

กรอบงานการเรียนรู้ที่มีผู้สอนแบบไม่เชิงเส้นสำหรับการเรียนรู้ฟังก์ชันระยะทาง



นาย รัฐฉัตร ฉัตรพัฒนศิริ

ศูนย์วิทยทรัพยากร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2552

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

A NON-LINEAR SEMI-SUPERVISED LEARNING FRAMEWORK
FOR DISTANCE METRIC LEARNING



Mr. Rattachat Chatpatanasiri

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering


Chulalongkorn University

Academic Year 2009

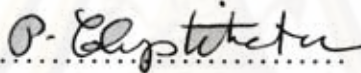
Copyright of Chulalongkorn University


Thesis Title A NON-LINEAR SEMI-SUPERVISED LEARNING FRAMEWORK
FOR DISTANCE METRIC LEARNING
By Mr. Ratthachat Chatpatanasiri
Field of Study Computer Engineering
Thesis Advisor Professor Boonserm Kijisirikul, Ph.D.

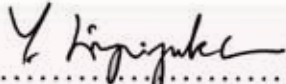
Accepted by the Faculty of Engineering, Chulalongkorn University in Partial
Fulfillment of the Requirements for the Doctoral Degree

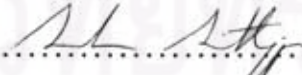

..... Dean of the Faculty of Engineering
(Associate Professor Boonsom Lerdhirunwong, Dr.Ing.)

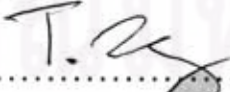
THESIS COMMITTEE


..... Chairman
(Professor Prabhas Chongstitvatana, Ph.D.)


..... Thesis Advisor
(Professor Boonserm Kijisirikul, Ph.D.)


..... Examiner
(Assistant Professor Yachai Limpiyakorn Ph.D.)


..... Examiner
(Assistant Professor Sukree Sinthupinyo, Ph.D.)


..... External Examiner
(Associate Professor Thanaruk Theeramunkong, Ph.D.)

รัฐฉัตร ฉัตรพัฒนศิริ : กรอบงานการเรียนรู้กึ่งมีผู้สอนแบบไม่เชิงเส้นสำหรับการเรียนรู้ฟังก์ชันระยะทาง (A NON-LINEAR SEMI-SUPERVISED LEARNING FRAMEWORK FOR DISTANCE METRIC LEARNING) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ศ.ดร. บุญเสริม กิจศิริกุล, 97 หน้า.

วิทยานิพนธ์ฉบับนี้ประกอบไปด้วยงานหลักสองส่วน ในส่วนแรกเป็นงานวิจัยที่พัฒนากรอบงานสำหรับเคอร์เนลไลซ์ระบบการเรียนรู้ระยะทางแบบมาฮาလာโนบิส โดยกรอบงานไม่เชิงเส้นที่พัฒนาขึ้นที่มีชื่อว่า “เคพีซีเอ ทริก” นี้มีข้อได้เปรียบมากกว่ากรอบงานที่ถูกพัฒนาขึ้นมาก่อนหน้าในหลายๆ ด้านด้วยกัน เช่น ผู้ใช้งานไม่จำเป็นต้องสร้างสมการคณิตศาสตร์ขึ้นมาใหม่ ผู้ใช้งานไม่จำเป็นต้องเขียนโปรแกรมใหม่ กรอบงานสามารถช่วยเพิ่มความเร็วให้กับอัลกอริทึม และกรอบงานนี้ไม่สร้างปัญหาจำพวกซิงกูลาร์ลิตี้เหมือนกรอบงานประเภทเก่า เป็นต้น ในงานนี้ ผู้เขียนได้พิสูจน์ทฤษฎีบทรีเฟรเชนเดอร์อย่างรัดกุมเพื่อแสดงความถูกต้องของกรอบงานที่สร้างขึ้นแม้ในมิติที่เป็นอนันต์แบบนับได้ นอกจากนี้ ในกรอบงานยังได้แสดงวิธีการเลือกเคอร์เนลที่เหมาะสมอย่างมีประสิทธิภาพอีกด้วย

ในงานส่วนที่สองเป็นส่วนของการพัฒนากรอบงานการเรียนรู้แบบกึ่งมีผู้สอนเพื่อการลดขนาดมิติของข้อมูลที่อยู่ในมานิโฟลด์ กรอบงานที่พัฒนาขึ้นได้ครอบคลุมกรอบงานที่ใช้เทคนิคสเป็คตรัลก่อนหน้าทั้งแบบมีผู้สอนและแบบไม่มีผู้สอน อัลกอริทึมที่ถูกสร้างบนกรอบงานที่นำเสนอในที่นี้สามารถใช้งานได้ทั้งแบบมีป้ายชื่อและแบบไม่มีป้ายชื่อ และสามารถใช้งานได้แม้ในสถานการณ์ที่ซับซ้อน เช่น ชุดข้อมูลเรียงตัวกันอยู่หลายคลัสเตอร์ และคลัสเตอร์นั้นอยู่บนมานิโฟลด์แบบไม่เชิงเส้น กรอบงานที่พัฒนาขึ้นนี้นอกจากจะมีขยายขอบเขตความสามารถของกรอบงานที่ถูกพัฒนาขึ้นมาก่อนแล้ว ยังได้ให้มุมมองที่เข้าใจง่ายเพื่ออธิบายความสัมพันธ์ของกรอบงานต่างๆ ที่ถูกพัฒนาขึ้นมาก่อนหน้าได้เป็นอย่างดี

