(a) Speaker 'bdl'

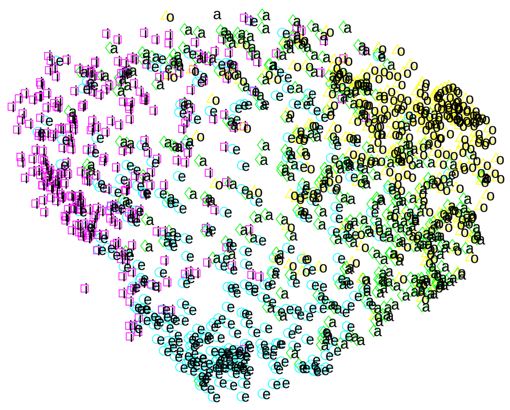

(b) Speaker 'slt'

Fig. 52. Individual mappings using kernel Isomap. (a) speaker 'bdl' and (b) speaker 'slt'. Two maps look different because they are from different speakers even though they are for the same set of phonemes.

First, I found one map of these vowels from four speakers' four vowels, where each phoneme consisted of 300 sample data points. Fig. 52 shows two speakers' individual maps from kernel Isomap. Even though they pronounced the same phonemes, their maps are different from each other. Furthermore, the clusters of phonemes are not well separated even in each map, since each map represents both linguistic information and speaker dependent information.

On the other hand, Fig. 53 is the integrated map from four speakers' maps, and it shows well defined clusters of phonemes, which means that this map represent the phoneme information but not speaker dependent information. In addition, when compared to international phonetic alphabet (IPA) map in Fig. 54, Fig. 53 shows an almost identical layout of vowels. Even though the integrated map does not exactly match with IPA, the integrated map in Fig. 53 looks more like IPA map than the individual maps in Fig. 52 do. If we try more phonemes from more speakers, it can

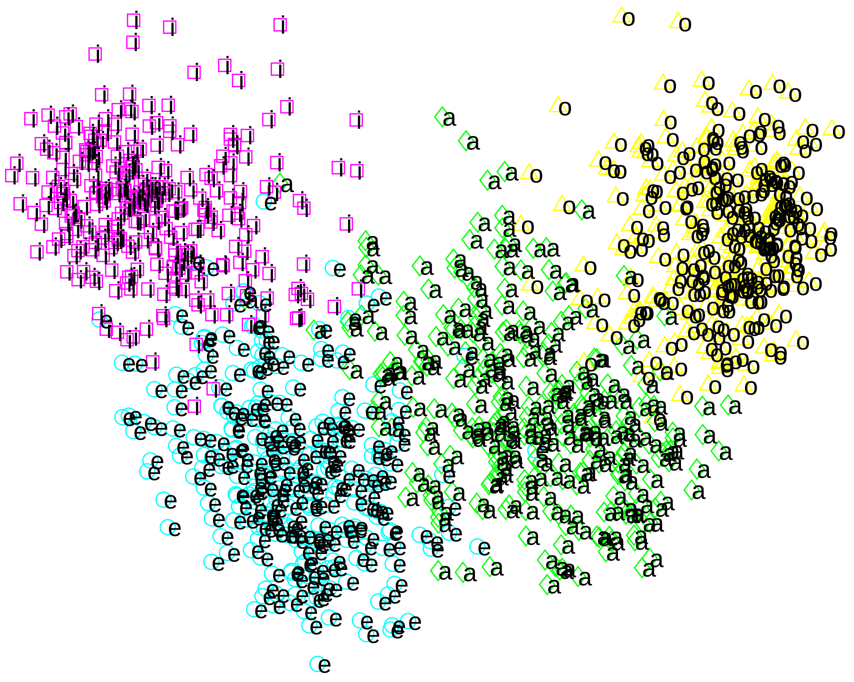


Fig. 53. Integrated phoneme manifold. The map through $\mathrm{MAI_d}$ with $\alpha = 3$, where the phonemes are better clustered within class and separated from each other across class than the individual maps in Fig. 52.

become closer to the Fig. 54.

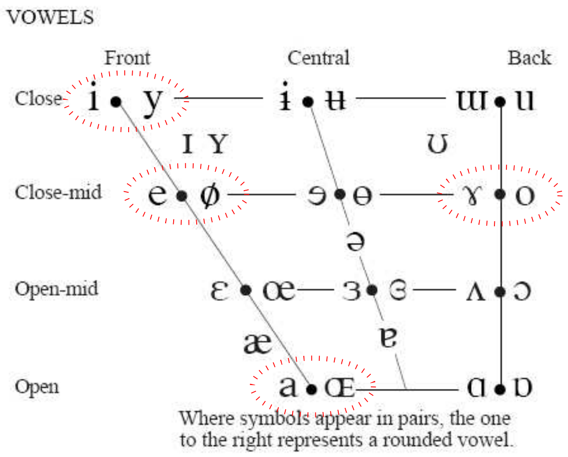


Fig. 54. International phonetic alphabet (IPA) of vowels. The red dotted circles show the vowels which are used in the experiments. Adapted from 'http://www.arts.gla.ac.uk/IPA/vowels.html.'

After getting the integrated map of phonemes, I tried to use this map for classification. I tested it with 6 different training data sizes. For each speaker's individual phoneme, I randomly selected 50, 100, 150, 200, 250, and 300 samples for training and the rest for testing. For each trial, we repeated the experiment 30 times with randomly chosen data points and averaged them.

Fig. 55 shows the classification results with the quadratic classifier. From this figure, we can see that other speakers' information is helpful for phoneme classification as long as the training data set is larger than a certain size. The average of classification rate of individual speaker data converges to 73.8% when 300 phonemes are used for training data, whereas $\mathrm{MAI_d}$, especially with $\alpha = 3$, reaches 76.4%. Note that the performance changes as the α value changes and we can pick the best one to get better results. Here, $\alpha = 3$ is the best among integer values for α.

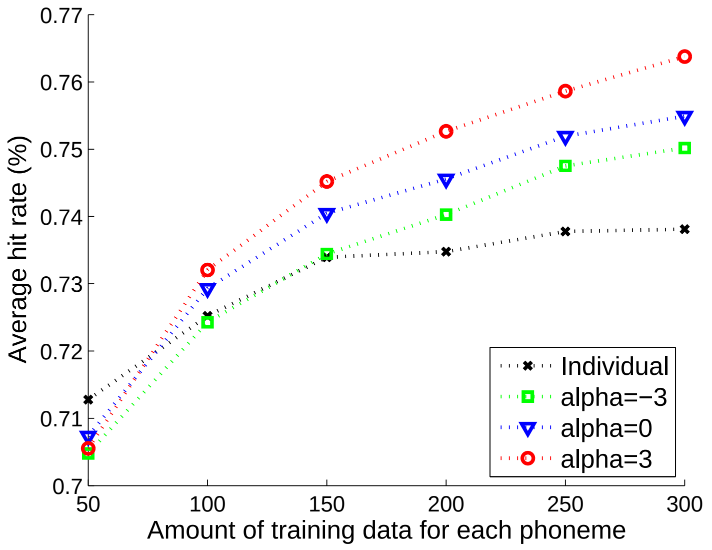


Fig. 55. MAI on classification task. Hit rates for $\mathrm{MAI_d}$ (squares, triangles and circles) and individual map (crosses) are shown. After around 50 phonemes for training, $\mathrm{MAI_d}$ becomes better than the individual map. As shown above, RAMS and kernel-based data fusion are (approximately) a special case of $\mathrm{MAI_d}$ with $\alpha = -3$ (squares).

In Fig. 55, however, when the training data size is smaller than around 80 phonemes, the proposed method is slightly worse than the individual-based map. The intersection is somewhere between 50 and 100. This phenomenon might be explained as follows. When the training data set is small, it is not enough to represent the real phoneme space. So, the test points could have been projected into a distorted map induced by the other speakers' information. But when the training data set is large enough, the projected space represents more likely the real phoneme space. So, from the test points, the speaker dependent noise is removed, which leads to better classification.

4. Discussion

I proposed a generalized MI method utilizing α-integration which led to *manifold α-integration* (MAI). MAI integrates multiple measurements each of which is assumed to lie on a separate manifold. I showed that MAI includes as its special case previous methods such as RAMS, kernel-based data fusion, or mixture of random walks. Furthermore, it can generalize to other integrated spaces in as many different ways as we want with a different α value. The experimental results confirmed that MAI integrates multiple measurements into one manifold in an effective manner, helping us to understand the data set better. For example, when I applied MAI to real world data sets, it found a better manifold than the individual manifolds. In classification tasks, the integrated manifold generally improved the accuracy when the training data set is reasonably large.

One beneficial property of MI is its coordinate invariance since it does not use any coordinate system except the final projection space. It uses only dissimilarity matrices. The performance is invariant to the coordinate systems that the data sets use. Meanwhile, simply staking the multiple data sets in one data space seems simple but its performance depends on which coordinate systems are used for each data set.

F. Summary

In this Chapter, I proposed a new concept, manifold integration, which is a concept combining manifold learning and data integration. Contrary to manifold learning or data integration, it uses both the statistical relation between the two data sets and the geometric relation between data points in each set. The MI approach can serve as an effective framework for analyzing multimodal data sets on multiple manifolds.

Specifically, I derived three algorithms: KODA, RAMS and MAI. They were developed with different assumptions so that they work with different kinds of data sets. However, all of them find a nonlinear mapping rule and can project new points on the same projected space as the training data set. Especially, MAI includes other previous methods as a special case. The three algorithms were tested with synthetic and real world data sets showing encouraging results.

CHAPTER V

ADVANCED APPLICATIONS OF MI

In this chapter, I propose MI as a framework to handle two well-known tasks: kernel integration and sensorimotor integration. Both tasks are not simply to integrate multimodal data sets. However, key parts of the tasks can be interpreted as MI. So, I show that MI can provide an alternative framework for those tasks. Here I use MAI as a representative MI method in the experiments.

A. Kernel Integration

MI can be applied to kernel integration. In this section, I briefly review kernel integration and show how MI can be used as a good framework for kernel integration. My approach includes previous methods as a special case.

1. What is Kernel Integration?

Kernel machines such as support vector machines (SVMs) [81], KPCA and KFD, have been successfully applied to many pattern recognition problems [32, 14, 82, 83, 84]. In all of these algorithms, designing and learning kernel matrices play a key role. These kernel matrices are usually obtained by kernel functions. A kernel function is defined as follows [14].

Definition 5 (Kernel function) *Let $\boldsymbol{X}$ be a data space. If $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j) \rangle$ for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}$ where ϕ is a nonlinear mapping from the data space $\boldsymbol{X}$ to feature space $\mathcal{H}$, then $\boldsymbol{K}(\cdot, \cdot)$ is a kernel function.*

In other words, if the matrix by $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semidefinite, $\boldsymbol{K}(\cdot, \cdot)$ is a kernel function. For example, $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle^d$ for $d \geq 2$ is a polynomial kernel.

Another popular kernel function is the exponential function given by $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2)$. See [14] for more examples of kernel functions and how to make new kernel functions from already known ones.

However, they usually use a single kernel designed (or learned) for one measured data set. When multiple measurements are available as we have discussed earlier, multiple kernel functions can be applied and each kernel function can be tuned up with each measured data for a specific task as in Fig. 56. Usually, these tuned kernel functions can have better kernel matrices than those from manifold learning for a given specific task. After tuning the kernels, each measured data has a different kernel function with different parameters.