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต
 สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
 ปีการศึกษา 2552

##4571838921 : MAJOR COMPUTER ENGINEERING

KEYWORD : KERNEL METHODS / METRIC LEARNING / SEMI-SUPERVISED

RATTHACHAT CHATPATANASIRI : A NON-LINEAR SEMI-SUPERVISED
LEARNING FRAMEWORK FOR DISTANCE METRIC LEARNING. THE-
SIS ADVISOR : Prof. Boonserm Kijirikul, Ph.D., 97 pp.

This dissertation consists of two main contributions. The first contribution focuses on developing a new framework of kernelizing Mahalanobis distance learners. The new KPCA trick framework offers several practical advantages over the classical kernel trick framework, e.g. no mathematical formulas and no reprogramming are required for a kernel implementation, a way to speed up an algorithm is provided with no extra work, the framework avoids troublesome problems such as singularity. Rigorous representer theorems in countably-infinite dimensional spaces are given to validate our framework. Furthermore, unlike previous works which always apply brute force methods to select a kernel, we derive a kernel alignment formula based on quadratic programming which can efficiently construct an appropriate kernel for a given dataset.

In the second contribution, we present a general framework of semi-supervised dimensionality reduction for manifold learning which naturally generalizes existing supervised and unsupervised learning frameworks which apply the spectral decomposition. Algorithms derived under our framework are able to employ both labeled and unlabeled examples and are able to handle complex problems where data form separate clusters of manifolds. Our framework offers simple views, explains relationships among existing frameworks and provides further extensions which can improve existing algorithms.

Department: Computer Engineering Student's Signature
Field of Study: Computer Engineering Advisor's Signature
Academic Year: 2009

Acknowledgements

It is not easy for me to write this section as I would like to mention many people who directly and indirectly helped or inspired me during my long PhD study for $7\frac{1}{2}$ years, including my 6 years at Chula, 10 months in UK, about 2 months in Germany and 7 months in Tokyo. I confess that at first I thought I can finish my PhD within 3-4 years. Practically, I used approximately twice of the time. This was due to my lack of research experiences which led me to change the main thesis topic for 4 times, until I finally settled with the current dissertation topic. During the $7\frac{1}{2}$ years, I met too many good friends, and it is difficult to mention them all in 1 page, the university limitation of this section. Thus, I postpone to the introduction section my gratitudes to the people outside MIND lab.

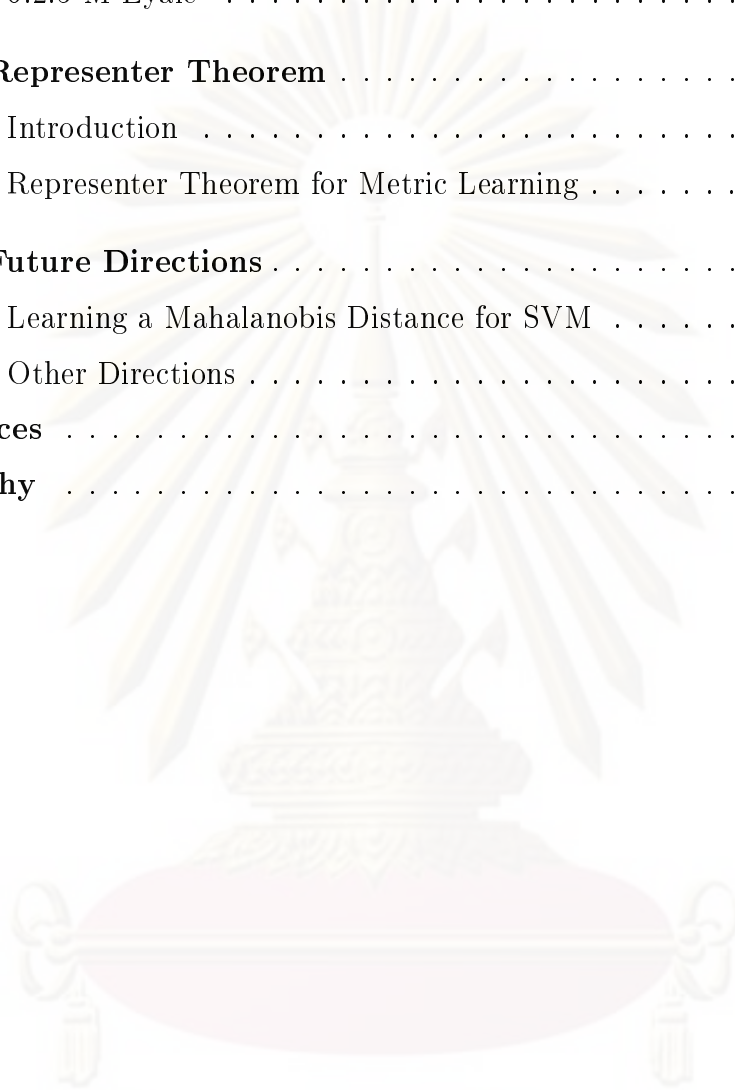
I have stayed in MIND lab for 6 years and have met many great friends. Prof. Boonserm Kijisirikul, my thesis advisor, always kindly supports me in every aspects, not only the research but also my life. Due to him, I was introduced to the extremely exciting fields of artificial intelligence and machine learning. I am in great debt to him. Prasertsak Pungprasertying stayed with me for about 4-5 years. He was the one who always listened to me whenever I wanted to talk about anything. Prasertask helped me implement many algorithms and prepare many datasets. He is not only my best colleague but also one of my best friend. Similarly, Tanasanee Phienthrakul and Anantaporn Srisawat always listened to me. I hope they agree that we had great time together. Teesid Korsrilabutr is responsible to the original idea of the weak representer theorem and unweighted combination of kernels presented in this thesis; the theorem become one of the pillars of my thesis. Further, he always helped me whenever I had problems regarding writing L^AT_EX. Pasakorn Tangchanachaianan is an amazing co-researcher. Even though he rarely talked, whenever he talked he always gave a new perspective of a problem. It was very fortunate for me to have the three junior members at MIND lab: Thanuphol Lerdlumnouchai, Pairit Nittayanuparp and Pittipol Kantavat. In the last 2 years, we had great time together talking about applications of machine learning on business and finance. Pittipol's master thesis introduced me to the world of stock market and Pairit introduced me to the derivative; he also invented the excellence cherngx.com, at which I enjoy watching everyday. Me and Thanupol have had a great fun together analyzing financial statements and various behaviors of stock markets. I thank Peeratham Wiriyathamabhum and Pathoomsiri Songsiri for helping me many tasks related to my thesis defense process: without them, I would be unable to graduate within this year. Due to space limitation, it is necessary to end this page and thank everyone else in MIND lab *here*.