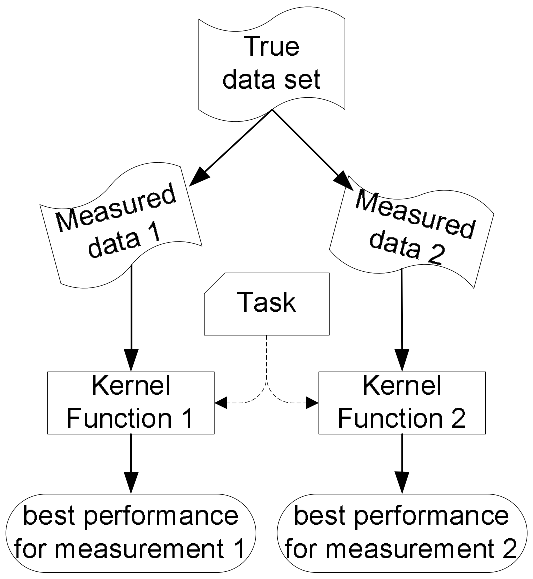


Fig. 56. Multiple kernel machines. Kernel functions can be tuned for multiple measurements separately. These kernel functions are designed to have optimized performance based on each measurement for the task. This approach can be generalized to more than two measurements.

Kernel integration is a process to integrate separately tuned kernels to analyze multiple measurements including heterogeneous data sources in computer vision, bioinformatics, audio processing problems and so on. Recently, Lanckriet proposed kernel fusion, which is a linear weighted sum of kernel matrices [21]. The main issue is to find an optimal way to combine the kernels like the product of kernels and the mixture of kernels.

In this section, I propose MI for combining separately tuned kernel matrices into one integrated kernel matrix, although kernel methods can be viewed as manifold learning.

2. Kernel Integration Based on Manifold Integration

Multiple measurements or multiple processing methods generate multiple data sets where multiple kernel functions can be applied. Given multiple kernel functions (or kernel matrices), my new approach consists of three steps: (1) converting kernel matrices into distance matrices, (2) integrating the distance matrices into one distance matrix, and (3) recovering an integrated kernel matrix from the integrated matrix. In the sense that manifold learning finds a low-dimensional space from a kernel matrix, a kernel matrix is equivalent to a manifold in manifold learning. A detailed procedure of the above approach is summarized in Fig. 57.

Since a kernel matrix itself does not include any target information (as in supervised learning), we need to handle two different cases for kernel integration: supervised case like KFD and unsupervised case like KPCA. I focus on the unsupervised case in this dissertation. The unsupervised version of kernel integration to the supervised case is left for future work.

To apply the MI approach to kernel integration, we need a transformation from a kernel matrix $\boldsymbol{K}$ to a corresponding distance (or dissimilarity) matrix $\boldsymbol{D}$ and vice versa. First, a transformation from a distance matrix to a kernel matrix is well known

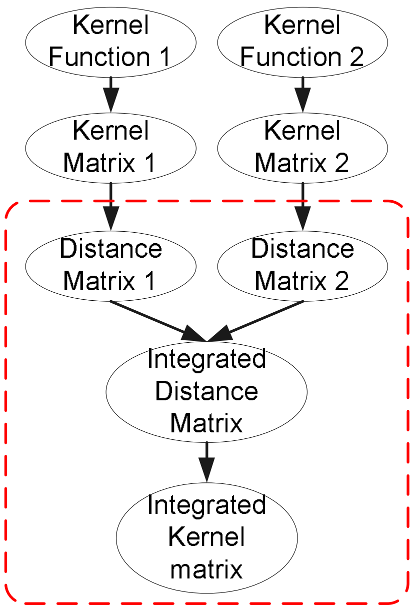


Fig. 57. Kernel integration based on manifold integration. Each kernel function generates a kernel matrix which is converted into a distance matrix. MI shown in the dotted box integrates these distance matrices and reconstructs an integrated kernel matrix. This approach can be applied to more than two kernel matrices.

and can be done as:

$$\boldsymbol{K} = -\frac{1}{2}\boldsymbol{H}\boldsymbol{D}^2\boldsymbol{H}, \tag{5.1}$$

where $\boldsymbol{D}^2 = [D_{ij}^2]$ means the element-wise square of the elements, and $\boldsymbol{H}$ is a centering matrix as in Eq. (2.2). That is, given a dissimilarity matrix that has a metric property, then there is a corresponding kernel matrix that is positive semidefinite.

As for a conversion from a kernel matrix to a corresponding distance matrix, I use the relation between distance and inner products of two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and the transformation is given by

$$\begin{aligned} D_{ij} &= \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)} \\ &= \sqrt{\boldsymbol{x}_i^\top\boldsymbol{x}_i - \boldsymbol{x}_i^\top\boldsymbol{x}_j - \boldsymbol{x}_j^\top\boldsymbol{x}_i + \boldsymbol{x}_j^\top\boldsymbol{x}_j} \\ &= \sqrt{K_{ii} - K_{ij} - K_{ji} + K_{jj}}. \end{aligned} \tag{5.2}$$

This approach can utilize existing kernel machine algorithms where the individual kernel machines on their own cannot give optimal results. For example, for texture images, visual processing and haptic processing can generate two different receptive fields (RFs): tactile receptive field (TRF) and visual receptive fields (VRF) as shown in [85]. These RFs can be processed by two different kernel machines separately, as shown in [86]. The proposed kernel integration approach can be applied to such a case to improve the performance of classification (note that in [86] KFDs were used which are supervised kernel machines so that the current approach cannot be applied to their RFs directly).

3. Simulation

In order to show how MI can work for multiple kernel matrices, I carried out experiment with a synthetic data set. The data set had 672 samples $\boldsymbol{x} \in [0,2] \times [0,2]$ and the kernel matrix (or inner product matrix) was 672×672.

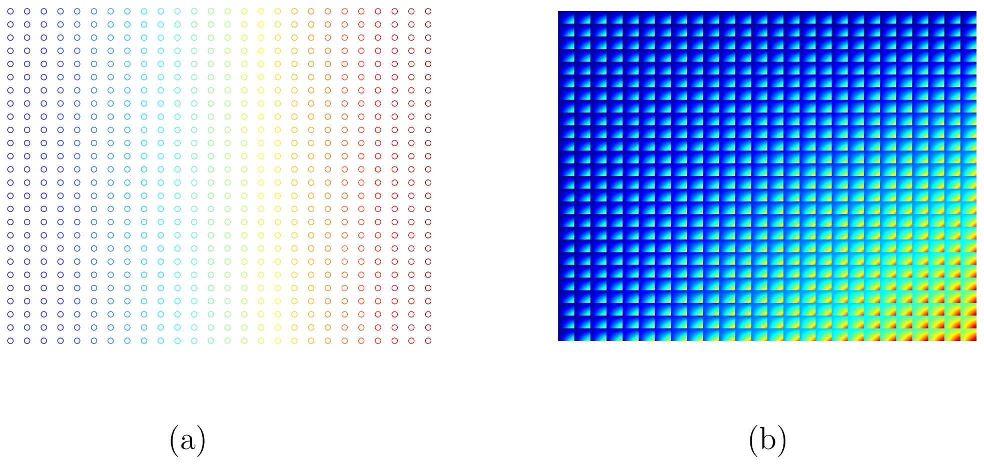


Fig. 58. The true map and the kernel matrix of a square data set in a two-dimensional space. (a) True map and (b) True kernel matrix.

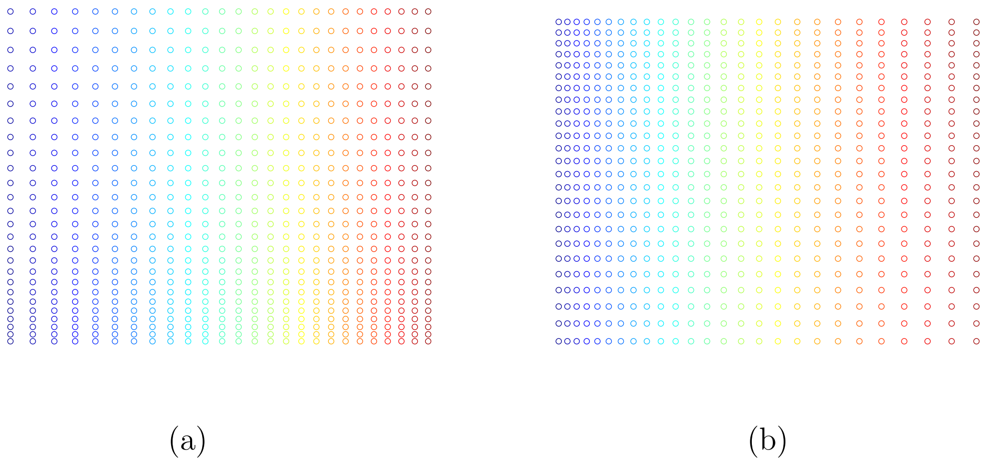


Fig. 59. Two measurements of a square data set. (a) Measurement $\boldsymbol{m}_1$ and (b) measurement $\boldsymbol{m}_2$.

The true map of the square data set in a two-dimensional space is shown in Fig.

58 (a) and the kernel matrix is shown in Fig. 58 (b). This data set was measured by two different distorted functions as shown in Fig. 59. In this simulation, given a true point $\boldsymbol{x} = (x_1, x_2)$, I generated two measurements as follows.

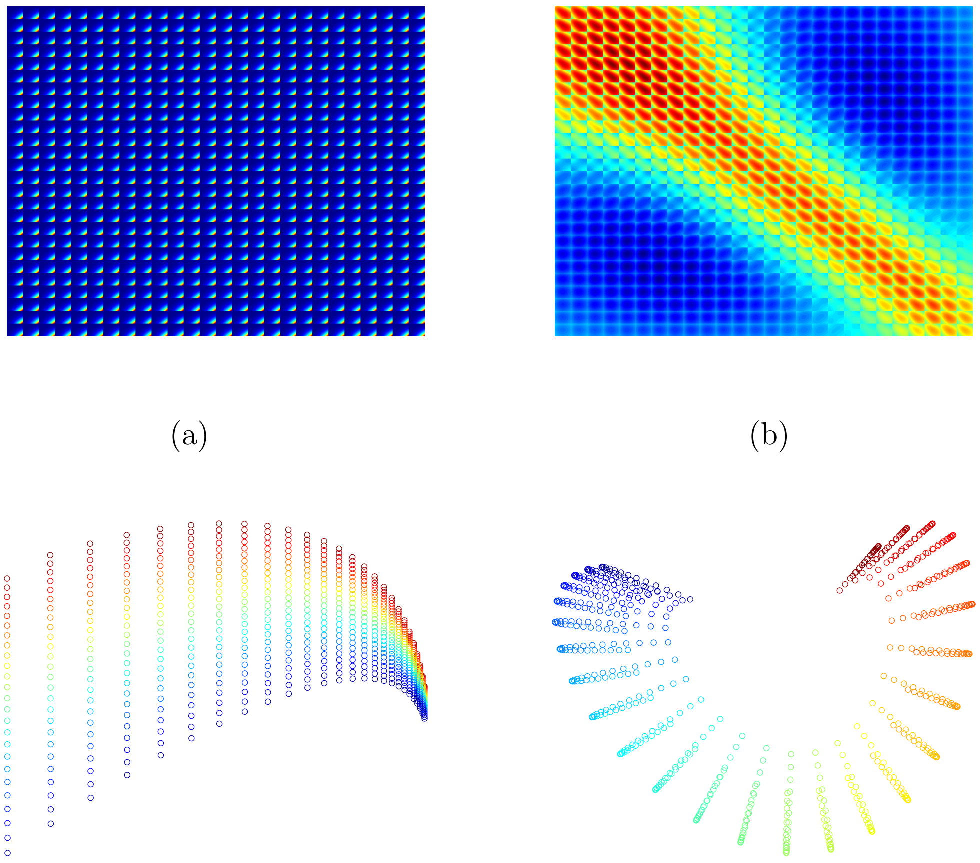
$$\boldsymbol{m}_1 = (\sqrt{x_1}, 4x_2^2), \tag{5.3}$$

$$\boldsymbol{m}_2 = (x_1^2, 4\sqrt{x_2}). \tag{5.4}$$

Then the corresponding kernel functions are "tuned" for the distorted measurements. Here the kernel functions are polynomial and exponential functions. I assume that these kernel functions are given. From these kernel functions, two kernel matrices can be obtained as in Fig. 60 (a-b). Then the mapping rules are obtained as in KPCA or kernel Isomap and the projected spaces are shown in Fig. 60 (c-d).