Contents

	Page
Abstract (Thai)	iv
Abstract (English)	v
Acknowledgements	vi
Contents	vii
List of Tables	x
List of Figures	xi
Chapter	
I Introduction	1
1.1 Objectives	2
1.2 Structure of the Thesis	2
1.3 Publications	3
1.4 Acknowledgement (cont.)	4
II Background	6
2.1 Mahalanobis Distance Learning	6
2.2 Important Examples	8
2.2.1 Support Vector Machine (SVM)	9
2.2.2 Principal Component Analysis (PCA)	10
2.2.3 Fisher Discriminant Analysis (FDA)	11
2.2.4 Neighborhood Component Analysis (NCA)	12
2.2.5 Large Margin Nearest Neighbor (LMNN)	13
2.2.6 Discriminant Neighborhood Embedding (DNE)	14
III Non-Linearization using a KPCA Trick	16
3.1 Existing Frameworks	16
3.1.1 The Basis-Expansion Framework	17
3.1.2 The kernel trick Framework	18
3.2 The KPCA trick Framework	20
3.2.1 Literature Notes	21
3.2.2 The KPCA trick Algorithm	21
3.3 Representer Theorems	22
3.4 Remarks	24

	Page
3.5 KPCA Trick versus Kernel Trick	25
3.5.1 KNCA	25
3.5.2 KLMNN	27
3.5.3 KDNE	28
3.5.4 Practical Investigation	29
IV Kernel Selection	32
4.1 Kernel Alignment	32
4.2 Unweighted Kernels	35
4.3 Numerical Experiments	37
V Spectral Semi-Supervised Learning Framework: Theory	44
5.1 Introduction	44
5.2 Spectral Semi-Supervised Learning Framework	46
5.2.1 The Framework	47
5.2.2 Linear Parameterization	49
5.2.3 Specification of the Cost and Constraint Matrices	50
5.2.3.1 The Cost Matrix C^ℓ and the Constraint Matrix B	51
5.2.3.2 The Cost Matrix C^u and the Hadamard Power Operator	54
5.2.4 Non-Linear Parameterization Using the KPCA Trick	56
5.2.4.1 The KPCA trick Algorithm	57
5.2.4.2 Representer Theorems	58
5.2.5 Remarks	60
5.3 Connection to Related Work	63
5.3.1 Sugiyama et al. (2008)	64
5.3.2 Song et al. (2008)	65
5.3.3 Zhang et al. (2007a)	66
5.3.4 Li et al. (2007)	66
5.3.5 Improvement over Previous Frameworks	67
VI Spectral Semi-Supervised Learning Framework: Practice	70
6.1 Experimental Setting	70
6.2 Numerical Results	72
6.2.1 Ionosphere	72

	Page
6.2.2 Balance	74
6.2.3 BCI	75
6.2.4 USPS	76
6.2.5 M-Eyale	77
VII Representer Theorem	79
7.1 Introduction	79
7.2 Representer Theorem for Metric Learning	79
VIII Future Directions	87
8.1 Learning a Mahalanobis Distance for SVM	88
8.2 Other Directions	89
References	91
Biography	97