In Fig. 60, the maps do not reflect the true map, but they are topologically isomorphic and the points are dense enough to apply manifold integration. I applied two different integration methods: linear weighted sum of kernel matrices and MI. Fig. 61 shows the two integrated kernels and the corresponding spaces. The projected space by MI looks more similar to the true map than the one by the linearly weighted sum. I used MAI with $\alpha = -3$. Note that the previous kernel integration methods do not have a mapping rule (such as [21]) for new data points, while the MI algorithm does. One problem in the proposed approach is that the integrated kernel matrix might not be positive semidefinite, while the linearly integrated kernel matrix is always positive semidefinite as long as each kernel is positive semidefinite. Even though we use kernel Isomap which modifies the integrated kernel so that it is positive semidefinite, more research is needed when kernels are integrated.

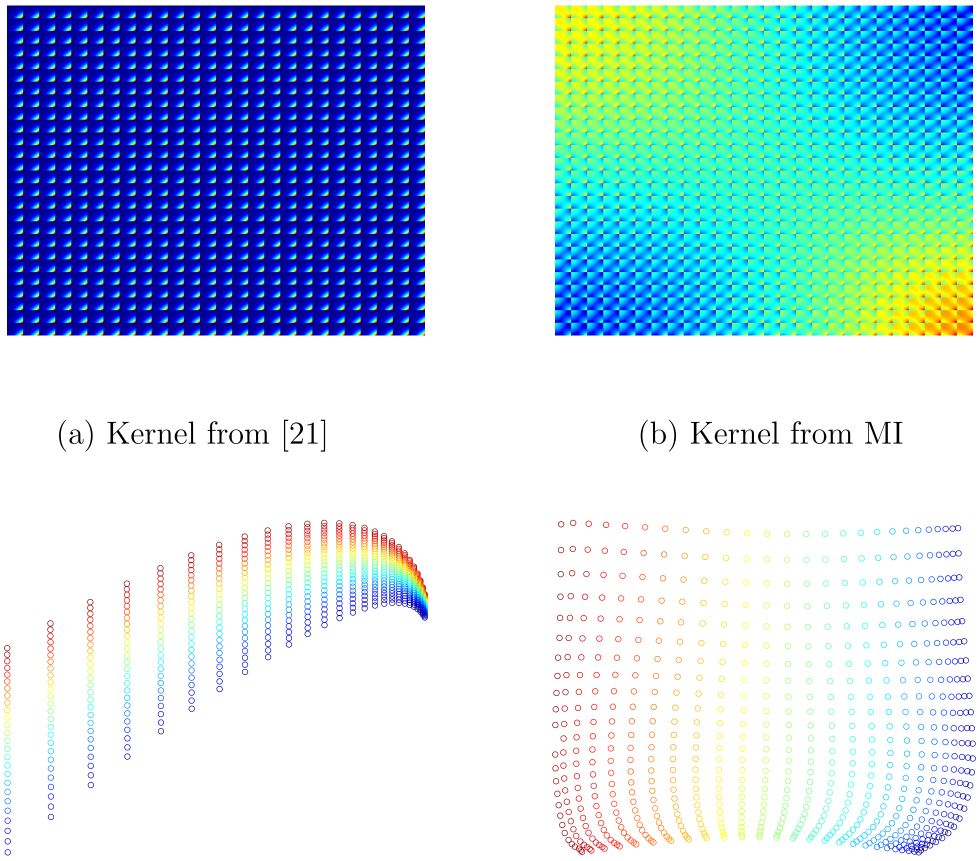
In sum, I showed how to apply MI to kernel integration. Also, I showed, with a synthetic data set, MI's performance advantage compared to kernel fusion method



(a) (b)

(c) (d)

Fig. 60. Projected spaces and kernel matrices individually. (a-b) Two kernel matrices from polynomial and exponential kernel functions and (c-d) the projected spaces through the two different kernel matrices for different measurements of the data set.



(a) Kernel from [21]

(b) Kernel from MI

(c) Map from [21]

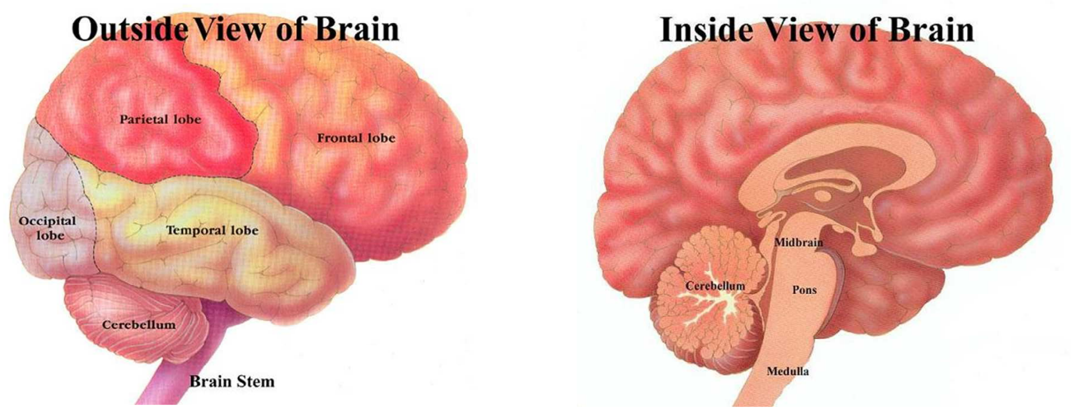
(d) Map from MI

Fig. 61. Projected spaces and integrated kernel matrices. Two integrated kernel matrices (a) by a linear sum as in [21] and (b) by MI. Integrated spaces are shown in (c) and (d) by the linear sum method and by MI (MAI with $\alpha = -3$). (d) is much closer to the true map (shown in Fig. 58 (a)) than (c) is. Also, (b) is closer to the true kernel matrix (shown in Fig. 58 (b))than (a) is.

based on a linear weighted sum [21].

B. Sensorimotor Integration

MI can be applied as a computational model for sensorimotor integration. In this section, I briefly review sensorimotor integration and then show that MI can be used as a good framework for sensorimotor integration.



(a) Outside view (b) Sagittal plane

Fig. 62. The cerebellum in the brain. (a) The cerebellum is located in the inferior posterior area in the brain, directly dorsal to the pons, and inferior to the occipital lobe. Motor coordination is a key function of the cerebellum. Adapted from http://www.omsusa.org/.

1. What is Sensorimotor Integration?

Sensorimotor integration is a mechanism that processes sensory information to guide actions and controls motor commands to manipulate sensory information. It has long been known that the cerebellum is the organ in the brain that acts as a motor coordinator [87, 88, 89, 90] and its absence produces spatial and temporal dysmetria [91]. The cerebellum allows the person to perform rapid alternating movements, speak clearly, and walk tandem [1]. See Fig. 62 for the location of the cerebellum in the brain.

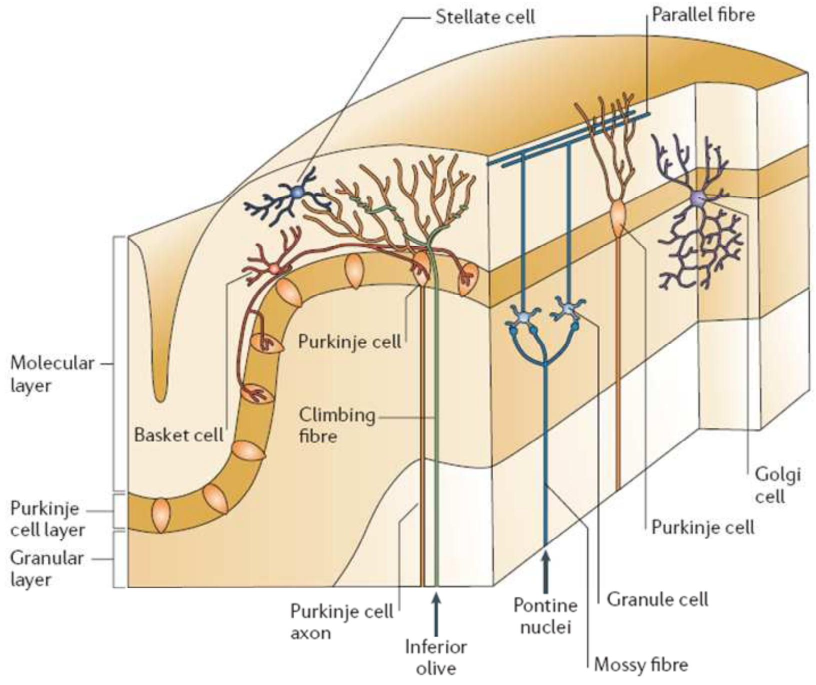


Fig. 63. Diagram of cerebellar cortex. According to Marr, the Purkinje cells are trained to connect olivary cells (motor-related) via climbing fibers and the context via mossy fibers-granule cells. After training, contextual information alone is enough to cause the next action [88]. Adapted from [92].

In the cerebellum, Purkinje cells are the primary integrative neurons and provide the only output of the cerebellar cortex. See Fig. 63 for the Purkinje cells in the cerebellar cortex. Each olivary cell responds to a cerebral instruction for an action. The inferior olivary cells and the cerebellar Purkinje cells have a special one-to-one relationship. Whenever an olivary cell fires, it sends an impulse to its corresponding Purkinje cell. This Purkinje cell is also exposed to the context[2] in which its olivary

[1]It is also known that the cerebellum takes a role in other cognitive functions like abstract reasoning, prosody and use of correct grammar [90].

[2]In Marr's paper, the exact meaning of 'context' is unclear, but we can assume that it means the information contained in the circumstantial sensory condition during which the Purkinje cells fired.

cell is fired. During training of an action with the corresponding context via mossy fibers and granule cells, each Purkinje cell can learn to recognize such contexts. After training, occurrence of the context alone is enough to fire the Purkinje cell, which then causes the next action. The action thus progresses as it did during training [88]. In Marr's model [88], the Purkinje cells work as classifiers and the granule cells are feature extractors.

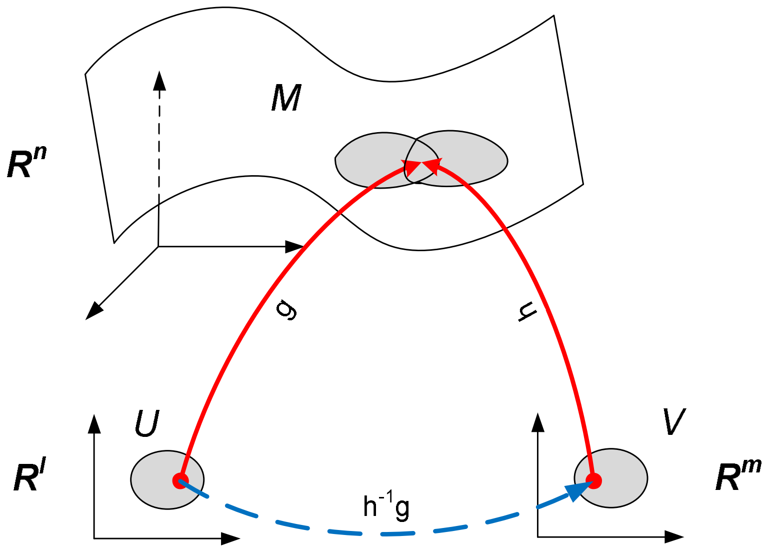


Fig. 64. Coordinate transformation. One manifold M can be represented by different coordinate systems U and V. In order to compare different representations, they should be represented by the same coordinate system.