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

List of Tables

	Page
4.1 The average accuracy with standard deviation of NCA and their kernel versions. On the bottom row, the win/draw/lose statistics of each kernelized algorithm compared to its original version is drawn.	38
4.2 The average accuracy with standard deviation of LMNN and their kernel versions.	39
4.3 The average accuracy with standard deviation of DNE and their kernel versions.	39
6.1 Details of each dataset: d_0, c, ℓ, u and t denote the numbers of input features, classes, a number of labeled examples, a number of unlabeled examples and a number of testing examples, respectively. ‘*’ denotes the transductive setting used in small datasets, where all examples which are not labeled are given as unlabeled examples and used as testing examples as well. d , determined by using prior knowledge, denotes the target dimensionality for each dataset. “GOOD NEIGHBORS” denotes a quantity which measures a goodness of unlabeled data for each dataset.	71
6.2 Percentage accuracies of SS-DNE and SS-LFDA derived from our framework compared to existing algorithms ($\ell = 10$, except M-EYALE where $\ell = 20$). SS-LFDA and SS-DNE are highlighted when they outperform their opponents (LPP* and DNE for SS-DNE, and LPP*, LFDA and SELF for SS-LFDA). Superscripts indicate %-confidence levels of the one-tailed paired t-test for differences in accuracies between our algorithms and their best opponents. No superscripts denote confidence levels which below 80%. The accuracy of LapLS is also shown for comparison.	73
6.3 Percentage accuracies of SS-DNE and SS-LFDA compared to existing algorithms ($\ell = 100$).	73

List of Figures

	Page
2.1 The decision surface (the so-called <i>voronoi</i> diagram) created by the nearest neighbor algorithm based on the Euclidean distance.	6
2.2 An example of a Mahalanobis distance.	7
2.3 (Left) An example of a hyperplane obtained by the perceptron algorithm. (Right) An example of a hyperplane obtained by the linear SVM classifier. Learning a hyperplane is in fact equivalent to learning the 1-dimensional linear subspace which is orthogonal to the hyperplane (as illustrated by the red dashed-dotted lines in the two examples).	9
2.4 Two projection examples on a data of two classes: the left projection nicely separates the two classes while the right projection does not. . . .	9
2.5 A QP formulation for the SVM algorithm.	10
2.6 (Weinberger et al. (2006)). This figure illustrates the main intuition behind the LMNN algorithm. Before applying LMNN, data in the input space may be positioned randomly, but after applying LMNN and obtaining a linear transformation, data in the transformed space will be more ordered in the sense that <i>for each point, its k-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin.</i>	13
2.7 An SDP formulation for the LMNN algorithm.	14
3.1 Two examples where data lie in non-linear manifold. Mahalanobis distance learners do not perform well on these examples. More discussions on these two datasets can be found in Chapter 4.	16

- 3.2 (Left) At the beginning, data lie in their original input space. Note that in this input space data are not linearly separable. (Right) After transforming using the basis functions denoted by $\phi(\cdot)$, data now lie in a new space where data become linearly separable. Therefore, applying an existing hyperplane learner in the transformed space can separate the training data. More concretely, by denoting each point by its coordinate in the input space $\mathbf{x} = (x_1, x_2)$, the ellipse formula which perfectly separates the data is in the form of $\frac{x_1^2}{a} + \frac{x_2^2}{b} = 1$. By expanding into a space of 2-degree polynomial of $\phi(\mathbf{x}) = (z_1, z_2, z_3)$ where $z_1 = x_1^2$, $z_2 = x_2^2$ and $z_3 = 2x_1x_2$, the ellipse formula becomes linear with respect to the new variables: $\frac{z_1}{a} + \frac{z_2}{b} = 1$ 17
- 3.3 The KPCA trick algorithm. 22
- 3.4 An example of usual experimental results on IONOSPHERE where the KPCA trick (Left) and the kernel trick (Right) result in insignificantly-different subspaces (only one test data is differently classified). Big points denote training data and little points denote testing data where their predicted labels are shown. 30
- 3.5 Examples of datasets, PIMA and M-USPS, which are not ill-conditioned with respect to KDNE. Therefore, the two frameworks result in usual cases of indifferent or insignificantly-different Mahalanobis distances similar to that of Figure 3.4. Points illustrate accuracies for the two frameworks on experiment settings varying from two different polynomial-kernel degrees, $d \in \{2, \dots, 5\}$, $n \in \{50, 100, 150\}$ and $k \in \{1, \dots, 5\}$ (notations are introduced Chapter 2). 31
- 3.6 Examples of datasets, IONOSPHERE and GLASS, which are, in a few cases, ill-conditioned with respect to KDNE (setting is the same as Figure 3.5): the two frameworks occasionally give significantly different Mahalanobis distances. 31

4.1	Two synthetic examples where NCA, LMNN and DNE cannot learn any efficient Mahalanobis distances for kNN. Note that in each example, data in each class lie on a simple non-linear 1-dimensional subspace (which, however, cannot be discovered by the three learners). In contrast, the kernel versions of the three algorithms (using the 2^{nd} -order polynomial kernel) can learn very efficient distances, i.e., the non-linear subspaces can be discovered by the kernelized algorithms.	38
4.2	This figure illustrates performance of UKDNE with different numbers of base kernels. It can be observed from the figure that the generalization performance of UKDNE is consistently non-decreasing and will be eventually stable as we add more and more base kernels.	43
5.1	Given a contour of data (higher a number of data, darker a color), a rationale human can easily find a likely partition as one shown in the left and an unlikely partition similar to one shown in the right.	46
5.2	Our semi-supervised learning framework.	50
5.3	An example when data form a multi-modal structure. An algorithm, e.g. FDA, which imposes the condition (1) will try to discover a new subspace (a dashed line) which merges two clusters A and B altogether. An obtained space is undesirable as data of the two classes are mixed together. In contrast, an algorithm which imposes the condition (1*) (instead of (1)) will discover a subspace (a thick line) which does not merge the two clusters A and B as there are no nearby examples (indicated by a link between a pair of examples) between the two clusters.	52
5.4	The KPCA trick algorithm for semi-supervised learning.	58
5.5	The first toy example. The projection axes of three algorithms, namely FDA, LFDA, LPP and PCA, are presented. Big circles and big crosses denote labeled examples while small circles and small crosses denote unlabeled examples. Their percentage accuracy over the unlabeled examples are shown on the top.	67

List of Figures (cont.)

	Page
5.6 The second toy example consisting of three clusters of two classes. . . .	68
5.7 (Top) The third toy example where only a semi-supervised learner is able to find a good projection. (Bottom) An undirected graph corresponding to the values of C^u used by LPP and SS-LFDA. In this figure, a pair of examples i and j has a link if and only if $c_{ij}^u > 0.1$. This graph explains why LPP projects the data in the axis shown in the top figure; LPP, which does not apply the label information, tries to choose a projection axis which squeezes the two clusters as much as possible. Note that we apply a local-scaling method, Eq.(5.10), to specify C^u	69
6.1 The undirected graph corresponding to C^u constructed on IONOSPHERE. Each link corresponds to a pair of nearby examples having $c_{ij}^u \geq 0.36$. The number ‘0.36’ is just chosen for visualizability.	74
6.2 Zoom-in on the square area of Figure 6.2.	75
6.3 For each number x in the the x-axis, its corresponding value on the y-axis is the <i>ratio between the number of good nearby examples</i> (having $c_{ij}^u > x$ and belonging to the same class) <i>and the number of nearby examples</i> (having $c_{ij}^u > x$). The ratios with respect to C^{u^α} are demonstrated where (Top) $\alpha = 1$ (the standard LPP), and where (Bottom) $\alpha = 8$ (LPP*).	76
6.4 Extended Yale B Face dataset. 21 examples images from various illumination conditions.	77
8.1 Our first formulation for SVM.	89
8.2 Our second formulation for SVM.	89

CHAPTER I

INTRODUCTION

Mahalanobis distance learning is one of the most active research areas in recent years (Tenenbaum et al., 2000; Roweis and Saul, 2000; Xing et al., 2003; Cheng et al., 2004; He and Niyogi, 2004; Chen et al., 2005; Goldberger et al., 2005; Saul et al., 2006; Hoi et al., 2006; Weinberger et al., 2006; Yang et al., 2006a; Sugiyama, 2006, 2007; Cai et al., 2007; Yan et al., 2007; Zhang et al., 2007b; Torresani and Lee, 2007) The task of Mahalanobis distance learning is, in fact, fundamental to machine learning. It contains well-established algorithms such as support vector machines, perceptrons, principal component analysis and Fisher discriminant analysis as important special cases. In this thesis, we focus on improving further the framework of Mahalanobis distance learning.

One important disadvantage of all Mahalanobis distance learners is the inability to learn a *non-linear* transformation. This limitation is due to the fact that learning a Mahalanobis distance is equivalent to learning a linear map. In many real-world applications, data indeed form several clusters and lie on a non-linear subspace (manifold). In those cases, learners which are not able to discover a non-linear subspace will perform poorly.

There is another disadvantage of Mahalanobis distance learners. As many recent developments of Mahalanobis distance learners focus on prediction tasks, a considerable number of labeled examples are required in order to achieve satisfiable performance. In many real-world applications such as image classification, web page classification and protein function prediction, a labeling process is costly and time consuming; in contrast, unlabeled examples can be easily obtained. Therefore, in such situations, it can be beneficial to incorporate the information which is contained in unlabeled examples into a learning problem. The task of learning from both labeled and unlabeled examples is recently promoted and generally called *semi-supervised learning* (Chapelle et al., 2006). Nevertheless, only very few semi-supervised Mahalanobis distance learners exist.

1.1 Objectives

In this thesis, general non-linearization and semi-supervised frameworks are presented in order to extend a Mahalanobis distance learner to cope with a problem where its data lie on a manifold and a problem where only a small number of labeled examples are provided.

1.2 Structure of the Thesis

The remaining parts of this thesis consists of other seven chapters.

- In Chapter 2, general background on Mahalanobis distance learning is introduced. Further, important specific examples namely *support vector machines*, *principal component analysis*, *neighborhood component analysis*, *large margin nearest neighbor* and *discriminant neighborhood analysis* are given.

- In Chapter 3, non-linearization of Mahalanobis distance learners will be considered. Standard frameworks, namely *basis expansion* and *kernel trick*, are presented. Later, inspired from the kernel trick framework, the new framework called *KPCA trick* is presented and compared to the others. In short, in contrast to the kernel trick, the KPCA trick does not require users to derive new mathematical formulas. Also, whenever an implementation of an original learner is available, users are not required to re-implement the kernel version of the original learner. Moreover, the new framework avoids problems such as singularity in eigen-decomposition and provides a convenient way to speed up a learner.