One problem in previous formulations of sensorimotor integration is that the sensory information to the cerebellum cannot be directly understood since the sensory information and the motor information are encoded differently in the cerebellum [92, 68, 93]. For these two different types of information to be understood to each other, they can be converted into a common representation (or an integrated manifold) or the sensory information can be converted onto the motor manifold, which is referred to as coordinate transformation [93]. The mapping in the cerebellum can be understood as coordinate transformation and this view has been experimentally supported [94, 95]. Fig. 64 shows a diagram of coordinate transformation with two different coordinate patches (or maps). Given two coordinate patches, g and h, in order to compare the different representations, a common representation can be obtained in a different space like M or a transformation from U to V can be given by a function composition, $h^{-1}g$.

Some computational models for sensorimotor integration (or robot motor control) have been proposed [96, 97] including Bayesian theory [98], Kalman filter [93], recurrent network [99], forward model [100], input-output regression [101, 102], and tensor network theory [103, 104]. Generally, these models are based on an assumption that the motor commands and their consequences can be understood as input-output relationships. They do not consider the geometric structure of sensory and motor information that can be represented by manifolds.

There are some other models based on manifolds of sensory and motor information including generic forward-backward model [105], and spatio-temporal manifold learning [106, 107]. In addition to the manifold learning based models, differential geometry has been applied to understand dependency between sensory information and motor information on a manifold in a high dimensional space [108, 109]. However, these models do not consider the sensory and the motor information on separate manifolds or do not use the input-output relations clearly.

In this section, I develop a framework using MI as a computational model for sensorimotor integration. There are two approaches to use MI: (1) finding an integrated manifold of the sensory and the motor manifolds (subsection 2) which is a straightforward application of MI, and (2) transforming the sensory or motor information onto the other manifold, respectively (subsection 3).

2. Integration of Sensory and Motor Manifolds

In this subsection, I focus on finding a common representation as in [93]. That is, the goal of this approach is to obtain two mapping functions that project the two different representations onto the same space, such as g and h in Fig. 64.

To find a common representation, Ghahramani and his colleagues suggested maximizing mutual information between sensory information and motor information on the common representation while preserving topographic order to find the two different mapping functions, but without considering structural information inherent in the data set [93]. Instead, we can simply obtain two such mapping rules from MI, based on the structural information from the sensory and the motor maps. In MI, we learn two coordinate patches (or mapping rules g and h), from the individual manifolds to the integrated manifold, and obtain the integrated map on $M \in \mathbb{R}^n$. So, when we have two different manifolds representing an unknown true map, these mapping rules project the manifolds onto the same manifold while preserving the topological properties of the source. Experimental results in subsequent sections will demonstrate how manifold integration works.

Specifically, given the two dissimilarity matrices, $\boldsymbol{D}^{\mathrm{S}}$ and $\boldsymbol{D}^{\mathrm{M}}$, from the sensory and the motor information respectively, the integrated dissimilarity is given by

$$\tilde{D}_{ij} = f_\alpha^{-1}\left(w_s f_\alpha(D_{ij}^{\mathrm{S}}) + w_m f_\alpha(D_{ij}^{\mathrm{M}})\right). \tag{5.5}$$

where $\tilde{D}_{ij}, D_{ij}^{\mathrm{S}}$, and D_{ij}^{M} are distances between the ith and the jth points on the integrated manifold, the sensory manifold, and the motor map; and w_s and w_m are weights for the sensory and the motor information. Once we obtain the integrated dissimilarity matrix, kernel Isomap is applied to the matrix. The projection property is inherited from kernel Isomap as shown in the previous Chapter.

3. Transformation from Sensory to Motor Space

Here I focus on the coordinate transformation from one map (sensory manifold) to the other map (motor manifold), which can be obtained by $h^{-1}g$ as shown in Fig. 64. Calculating a general solution of h^{-1} might not be possible in manifold learning, since h is usually a function from a high dimensional space to a lower dimensional space. However, since the embedded manifold in M can be represented by the same number of dimensions as V, we can obtain the inverse of h locally based on each point's neighbors.

I develop two computational models for the coordinate transformation from the sensory information to the motor information. One does not use MI and the other one does so that the transformation is defined like $h^{-1}g$ through the integrated manifold. Note that these models can be applied to the transformation of motor information to sensory information without any modification.

a. Without Manifold Integration

Simply we can project sensory information directly onto the motor manifold using the local information on the sensory manifold. That is, assuming that the sensory and motor manifolds are locally very similar, the local information from the sensory manifold such as distances between a point to its neighbors can be reused in the motor manifold. The local information can be represented by weights of each point's neighbors. Let $\boldsymbol{x}_s$ be a point for the sensory information on the sensory manifold. As in locally linear embedding (LLE) [8], $\boldsymbol{x}_s$ can be approximately represented by neighbors on the sensory manifold considering that the manifold is locally Euclidean,

$$\boldsymbol{x}_s = \sum_{i \in \mathcal{I}^{\mathrm{S}}_{\boldsymbol{x}_s}} w^{\mathrm{S}}_i \boldsymbol{x}^{\mathrm{S}}_i, \tag{5.6}$$

where $\mathcal{I}^{\mathrm{S}}_{\boldsymbol{x}_s}$ is an index set of neighbors of $\boldsymbol{x}_s$ on the sensory manifold and w^{S}_i are weights for $\boldsymbol{x}^{\mathrm{S}}_i$ which are points on the sensory manifold. Then the projection of the sensory information onto the motor manifold can be approximately defined as

$$\boldsymbol{x}_m = \sum_{i \in \mathcal{I}^{\mathrm{S}}_{\boldsymbol{x}_s}} w^{\mathrm{S}}_i \boldsymbol{x}^{\mathrm{M}}_i. \tag{5.7}$$

Note that $\boldsymbol{x}_m$ in Eq. (5.7) uses the same weights but with the points on the motor manifold instead of on the sensory manifold. This method does not consider the fact that Riemannian metrics on two different manifolds might be different. However, if the samples are dense enough, the resulting projection will be very close to the true transformation.

To obtain the best weights, an objective function can be defined based on the error between the estimated distances of $\boldsymbol{x}_s$ to the neighbors and the actual distances, D^{S}_{sj}, given by

$$\mathcal{J}^{\mathrm{S}} = \sum_{j \in \mathcal{I}^{\mathrm{S}}_{\boldsymbol{x}_s}} \left((\boldsymbol{x}_s - \boldsymbol{x}^{\mathrm{S}}_j)^{\top} (\boldsymbol{x}_s - \boldsymbol{x}^{\mathrm{S}}_j) - (D^{\mathrm{S}}_{sj})^2 \right)^2. \tag{5.8}$$

Then our algorithm is simply given by gradient descent as follows.

$$\Delta w^{\mathrm{S}}_i = -\eta^{\mathrm{S}} \frac{\partial \mathcal{J}^{\mathrm{S}}}{\partial w^{\mathrm{S}}_i}, \tag{5.9}$$

where η^{S} is the learning rate and the gradient is given by

$$\frac{\partial \mathcal{J}^{\mathrm{S}}}{\partial w^{\mathrm{S}}_i} = 4 \sum_{j \in \mathcal{I}^{\mathrm{S}}_{\boldsymbol{x}_s}} \left((\boldsymbol{x}_s - \boldsymbol{x}^{\mathrm{S}}_j)^{\top} (\boldsymbol{x}_s - \boldsymbol{x}^{\mathrm{S}}_j) - (D^{\mathrm{S}}_{sj})^2 \right) (\boldsymbol{x}_s - \boldsymbol{x}^{\mathrm{S}}_j)^{\top} \boldsymbol{x}^{\mathrm{S}}_i. \tag{5.10}$$

After each learning step, the weights are normalized by the sum of all weights. This learning procedure takes time to obtain the optimized weights. For real-time transformation, we need a more elaborate method.

One other issue is that since gradient descent does not converge to the global minimum, we need good initial values for the weights. Intuitively, the weights should have an inverse relation with the distance, so it can be initialized by

$$w_i^{\mathrm{S}} = \frac{1}{Z_s^{\mathrm{S}}} \frac{1}{D_{si}^{\mathrm{S}}}, \text{ for } \ i \in \mathcal{I}_{\boldsymbol{x}_s}^{\mathrm{S}}, \tag{5.11}$$

where Z_s^{S} is a normalization term to make sure that $\sum_{j \in \mathcal{I}_{\boldsymbol{x}_s}^{\mathrm{S}}} w_j^{\mathrm{S}} = 1$. Actually these initial values are very close to the optimal solution[3].

b. Through Manifold Integration

We can formulate a new coordinate transformation method using MI. Once we have an integrated manifold of the two manifolds, when only sensory information is available, we can project it onto the integrated manifold using the projection property of MI. Let $\boldsymbol{x}_s$ be a point for the sensory information on the sensory manifold as before and $\tilde{\boldsymbol{x}}$ the projected point on the integrated manifold. Then we calculate the distances to the neighbors on both the sensory manifold and the integrated manifold, D_{sj}^{S} and $\tilde{D}_{sj}$, for all the neighbors on each manifold. Then, from Eq. (5.5), we can obtain the distances on the motor manifold as follows.

$$D_{sj}^{\mathrm{M}} = f_\alpha^{-1} \left(\frac{1}{w_m} \left[f_\alpha(\tilde{D}_{sj}) - w_s f_\alpha(D_{sj}^{\mathrm{S}}) \right] \right). \tag{5.12}$$

Using the fact that a manifold is locally Euclidean, given the distances from a point to its neighbors, the projection to the manifold is given by a weighted sum of the neighbors. So, given the sensory information, $\boldsymbol{x}_s$, the transformation onto the

[3]When the number of neighbors is 2, these values are actually the optimal solution. As the number of neighbors increases, these are not the optimal solution but they are still similar to the optimal solution.

motor manifold can be defined as

$$\boldsymbol{x}_m = \sum_{i \in \mathcal{I}^{\mathrm{I}}_{\tilde{\boldsymbol{x}}}} w_i^{\mathrm{M}} \boldsymbol{x}_i^{\mathrm{M}}, \tag{5.13}$$

where $\mathcal{I}^{\mathrm{I}}_{\tilde{\boldsymbol{x}}}$ is an index set of points that are neighbors to $\tilde{\boldsymbol{x}}$ on the integrated manifold. Here, the weights w_i^{M} should be non-negative and must sum up to 1. Note that we use neighbors on the integrated manifold instead of those on the sensory manifold, contrary to the previous section.

To obtain the best weights as before, an objective function can be defined by the error between the estimated distances to the neighbors and the true distances from Eq. (5.12) given by

$$\mathcal{J}^{\mathrm{M}} = \sum_{j \in \mathcal{I}^{\mathrm{I}}_{\tilde{\boldsymbol{x}}}} \left((\boldsymbol{x}_m - \boldsymbol{x}_j^{\mathrm{M}})^\top (\boldsymbol{x}_m - \boldsymbol{x}_j^{\mathrm{M}}) - (D_{mj}^{\mathrm{M}})^2 \right)^2 . \tag{5.14}$$

Then the algorithm simply utilizes gradient descent.

$$\Delta w_i^{\mathrm{M}} = -\eta^{\mathrm{M}} \frac{\partial \mathcal{J}^{\mathrm{M}}}{\partial w_i^{\mathrm{M}}}, \tag{5.15}$$

where η^{M} is the learning rate and the gradient is given by

$$\frac{\partial \mathcal{J}^{\mathrm{M}}}{\partial w_i^{\mathrm{M}}} = 4 \sum_{j \in \mathcal{I}^{\mathrm{I}}_{\tilde{\boldsymbol{x}}}} \left((\boldsymbol{x}_m - \boldsymbol{x}_j^{\mathrm{M}})^\top (\boldsymbol{x}_m - \boldsymbol{x}_j^{\mathrm{M}}) - (D_{mj}^{\mathrm{M}})^2 \right) (\boldsymbol{x}_m - \boldsymbol{x}_j^{\mathrm{M}})^\top \boldsymbol{x}_i^{\mathrm{M}}. \tag{5.16}$$

After each learning step, the weights are normalized as before.