- In the kernel trick and KPCA trick frameworks, kernel selection is fundamental. In Chapter 4, the problem of efficient kernel selection is dealt with. Firstly, we investigate the *kernel alignment* method proposed in previous works (Lanckriet et al., 2004; Zhu et al., 2005) to see whether it is appropriate for a kernelized Mahalanobis distance learner or not. New kernel alignment formulas based on quadratic programming and linear programming are derived. Secondly, we investigate a simple method which constructs an unweighted combination of base kernels. A theoretical result is provided to support this simple approach. Kernel constructions based on our two approaches require much shorter running

time when compared to the standard cross validation approach.

- In Chapter 5, we present a general semi-supervised dimensionality reduction framework which is able to employ information from both labeled and unlabeled examples. Algorithms developed in our framework are able to discover a nice (low-dimensional) subspace even when training examples of each class form separate clusters of complicated non-linear manifolds. In fact, many previous supervised and unsupervised algorithms can be casted as instances in our framework. Moreover, recent existing semi-supervised frameworks known to us (Li et al., 2007; Sugiyama et al., 2008; Song et al., 2008) can be viewed as special cases of our framework as well.

- In Chapter 6, practical performance of our semi-supervised learning framework is given in details.

- In Chapter 7, three representer theorems in the context of Mahalanobis distance learning are proven. Our theorems justify both the kernel trick and the KPCA trick frameworks in general learning settings, including unsupervised learning, supervised learning and semi-supervised learning. Moreover, the theorems validate kernelized algorithms learning a Mahalanobis distance in any separable Hilbert space and also cover kernelized algorithms performing dimensionality reduction. These theorems are extensions of that of Schölkopf et al. (2001) which cannot be applied to a general case of learning a countably-infinite dimensionality linear map.

- In Chapter 8, we conclude the thesis by discussing promising research directions in the future.

1.3 Publications

The main content of this dissertation are divided into two international journal papers and will be published on *Neurocomputing, special issue on subspace learning* (Chatpatanasiri et al., 2010; Chatpatanasiri and Kijirikul, 2010).

1.4 Acknowledgement (cont.)

Outside MIND lab, Dr. Wicharn Lewkeeratiyutkul is the one who taught me what really is mathematics. His teaching is always rigorous, intuitive and extremely friendly. I and Teesid applied the knowledge which he taught to our own works, especially the representer theorems presented in this dissertation. At Kasetsart University, Dr. Arnon Rungsawang first invited me to the world of research. Dr. Chaiyong and Dr. Nuanwan Soonthornphisaj introduced me to Prof. Boonserm, and also brought me to Freiburg University to have some great experience and great friends, especially Tapani Raiko. Dr. Jittat Fakcharoenphol and Dr. Parinya Chalermsook always inspire me about the fascinating world of theoretical research. Without them, I would quit my interest in theory long time ago.

At Cambridge, I owe a lot to Prof. David J. C. MacKay who gave me the real insights to Bayesian inference, Monte Carlo methods and information theory. Due to him, I believe my research skill is upgraded to a level beyond beginner, say, a beginning of the intermediate level. Furthermore, Prof. MacKay was the one who triggered my interests in sustainable technology and global warming. Also, I learned a lot from Phil Cowan, Oliver Stegle, Seb Wills, Philip Sterne and Sunny Li. Thanks to Dr. Kevin Knuth who supports me at the MAXENT workshop located in San Jose, USA. Thanks to NOKIA for supporting me on the Machine Learning Summer School program at Max-Planck Institute so that, there, I met many living legends and also good friends such as Bernd Gutmann and Florian Bloch. At the final year of my study, I had a great chance to visit Tokyo Institute of Technology to study on Meteorology, Hydrology and Sustainable Technology under the JENESYS program. I thanks Prof. Kumiko Yokoi, Prof. Manabu Kanda, Dr. Atsushi Inagaki, Nakayoshi Makoto, Kobayashi Kenji, Konno Rai, Nagano Kumiko, Fukumoto Eriko, Onomura Shiho, Takimoto Hiroshi, Cris Castillo, Yamashita Yoshimi, Yuko Okamoto, Ryo Shimoju and of course Alvin Christopher Varquez.

Due to several changes of my dissertation topics and titles, 8 thesis committees (besides my advisor) are involved during my study, I would like to thanks

them all: Prof. Prabhas, Dr. Attawith, Dr. Athasit, Dr. Chotirat, Dr. Yachai, Dr. Sukree, Dr. Thanaruk and Dr. Arnon. I also thanks the department's officers to help me for all my document stuffs and library stuffs.

Beside my academic friends, there are still many whom I have to mention: I think I am one of the most luckiest man in the universe. In addition to the people I mentioned above, I always got loves and supports from many people whom I have a great relationship with. All my family and my girlfriend – โฉ้ – have nevery complaint about my very long study and have always supported me. This helped me focused my life entirely on the study and so I have had a very good life. I'm very grateful for that.

I also always get supports from my Kasetsart University's CPE friends (in random orders): เต้ เต้ หนุ่ม ชาตรี ฝน นพ ออย ต้อม แอม เจน กุ้ง เบน โหลด ออบ ฝั่ง พัด ฮง ตี๋ว ชู่ ซาติ ปุ้ย ประกุด กุ่ย วิด ไร่ค เอ เอ็ด จิม เดร รัช บิ๊ก กอล์ฟ โก่อ้ ปอง เอส เอื้อ นุช อั้น คุ พี่อิงค์ พี่ตวง อ.กฤษณะ อ.เฉลิมศักดิ์ and also non CPE friends, เป้ ต้า เก้ หนู วอ หนุ่ย มิงค์ ทิพย์ เกต โบ. They all have brought happiness to my life.

At Chulalongkorn University (CU), in the CP and MATH departments, I was also truly lucky to have a great friends including มิน กิม เพชร กี้ก พี่นัท บอม พี่เมานิว ปุ้ย ยอด กอล์ฟ วุด ดิง พี่น่าน วิชญ์ บีม พี่แต่้ใหม่ อ้อ วุด.

Always on my mind, there are other friends: แสง วิ หรั่ง แม้ว สารจน์ ยอด โค้ช ฐา วุฒิ รุณ อาทิตย์ ป๊อด หนิง แจง ฟัก เจ พี่ป้อม ไตตัส

Finally, as in every of my paper, I have to say these important statements: this dissertation is mainly supported by Thailand Research Fund under the Royal Golden Jubilee's PhD scholarship program. The two journal papers are also supported by the 90th Anniversary of Chulalongkorn University Fund (Ratchadaphiseksomphot Endowment).

CHAPTER II

BACKGROUND

Almost, if not all, machine learning algorithms solving the problems of classification, regression, clustering, ranking and novelty detection (Schölkopf and Smola, 2001; Shawe-Taylor and Cristianini, 2004; Bishop, 2006) base their hypothesis constructions on the information of *Euclidean distances* among input data. See Figure 2.1 for an example. In fact, a choice of a distance function plays a crucial role on efficiency of these algorithms, and the use of the Euclidean distance function is not always appropriate to some applications. For an example, an object recognition problem where two images of one object can be very dissimilar, with respect to their Euclidean distance, due to a direction and a position of a camera. Therefore, the ability to learn the best distance function with respect to a given training set can improve the performance of a learner. In this thesis, we are interested in the class of *Mahalanobis distance functions on manifolds* which generalizes the Euclidean distance function.

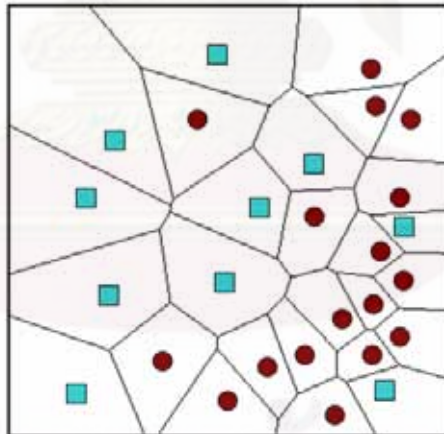


Figure 2.1: The decision surface (the so-called *voronoi* diagram) created by the nearest neighbor algorithm based on the Euclidean distance.

2.1 Mahalanobis Distance Learning

Let $\{\mathbf{x}_i, y_i\}_{i=1}^n$ denote a training set of n labeled examples with inputs $\mathbf{x}_i \in \mathbb{R}^D$ and corresponding class labels $y_i \in \{c_1, \dots, c_p\}$. Here, we denote $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$

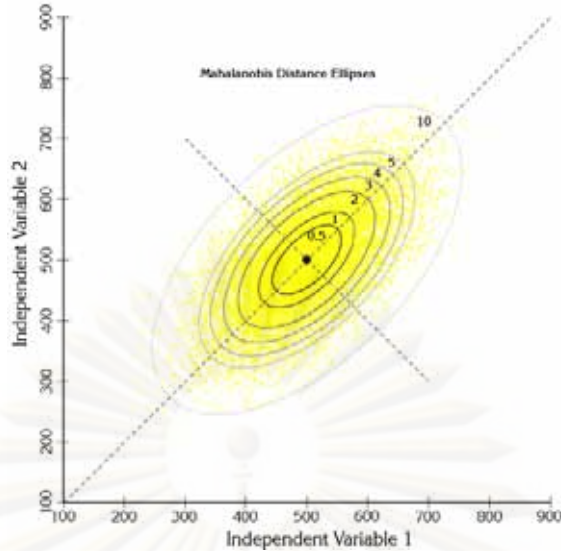


Figure 2.2: An example of a Mahalanobis distance.

as the matrix of input data. A distance function which belongs to the class of Mahalanobis distance can be represented by a symmetric positive semi-definite (PSD) matrix $M \in \mathbb{S}_+^D$. Here, we denote \mathbb{S}_+^D as a space of $D \times D$ PSD matrices. Given two points \mathbf{x}_i and \mathbf{x}_j , and a PSD matrix M , the Mahalanobis distance with respect to M between the two points is defined as

$$\|\mathbf{x}_i - \mathbf{x}_j\|_M = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T M (\mathbf{x}_i - \mathbf{x}_j)}.$$

A Mahalanobis distance is indeed a generalization of the Euclidean distance. Simply substituting $M = I$ results in Euclidean distance. In general, all points having equal distances from the origin form an ellipsoidal shape as shown in Figure 2.2. Note that in the case of the Euclidean distance this ellipsoidal shape becomes spherical.