Again, we need good initial values for the weights and they can be calculated as

$$w_i^{\mathrm{M}} = \frac{1}{Z_s^{\mathrm{M}}} \frac{1}{D_{si}^{\mathrm{M}}}, \ \text{for} \ \ i \in \mathcal{I}^{\mathrm{I}}_{\tilde{\boldsymbol{x}}}, \tag{5.17}$$

where Z_s^{M} is a normalization term to make sure $\sum_{j \in \mathcal{I}^{\mathrm{I}}_{\tilde{\boldsymbol{x}}}} w_j^{\mathrm{M}} = 1$. Note that again the neighbors are found on the integrated manifold, not on the sensory manifold.

This method is mathematically elegant, since the projection in Eq. (5.13) is based on the distances on the integrated manifold, which is equivalent to using the corresponding Riemannian metric on the integrated manifold. One more beneficial property of the proposed method is its coordinate invariance. So, its performance is constant no matter what kinds of coordinate system is used by sensory and motor information.

4. Simulation

In order to show the effectiveness of the method, I carried out experiments with a synthetic data set for integration on a common representation and coordinate transformation through integration. Simulated sensory and motor information were obtained using the simulation configuration shown in Fig. 65.

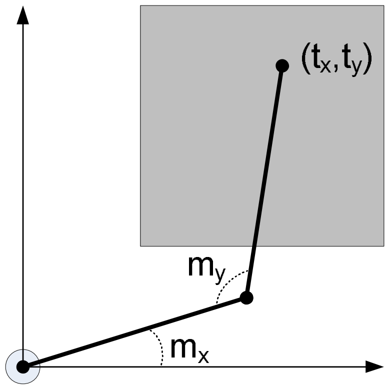


Fig. 65. Simulation of sensory and motor maps. The sensory information, mimicking a non-linear transformation (e.g., log-polar transform) in the visual system, is a distorted version of the true square map. The motor information is based on two angles of an articulated arm to reach locations within the true coordinate. The arm consisted of two sticks of the same length.

The sensory information, mimicking a non-linear transformation (e.g., log-polar transform) in the visual system, is a distorted version of the true square map. The motor information is based on two angles of an articulated arm to reach locations within the true coordinate. The arm consisted of two sticks of the same length. Given a point (t_x, t_y) in the true map with $t_x \in [0, 2]$ and $t_y \in [0, 2]$, the corresponding point in the sensory map (s_x, s_y) was given by

$$\begin{cases} s_x = t_x + 0.05, \\ s_y = \sqrt{t_y + 0.05}. \end{cases} \tag{5.18}$$

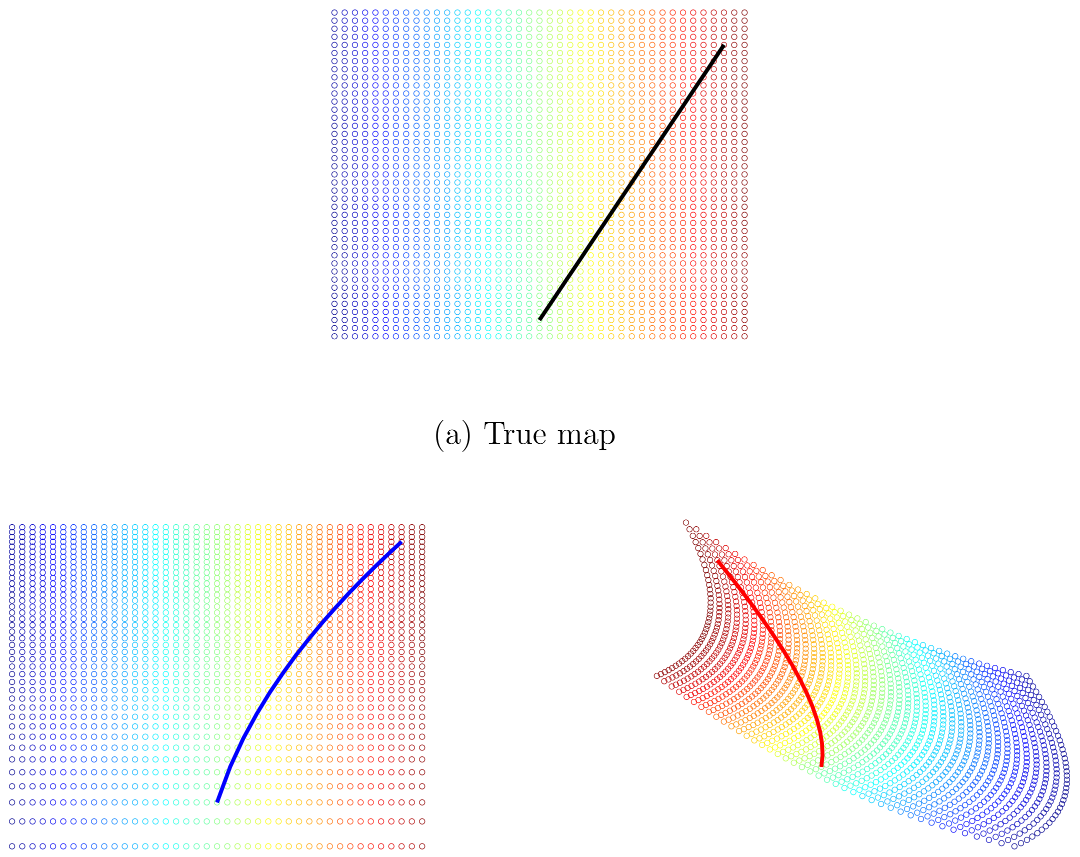
With the two sticks a_1 and a_2 pointing at (t_x, t_y) with the end of a_2, the corresponding point in the motor map (m_x, m_y) was calculated by

$$\begin{cases} m_x = \text{angle between } a_1 \text{ and the horizontal axis}, \\ m_y = \text{angle between } a_1 \text{ and } a_2, \end{cases} \tag{5.19}$$

where the angles are in radian. These distorted maps are shown in Fig. 66. In Fig. 66, the projections of the straight line from the true map in the two different maps result in curves and cannot be compared directly.

a. Integration of Sensory and Motor Manifolds

Fig. 67 shows how the integrated manifold can map and align the two different curves from the two different manifolds. For example, when we draw a straight line on the true map, we get two kinds of information at the same time, sensory and motor, as the two curves on the two different maps in Fig. 66. While we cannot directly compare the two curves on the two different maps, with MI we can project the two curves onto the integrated map and compare them as in Fig. 67. The blue curves are from the sensory space and the red curves from the motor space. In the integrated map, the two curves closely overlap, although they are not perfectly the same because the maps are not perfect. This error can be reduced if we increase the sample density in the map. We can compare the sensory and the motor information directly on the



(a) True map

(b) Sensory map

(c) Motor map

Fig. 66. Sensory and motor manifolds. (a) true manifold and a straight line (reference), (b) sensory manifold and the projection of the reference line, and (c) motor manifold and the projection of the line. The curves with the same color in the three manifolds represent the same location [110].

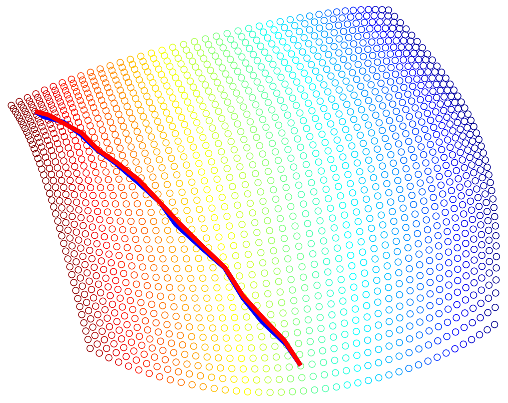


Fig. 67. Integration of the sensory and the motor maps using MI. An integrated map and the two projections of the reference line from the sensory and the motor space in Fig. 66, respectively. MAI_d with $\alpha = 3$ was used for MI.

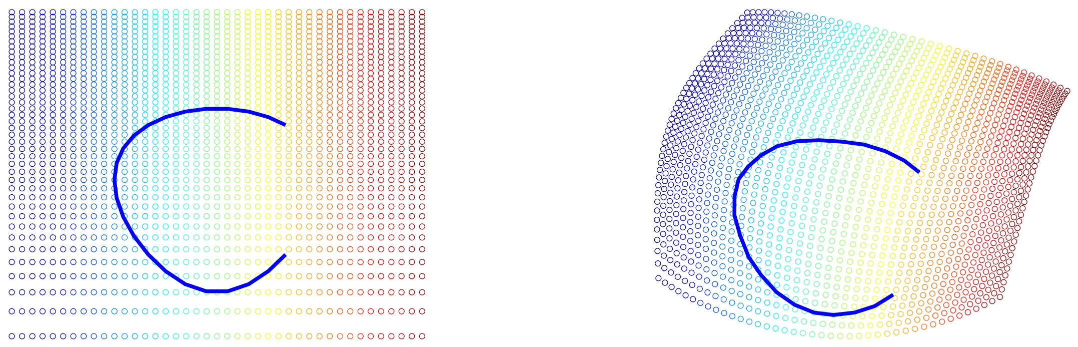
integrated manifold that gives a common representation.

b. Transformation from Sensory to Motor Space

To make the simulation more realistic, I increased the dimension of the motor information into three which is higher than the integrated manifold as well as the sensory manifold. That is, h is now a function from a higher dimensional space to a lower dimensional space and the inverse might not be possible. In this simulation, the third dimension for the motor manifold is simply added to Eq. (5.19) as

$$m_z = \sin(m_x), \tag{5.20}$$

where the embedded manifold of the motor information can still be represented by a two-dimensional coordinated system. After the integration as shown in the previous experiment, only the sensory information is given as a ‘C’-shaped curve in Fig. 68 (a). Note that the integrated manifold is obtained from two-dimensional sensory information and three dimensional motor information.



(a) Sensory manifold (Given) (b) Integrated manifold (Estimated)

Fig. 68. Sensory and motor transformation. (a) A 'C'-shaped curve on the sensory manifold and (b) the projection of the curve onto the integrated manifold.

When the sensory information is given, we can project it onto the integrated map as in Fig. 68 (b). Then based on the proposed algorithms we can project the sensory information on the motor manifold as in Fig. 69. That is, the sensory information is transformed into the form of motor information. In Fig. 69, the red curves are the transformed ones from the 'C'-shaped curve on the sensory manifold and the black dotted curves with cross marks are the true motor command which is unknown to the algorithm. Note that the red and the black curves almost completely overlap which shows that the algorithm works well. Here the data is assumed to be noise free.

Comparison of the two methods: As described in the previous section, there are two ways to perform coordinate transformation from the sensory information to the motor information: (1) without MI and (2) through MI. Since the difference in the projected result is almost invisible (as can be seen in Fig. 69) is almost invisible, the error between the estimated and the true points on the motor manifold was measured (Fig. 70), with different weights for the sensory and the motor information and different noise conditions in the sensory information. During the MI process, the number of neighbors was 9. For robust comparison, instead of one curve, 1,000 points

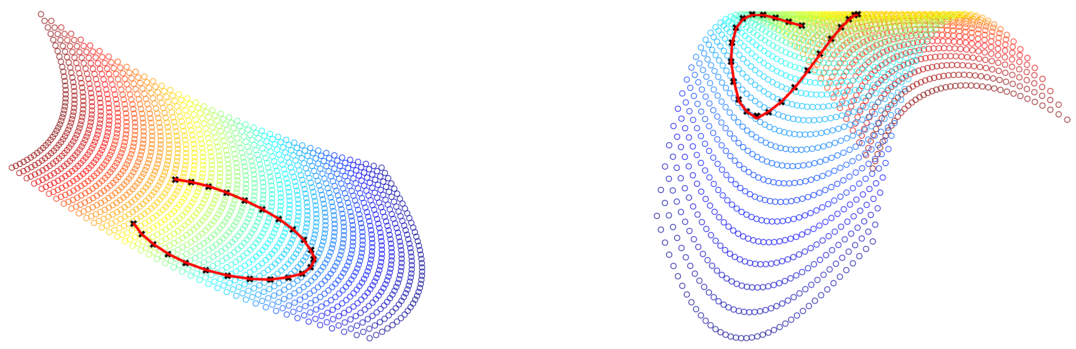
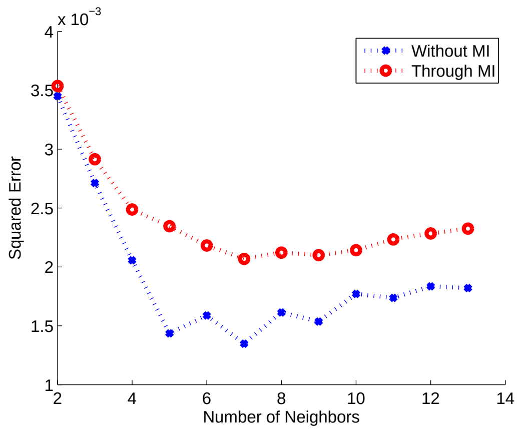

(a) First two dimensions (Estimated) (b) Last two dimensions (Estimated)

Fig. 69. The estimated and the true motor commands in the motor manifold, on different dimensions. In each of (a) and (b), the red curve is the estimated curve and the black curve marked with crosses is the true curve unknown to the algorithm. The circles with the same color in the manifolds represent the same location. Here the transformation is obtained using the integrated manifold.

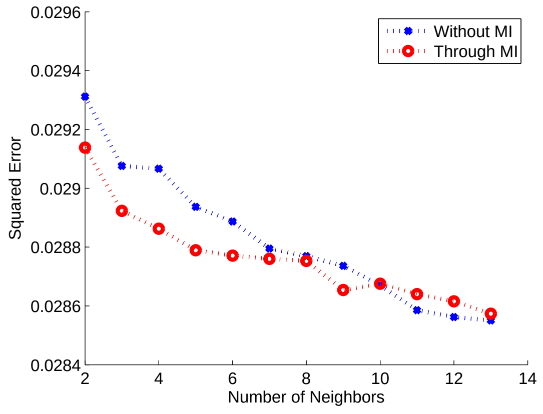
on the sensory manifold were randomly generated and the average of the errors is presented on the figures. For noisy sensory information, random noise (variance of 0.034) was added to each point [4]. Each case was tested with different neighbor sizes from 2 to 13.

Since the integrated manifold is obtained using local information on the sensory manifold, when there is no noise, simple transformation has slightly smaller error than the transformation through the integrated manifold as shown in Fig. 70 (a,c). However, when the sensory information is noisy, the transformation through the integrated manifold is slightly better especially when the neighbors are small as shown in Fig. 70 (b,d). That is because the integrated manifold is based on both sensory and motor information so that it is more stable than only the sensory manifold when

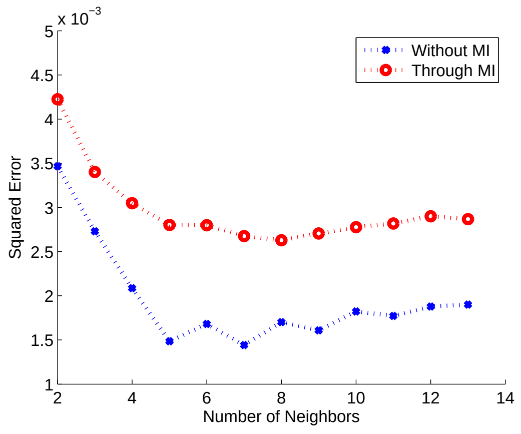
[4]The same integrated manifold was used with the noiseless sensory and motor information.



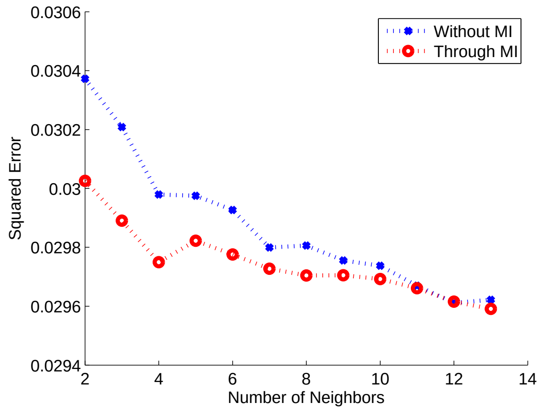
(a) Without noise, $w_s = 0.6$



(b) With noise, $w_s = 0.6$



(c) Without noise, $w_s = 0.4$



(d) With noise, $w_s = 0.4$

Fig. 70. Errors in the coordinate transformation in four cases. (a,c) without noise on the sensory information, (b,d) with noise on the sensor information, (a,b) with the weight 0.6 on the sensory side, and (c,d) with the weight 0.6 on the motor side.

sensory information is contaminated. In Fig. 70 (a,c), the error increases again after around 9 which is the neighbor size for manifold integration, since more neighbors break the assumption that the manifold is locally Euclidean. Also, the blue curves are not affected by different weights since the transformation is not related to the integrated manifold, while the red curves different with different weights. Note that the differences in error are very small because the individual and integrated manifolds are well distributed without noise.

As a computational model for sensorimotor integration, I formulated a manifold integration approach equipped with coordinate transformation from one manifold to the other manifold. Compared to other models such as tensor network theory, the method described above is easy to understand and implement. Also, the proposed method is mathematically elegant. One more beneficial property of the proposed method is its coordinate invariance. Although I used synthetic data sets, the results suggest that manifold integration can be used for sensorimotor integration. Also, the proposed approach can be applied to any other task that needs a coordinate transformation.

C. Summary

I applied the MI approach to two advanced integration tasks: kernel integration and sensorimotor integration, with discussions of MI compared to select previous methods. Although I used synthetic data sets for both tasks, the results are robust and suggest that MI can be used as a good framework for those tasks.

CHAPTER VI

DISCUSSION AND FUTURE WORK

In this Chapter, I will focus my discussion on manifold integration, its merits, constraints, limits and future direction.

A. Main Contributions

The main contribution of this dissertation is the new framework I developed to utilize both structural information and multiple measurements in the data set. The three algorithms I developed integrate manifold learning (for the utilization of structural information to find invariant properties) and data integration (for multiple measurements), showing promising results in a wide range of applications.

In KODA, the contribution lies in the nonlinear integration of multiple measurements with maximization of the separability between different classes. KODA is different from KPCA since KODA is a supervised method and also distinguished from KFD since KODA defines a noise term instead of the within-scatter matrix in KFD. The noise term is defined as the distance between points that are supposed to have the same invariant features.

In RAMS, the contribution is that RAMS is the first nonlinear manifold integration algorithm that integrates geometrically separate multiple manifolds. Contrary to KODA which defines distances between points on different manifolds, RAMS integrates the structural information statistically (not geometrically) based on the transition matrices of all manifolds. So, RAMS finds invariant features from multiple measurements as long as the measurements satisfy the constraints in the manifold integration concept.

For MAI, the main contribution is in the use of a general data integration method,

α-integration, thus providing a general manifold integration method that encompasses previous approaches as a special case. This generalized method can overcome the independence assumption in RAMS using different α values which determine the relations of multiple measurements. So, MAI integrates properly multiple measurements and extracts invariant features, according to the relation of the measurements.

I showed that manifold integration is effective in analyzing multimodal data sets with various experiments and can be used as a new framework for other integration tasks (e.g., kernel integration and sensorimotor integration). So, another important contribution of this dissertation is that this research can help achieve better performance in diverse applications where multiple measurements are available, and can open new directions in manifold learning research as well as in data integration research.

B. The Constraints of Manifold Integration

First, the constraints for manifold integration described in Table IV in Chapter IV are just for the current version of manifold integration. Manifold integration can be generalized by loosing the constraints.

The first constraint is inherited from manifold learning. Most manifold learning algorithms have the same constraint and it is hard to avoid (See [111, 10] for some discussion about this.).

The second constraint can be alleviated. As shown in Fig. 64, an integrated manifold can be drawn based on the intersection of the manifolds. That is, instead of using all the data points, as long as there are some data points that are measured on all (or some) of the manifolds, we can find out an integrated manifold. Then a remaining issue is how to expand the integrated manifold with the other data points that are not on all (or some) of the manifolds. This can be really useful for some practical situations. For example, when multi-robots are building one integrated map, each robot can navigate and build its own map where the maps are only partially overlapping [112]. Then based on the common intersection of the maps, the integrated map can be completed including exclusive (non-common) maps from all the robots.

As mentioned in Chapter IV, in many cases like face and voice data set for personal identification, the third constraint could cause manifold integration to fail. However, even if multiple manifolds are topologically different, locally they can be the same. So, instead of considering integrated manifold as a whole, we can find an integrated manifold as a connected manifold of sub-manifolds each of which is an integrated manifold of topologically isomorphic sub-manifolds of the multiple manifolds. In this sense, this constraint is somehow interwoven with the second constraint.

C. Issues in the Manifold Integration Algorithms

Besides the concept and constraints of manifold integration, the algorithms and the applications presented in this dissertation have some issues to discuss or to be solved in the future.

In MAI, the α value and the weight vector $\boldsymbol{w}$ are given in advance and fixed. As discussed in Chapter III, it is desirable to be able to learn the parameters in MAI. If some target values are given for integrated distances, we can apply the learning algorithms presented in Chapter III to MAI directly. But then one question arises: "how often is it that, in practice we have target values for integrated distances?" That is, when we have multiple distance matrices for a true data set, can we have some true distances for the data set? This is always an issue in manifold integration even when manifold integration is applied to other integration tasks like kernel integration and sensorimotor integration.

One remaining issue on kernel integration is how to apply the unsupervised version of manifold integration to the supervised cases. For example, in KFD, a supervised kernel machine, there are two kernels for one measurement: one is from the 'between scatter' matrix, $\boldsymbol{K}_{\mathrm{B}}$, and the other from the 'within scatter' matrix, $\boldsymbol{K}_{\mathrm{W}}$. Then are we supposed to integrate separately the kernel matrices from the 'between scatter' matrices of multiple measurements and the kernel matrices from the 'within scatter' matrices? Or do we have to integrate the final form of kernel matrices in the generalized eigenvalue form like $\boldsymbol{K}_{\mathrm{W}}^{-1}\boldsymbol{K}_{\mathrm{B}}$ [113]? Or else, do we have to figure out a new way? We need more research on this issue.

Since I focused on the mathematical formulation of manifold integration, I did not use any real-world data sets for kernel integration and sensorimotor integration. However, based on the experiments with synthetic data sets, I expect the proposed approaches to work well with real world data sets. It would be an interesting direction for future work to apply manifold integration to kernel integration and sensorimotor integration with real-world data sets.

In sensorimotor integration, for one invariant object in the external world, tensor network theory uses two non-orthogonal frames of reference for sensory and motor information. However, while these non-orthogonal frames in tensor network theory are linear, manifold integration implicitly has two nonlinear frames for the two references. Actually, manifold integration does not need any coordinate system for sensory and motor information, and it works based on the relative geometric relations. Any nonlinear form of manifold can be represented by this relative geometric relations, thus manifold-based integration is more general and flexible.

D. Future of Manifold Integration

Even though I showed many applications of manifold integration, it can be applied to many other areas with additional work discussed above.

Due to the rapid technological development, data can be measured in many different ways for the same task and multiple measurements need to be integrated. For example, speech recognition systems using both sound and video (of the lips) can outperform audio-only recognition systems [114, 115, 116]. The McGurk effect shows that speech perception in our brain is multimodal combining audio and visual information interactively [117]. It would be interesting to model the McGurk effect with manifold integration. Also, EEG, MEG and fMRI data from a single subject can be recorded at the same time when he or she performs a specific task. In such cases, multimodal data analysis is required for reliable results. Since each modality of such data sets can be analyzed on a manifold, manifold integration can be helpful in obtaining more reliable analysis from the multimodal data sets. Basically manifold integration can be applied to any multimodal data set as long as they satisfy the constraints of manifold integration.

Not only for multimodal data sets, manifold integration can also be used as a coordinate transformation tool as in sensorimotor integration. The application of manifold integration to sensorimotor integration can be easily revised and applied to some other areas like robotics. It can be applied to more than just a transformation between an eye and an arm. For example, for automatic driving, camera information and internal/external context like tire pressure or brake condition can be combined and transformed based on manifold integration into motor information to control the wheels of the car.

E. Summary

In sum, I discussed several issues in manifold integration. Some of these are interesting directions for future work. The issues mentioned above may or may not be so easy to resolve. However, any progress in them will be helpful for better multimodal data analysis and for practical applications of manifold integration to other integration tasks. In addition, I presented some future direction for manifold integration research. Although they cannot be achieved with the current manifold integration techniques, they can be in the future, with more effort.

CHAPTER VII

CONCLUSION

Manifold learning and data integration have been important and popular research topics. However, they have not been considered together in a single framework to deal with multimodal data sets more effectively than when each approach is used independently from each other. In prior work, I studied separately manifold learning and data integration, and developed algorithms for each. For manifold learning, I derived an advanced algorithm called kernel Isomap that possesses a desirable projection property, and for data integration, I extended the application of α-integration and developed learning rules for parameters that previously had to be set manually. For both algorithms, I demonstrated the effectiveness using synthetic and real-world tasks.

To analyze multimodal data sets, I formulated a new concept, manifold integration, a data integration method using the structure of data expressed by multiple manifolds. Manifold integration combines manifold learning and data integration to integrate information from multiple manifolds. So, contrary to manifold learning or data integration as separate methods, my method uses both statistical relation between the two (or more) sets and the geometric relation between data points in each set.

For manifold integration, I derived three algorithms: KODA, RAMS and MAI. They depended on different assumptions so that they work with different kinds of data set. However, all of them can find a nonlinear mapping rule and can project new points on the projected space constructed from the training data set. Also, manifold integration is coordinate invariant since it does not use any coordinate system except in the final projection space. The three algorithms were tested with synthetic and real world data sets, showing robust results.

Manifold integration provides an effective framework for advanced integration tasks such as kernel integration and sensorimotor integration as well as for integrating general multimodal data sets on multiple manifolds. Although I used synthetic data sets for both application, they show the possibility that manifold integration can be used successfully for those tasks.

Finally, I expect that my research on manifold integration can catalyze the two separate research areas of manifold learning and data integration.

REFERENCES

[1] H. S. Seung and D. D. Lee, “The manifold ways of perception,” *Science*, vol. 290, pp. 2268–2269, 2000.

[2] I. T. Jolliffe, *Principal Component Analysis.* New York, NY: Springer-Verlag, 1986.

[3] T. Cox and M. Cox, *Multidimensional Scaling*, 2nd ed. London: Chapman & Hall, 2001.

[4] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* New York, NY: John Wiley & Sons, 2001.

[5] R. S. Millman and G. D. Parker, *Elements of Differential Geometry.* Englewood Cliffs, NJ: Prentice-Hall, 1977.

[6] M. Spivak, *A Comprehensive Introduction to Differential Geometry*, 3rd ed. Houston, TX: Publish or Perish, 1999, vol. 1.

[7] J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, pp. 2319–2323, 2000.

[8] S. T. Roweis and L. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, pp. 2323–2326, 2000.

[9] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural Computation*, vol. 15, pp. 1373–1396, 2003.

[10] H. Choi and S. Choi, “Robust kernel Isomap,” *Pattern Recognition*, vol. 40, no. 3, pp. 853–862, Mar. 2007.

[11] ——, “Kernel Isomap on noisy manifold,” in *Proc. Int’l Conf. Development and Learning*, Osaka, Japan, 2005, pp. 208–213.

[12] H. Choi, B. Paulson, and T. Hammond, “Gesture recognition based on manifold learning,” in *Proc. Int’l Workshop on Structural and Syntactic Pattern Recognition.* Orlando, FL: Springer-Verlag, 2008, (LNCS 5342 pp. 247-256).

[13] B. Schölkopf, A. J. Smola, and K. R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[14] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge: Cambridge University Press, 2000.

[15] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, “A kernel view of the dimensionality reduction of manifolds,” in *Proc. Int’l Conf. Machine Learning*, Banff, Canada, 2004, pp. 369–376.

[16] D. L. Hall and J. Llinas, “An introduction to multisensor data fusion,” *Proceedings of the IEEE*, vol. 85, no. 1, pp. 369–376, 1997.

[17] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 1988.

[18] A. P. Dempster, “Upper and lower probabilities induced by a multivalued mapping,” *The annals of Statistics*, vol. 28, pp. 325–339, 1967.

[19] G. Shafer, *A Mathematical Theory of Evidence.* Princeton, NJ: Princeton University Press, 1976.

[20] C. M. Bishop, *Neural Networks for Pattern Recognition.* New York, NY: Oxford University Press, 1995.

[21] G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble, "Kernel-based data fusion and its application to protein function prediction in yeast," in *Proc. Pacific Symposium on Biocomputing (PSB)*, vol. 9, Big Island, HI, 2004, pp. 300–311.

[22] C.-C. Ho, K. MacDorman, and Z. A. D. Pramono, "Human emotion and the uncanny valley: A GLM, MDS, and Isomap analysis of robot video ratings," in *Proc. of the Third ACM/IEEE Int'l Conf. on Human-Robot Interaction*, Amsterdam, the Netherlands, Mar. 2008, pp. 169–176.

[23] H. Choi and T. Hammond, "Sketch recognition based on manifold learning," in *Proc. Association for the Advancement of Artificial Intelligent (AAAI)*, vol. 3, Chicago, IL, 2008, pp. 1786–1787.

[24] R.-J. Gu and W.-B. Xu, "Face recognition based on supervised kernel Isomap," in *Computational Intelligence and Security: Int'l Conf., CIS 2006*, 2006, pp. 674–677.

[25] ——, "Weighted kernel Isomap for data visualization and pattern classification," in *Computational Intelligence and Security: Int'l Conf., CIS 2006*, 2006, pp. 1050–1057.

[26] C. K. I. Williams, "On a connection between kernel PCA and metric multidimensional scaling," *Machine Learning*, vol. 46, pp. 11–19, 2002.

[27] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K. Müller, "Fisher discriminant analysis with kernels," in *Proc. IEEE Neural Networks for Signal Processing Workshop*, 1999, pp. 41–48.

[28] H. Choi and S. Choi, "Kernel Isomap," *Electronics Letters*, vol. 40, no. 25, pp. 1612–1613, Dec. 2004.

[29] L. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, Jun. 2003.

[30] G. Hinton and S. Roweis, "Stochastic neighbor embedding," in *Advances in Neural Information Processing Systems*, vol. 15. Cambridge, MA: MIT Press, 2003, pp. 857–864.

[31] X. He and P. Niyogi, "Locality preserving projections," in *Advances in Neural Information Processing Systems*, vol. 16. Cambridge, MA: MIT Press, 2004.

[32] B. Schölkopf and A. J. Smola, *Learning with Kernels.* Cambridge, MA: MIT Press, 2002.

[33] D. L. Hall and J. Llinas, *Handbook of Multisensor Data Fusion*, 2nd ed. Boca Raton, FL: CRC Press, 2001.

[34] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transaction of the ASMEJournal of Basic Engineering*, pp. 35–45, Mar. 1960.

[35] S. Amari, "Integration of stochastic models by minimizing α-divergence," *Neural Computation*, vol. 19, pp. 2780–2796, 2007.

[36] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–81, 1991.

[37] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

[38] S. Amari and H. Nagaoka, *Methods of Information Geometry.* Providence, RI: American Mathematical Society, 2000.

[39] G. H. Hardy, J. E. Littlewood, and G. Polya, *Inequalities*, 2nd ed. Cambridge: Cambridge University Press, 1994.

[40] S. M. Ali and S. D. Silvey, “A general class of coefficients of divergence of one distribution from another,” *Journal of the Royal Statistical Society B*, vol. 28, pp. 131–142, 1966.

[41] I. Csiszár, “Information measures: A critical survey,” in *Trans. 7th Prague Conference on Information Theory*, vol. A, 1974, pp. 73–86.

[42] B. Bollobás, *Modern Graph Theory.* New York, NY: Springer, 1998.

[43] F. Cailliez, “The analytical solution of the additive constant problem,” *Psychometrika*, vol. 48, no. 2, pp. 305–308, 1983.

[44] D. Han, H. Choi, C. Park, and Y. Choe, “Fast and accurate retinal vasculature tracing and kernel-Isomap-based feature selection,” in *Proc. Int’l Joint Conf. Neural Networks*, Atlanta, Geogia, Jun. 2009, pp. 1160–1167.

[45] R. Duraiswami and V. C. Raykar, “The manifolds of spatial hearing,” in *Proc. IEEE Int’l Conf. Acoustics, Speech, and Signal Processing*, 2005, pp. 285–288.

[46] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, “The CIPIC HRTF database,” in *Proc. 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001, pp. 99–102.

[47] S. Hettich, C. L. Blake, and C. J. Merz, “UCI repository of machine learning databases,” 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[48] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *Advances in Neural Information Processing Systems*, vol. 14. Cambridge, MA: MIT Press, 2002, pp. 585–591.

[49] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from examples,” Department of Computer Science, University of Chicago, Tech. Rep. TR-2004-06, 2004.

[50] D. Rubine, “Specifying gestures by example,” *Computer Graphics*, vol. 25, no. 4, pp. 329–337, Jul. 1991.

[51] J. Wobbrock, A. Wilson, and Y. Li, “Gestures without libraries, toolkits, or training: A $1 recognizer for user interface prototypes,” in *Proc. of the 20th Annual ACM Symposium on User Interface Software and Technology*, Newport, RI, 2007, pp. 159–168.

[52] K. Sentz and S. Ferson, “Combination of evidence in Dempster-Shafer theory,” Albuquerque, NM, Tech. Rep. Sandia report SAND2002-0835, 2002.

[53] N. Friedman, D. Geiger, and M. Goldszmidt, “Bayesian network classifiers,” *Machine Learning*, pp. 131–163, 1997.

[54] C. K. Murphy, “Combining belief functions when evidence conflicts,” *Decision Support Systems*, vol. 29, pp. 1–9, 2000.

[55] D. Yong, S. WenKang, Z. ZhenFu, and L. Qi, “Combining belief functions based on distance of evidence,” *Decision Support Systems*, vol. 38, pp. 489–493, 2004.

[56] H. Choi, A. Katake, S. Choi, Y. Kang, and Y. Choe, "Probabilistic combination of multiple evidence," in *Proc. Int'l Conf. Neural Information Processing*, Bangkok, Thailand, 2009, (Part I, LNCS 5863, pp. 302–311).

[57] G. Xu, W. Tian, L. Qian, and X. Zhang, "A novel conflict reassignment method based on grey relational analysis (GRA)," *Pattern Recognition Letters*, vol. 28, pp. 2080–2087, 2007.

[58] H. Choi, A. Katake, S. Choi, and Y. Choe, "Alpha-integration of multiple evidence," in *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, Dallas, TX, 2010, pp. 2210–2213.

[59] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eigenics*, vol. 7, pp. 179–188, 1936.

[60] S. Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision*, 2006, pp. 531–542, (Part IV, LNCS 3954).

[61] S. J. D. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *IEEE 11th International Conference on Computer Vision*, Rio de Janeiro, Brazil, Oct. 2007, pp. 1–8.

[62] D. J. Bartholomew, *Latent Variable Models and Factor Analysis.* London: Charles Griffin & Co. Ltd., 1987.

[63] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society B*, vol. 61, no. 3, pp. 611–622, 1999.

[64] T. Minka, "Divergence measures and message passing," Microsoft Research, Tech. Rep. MSR-TR-2005-173, 2005.

[65] A. Cichocki, H. Lee, Y.-D. Kim, and S. Choi, “Nonnegative matrix factorization with α-divergence,” *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1433–1440, Jul. 2008.

[66] Y. D. Kim, A. Cichocki, and S. Choi, “Nonnegative Tucker decomposition with alpha-divergence,” in *Proc. IEEE Int’l Conf. Acoustics, Speech, and Signal Processing*, Las Vegas, NV, 2008, pp. 1829–1832.

[67] H. Choi, S. Choi, A. Katake, and Y. Choe, “Learning α-integration with partially-labeled data,” in *Proc. IEEE Int’l Conf. Acoustics, Speech, and Signal Processing*, Dallas, TX, 2010, pp. 2058–2061.

[68] A. J. Pellionisz, “Brain theory: Connecting neurobiology to robotics tensor analaysis: Utilizing intrinsic coordinates to describe, understnd and engineer functional geometries of intelligent organisms,” *Journal of Theoretical Neurobiology*, vol. 2, pp. 185–211, 1983.

[69] J. Laub and K. R. Müller, “Feature discovery in non-metric pairwise data,” *Journal of Machine Learning Research*, vol. 5, pp. 801–818, 2004.

[70] C. Ong, X. Mary, S. Canu, and A. Smola, “Learning with non-positive kernels,” in *Proc. Int’l Conf. Machine Learning*, Banff, Canada, 2004, pp. 639–646.

[71] H. Abdi, D. Valentin, A. J. O’Toole, and B. Edelman, “DISTATIS: The analysis of multiple distance matrices,” in *Proc. Int’l Conf. Computer Vision and Pattern Recognition*, San Diego, CA, 2005, pp. 42–47.

[72] K. I. Diamantaras and S. Y. Kung, *Principal Component Neural Networks: Theory and Applications.* New York, NY: John Wiley & Sons, 1996.

[73] N. Malayath, H. Hermansky, and A. Kain, "Towards decomposing the sources of variability in speech," in *Proc. EUROSPEECH*, Rhodes, Greece, 1997, pp. 497–500.

[74] H. Choi, R. Gutierrez-Osuna, S. Choi, and Y. Choe, "Kernel oriented discriminant analysis for speaker-independent phoneme spaces," in *Proc. Int'l Conf. Pattern Recognition*, Tampa, FL, 2008, pp. 1–4.

[75] J. Kominek and A. W. Black, "CMU ARCTIC databases for speech synthesis," 2003. [Online]. Available: http://www.festvox.org/cmu_arctic/

[76] K. Fukunaga, *An Introduction to Statistical Pattern Recognition.* New York, NY: Academic Press, 1990.

[77] H. Choi, S. Choi, and Y. Choe, "Manifold integration with Markov random walks," in *Proc. Association for the Advancement of Artificial Intelligent (AAAI)*, vol. 1, Chicago, IL, 2008, pp. 424–429.

[78] M. Szummer and T. Jaakkola, "Partially labeled classification with Markov random walks," in *Advances in Neural Information Processing Systems*, vol. 14. Cambridge, MA: MIT Press, 2002, pp. 945–952.

[79] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proc. Int'l Conf. Machine Learning*, 2002, pp. 315–322.

[80] D. Zhou and C. Burges, "Spectral clustering and transductive learning with multiple views," in *Proc. Int'l Conf. Machine Learning*, 2007, pp. 1159–1166.

[81] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 169–184.

[82] Y. Bengio, O. Delalleau, N. Le Roux, J. F. Paiement, and M. Ouimet, "Learning eigenfunctions links spectral embedding and kernel PCA," *Neural Computation*, vol. 16, pp. 2197–2219, 2004.

[83] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, spectral clustering and normalized cuts," in *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, Seattle, WA, 2004, pp. 551–556.

[84] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Trans. Neural Networks*, vol. 13, no. 3, pp. 780–784, 2002.

[85] Y. Bai, C. Park, and Y. Choe, "Relative advantage of touch over vision in the exploration of texture," in *Proc. Int'l Conf. Pattern Recognition*, Tempa, Florida, 2008, pp. 1–4.

[86] C. Park, H. Choi, and Y. Choe, "Self-organization of tactile receptive fields: Exploring their textural origin and their representational properties," in *7th International Workshop on Self-Organizing Maps*, St. Augustine, FL, 2009, pp. 228–236.

[87] P. Flourens, *Experimental Researches on the Properties and the Functions of the Nervous System in Vertebrate Animals*, 2nd ed. Paris: Bailliere, 1824.

[88] D. Marr, "A theory of cerebellar cortex," *Journal of Physiology*, vol. 202, pp. 437–470, 1969.

[89] A. J. Pellionisz, "David Marr: A theory of the cerebellar cortex," *Brain Theory*, pp. 253–257, 1986.

[90] E. J. Fine, C. C. Ionita, and L. Lohr, "The history of the development of the

cerebellar examination," *Seminars in Neurology*, vol. 22, no. 4, pp. 375–384, 2002.

[91] G. Holmes, "The cerebellum of man," *Brain*, vol. 62, pp. 1–30, 1939.

[92] N. Ramnani, "The primate cortico-cerebellar system: Anatomy and function," *Nature Reviews Neuroscience*, vol. 7, pp. 511–522, 2006.

[93] Z. Ghahramani, D. M. Wolpert, and M. I. Jordan, "Computational models of sensorimotor integration," *Science*, vol. 269, pp. 1880–1882, 1997.

[94] A. J. Pellionisz and R. Llinas, "Space-time representation in the brain. The cerebellum as a predictive space-time metric tensor," *Neuroscience*, vol. 7, no. 12, pp. 2949–2970, 1982.

[95] C. C. Gielen and E. J. van Zuylen, "Coordination of arm muscles during flexion and supination: application of the tensor analysis approach," *Neuroscience*, vol. 17, no. 3, pp. 527–539, 1986.

[96] E. Todorov, "Optimality principles in sensorimotor control," *Nature Neuroscience*, vol. 7, no. 9, 2004.

[97] M. I. Jordan and D. M. Wolpert, *The Cognitive Neuroscience.* Cambridge, MA: MIT Press, 1999, ch. Computational Motor Control.

[98] K. P. Körding and D. M. Wolpert, "Bayesian decision theory in sensorimotor control," *TRENDS in Cognitive Science*, vol. 10, no. 7, pp. 319–326, 2006.

[99] S. Denéve, J.-R. Duhamel, and A. Pouget, "Optimal sensorimotor integration in recurrent cortical networks: A neural implementation of Kalman filters," *Journal of Neuroscience*, vol. 27, no. 21, pp. 5744–5756, 2007.

[100] D. M. Wolpert and R. C. Miall, “Forward models for physiological motor control,” *Neural Network*, vol. 9, pp. 1265–1279, 1996.

[101] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning,” *Artificial Intelligence Review*, vol. 11, pp. 11–73, 1997.

[102] C. W. Wampler, “Manipulator inverse kinematics solution based on damped least-squares solutions,” *IEEE Trans. Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[103] A. J. Pellionisz, “Tensor network theory of the central nervous system,” in *Encyclopedia of Neuroscience*, G. Adelman, Ed. Boston, MA: Birkhauser, 1987, vol. II, pp. 1196–1198.

[104] A. J. Pellionisz and R. Llinas, “Tensor network theory of the metaorganization of functional geometries in the central nervous system,” *Neuroscience*, vol. 16, no. 2, pp. 245–273, 1985.

[105] M. Lopes and B. Damas, “A learning framework for generic sensory-motor maps,” in *IEEE/RSJ Int. Cof. on Intelligent Robots and Systems*, San Diego, CA, 2007, pp. 1533–1538.

[106] O. C. Jenkins, R. Bodenheimer, and R. Peters, “Manipulation manifolds: Explorations into uncovering manifolds in sensory-motor spaces,” in *Proc. Int’l Conf. Development and Learning*, Bloomington, IN, 2006.

[107] R. A. P. II and O. C. Jenkins, “Uncovering manifold structures in robonauts sensory-data state space,” in *IEEE-RAS International Conference on Humanoid Robotics*, Tsukuba, Japan, 2005, pp. 369–374.

[108] D. Philipona, J. K. O'Regan, and J. P. Nadal, "Is there something out there? Inferring space from sensorimotor dependencies," *Neural Computation*, vol. 15, pp. 2029–2050, 2003.

[109] D. Philipona, J. K. O'Regan, J. P. Nadal, and O. J. M. D. Coenen, "Perception of the structure of the physical world using unknown multimodal sensors and effectors," in *Advances in Neural Information Processing Systems*, S. T. L. Saul and B. Schölkopf, Eds., vol. 16. Cambridge, MA: MIT press, 2004, pp. 945–952.

[110] H. Choi, S. Choi, A. Katake, and Y. Choe, "Sensorimotor integration on manifolds," in *The Southwest Cognition Conference (Armadillo)*, Houston, TX, 2010, poster presentation.

[111] M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford, "The Isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, p. 7, Jan. 2002.

[112] A. Howard, G. S. Sukhatme, and M. J. Mataric, "Multirobot simultaneous localization and mapping using manifold representations," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1360–1369, 2006.

[113] G. Strang, *Linear Algebra and Its Applications*, 3rd ed. Orlando, FL: Harcourt Brace Jovanovich, Inc, 1988.

[114] D. G. Stork and M. E. Hennecke, *Speechreading by Humans and Machines: Models, Systems and Applications.* New York, NY: Springer-Verlag, 1996.

[115] J. R. Movellan and P. Mineiro, "Robust sensor fusion: Analysis and application to audio-visual speech recognition," *Machine Learning*, vol. 32, no. 2, pp. 85–100, Aug. 1998.

[116] X. Z. Zhang, R. M. Merserratt, and M. Clements, “Bimodal fusion in audio-visual speech recognition,” in *Proc. Int’l Conf. on Image Processing*, 2002, pp. I: 964–967.

[117] H. McGurk and J. MacDonald, “Hearing lips and seeing voices,” *Nature*, vol. 264, pp. 746–748, 1976.

VITA

Hee Youl Choi received his B.S. and M.S. degrees in computer science from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 2002 and 2005. As a computer programmer in OromInfo for 3 years, he developed several web service and server programs for digital libraries, such as database managements, search engines, inter-library-loan (ILL) service, image viewers, and book alert services. Also he worked in POSTECH as a researcher on machine learning for 6 months and had an internship in Starvision Technologies on pattern recognition for almost 4 months.

His master thesis was about independent component analysis (ICA) titled "Relative Trust-Region Learning for ICA" where he developed a new optimization algorithm for ICA. He received his Ph.D. in the Department of Computer Science and Engineering at Texas A&M University in 2010. His research interests include manifold integration, sensorimotor integration, active perception, computational neuroscience, manifold learning, kernel methods, differential geometry, independent component analysis, blind source separation, neural networks, pattern recognition, machine learning, camera calibration, image/signal processing, graphical models, and concepts learning.

Dr. Choi may be reached at 301 Harvey R. Bright Building, College Station, TX 77843-3112 or at heeyoul@gmail.com.

The typist for this dissertation was Hee Youl Choi.