The goal of Mahalanobis distance learners is to find a PSD matrix M^* that minimizes a reasonable objective function $f(\cdot)$:

$$M^* = \arg \min_{M \in \mathbb{S}_+^D} f(M). \quad (2.1)$$

Since the PSD matrix M can be decomposed to $A^T A$, we can equivalently restate

our problem as learning the best linear map, or matrix, A :

$$A^* = \arg \min_{A \in \mathbb{R}^{d \times D}} f(A^T A). \quad (2.2)$$

To simplify the notation, we will simply write $f(A)$ instead of $f(A^T A)$. Note that $d = D$ in the standard setting of learning a full-rank Mahalanobis distance; for the purpose of dimensionality reduction we can learn a low-rank projection by restricting $d < D$. In conventional pattern recognition tasks, after learning the best linear map A^* , A^* will be used by kNN to compute the distance between two points in the transformed space as $(\mathbf{x}_i - \mathbf{x}_j)^T M^*(\mathbf{x}_i - \mathbf{x}_j) = \|A^* \mathbf{x}_i - A^* \mathbf{x}_j\|^2$.

2.2 Important Examples

Learning a Mahalanobis distance, or a linear map, has several important special cases (Bishop, 2006).

- Many popular learners have their objectives to learn a *hyperplane*, which is in fact a *1-dimensional output linear map*, also called a *linear functional*, $A \in \mathbb{R}^{1 \times D}$. Popular examples belonging to this class include support vector machines, perceptrons and least-square methods (Figure 2.3).

- There is also an important class of *linear dimensionality reduction* algorithms which seek a weighted projection from a high-dimensional space \mathbb{R}^D to a low-dimensional space \mathbb{R}^d (Figure 2.4). These algorithms therefore learn a *d-dimensional output linear map* $A \in \mathbb{R}^{d \times D}$. Prime examples of algorithms belonging to this class are principal component analysis, Fisher discriminant analysis and their variations. In Chapter 5, we present a general framework generalizing many popular dimensionality reduction algorithms.

- Recently, algorithms aiming to learn a *full-rank linear map* $A \in \mathbb{R}^{D \times D}$ are presented. An invention of an algorithm in this class is now an active research area. Popular learners include neighborhood component analysis (Goldberger et al., 2005) and large margin nearest neighbor (Weinberger et al., 2006).

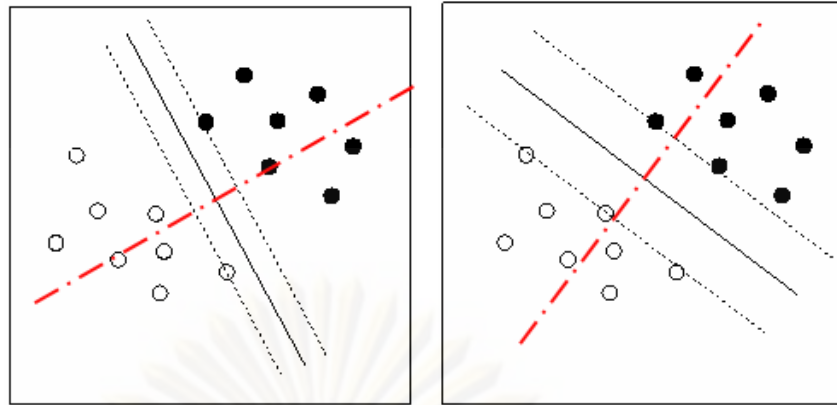


Figure 2.3: (Left) An example of a hyperplane obtained by the perceptron algorithm. (Right) An example of a hyperplane obtained by the linear SVM classifier. Learning a hyperplane is in fact equivalent to learning the 1-dimensional linear subspace which is orthogonal to the hyperplane (as illustrated by the red dashed-dotted lines in the two examples).

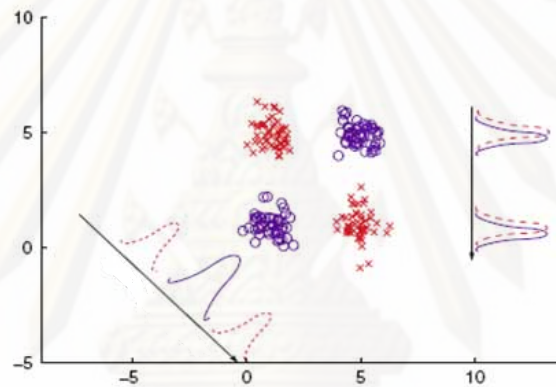


Figure 2.4: Two projection examples on a data of two classes: the left projection nicely separates the two classes while the right projection does not.

2.2.1 Support Vector Machine (SVM)

SVM is an algorithm designed for solving a binary classification problem. SVM aims to learn a hyperplane which can be used to classify data into their proper classes, positive or negative. Although, before the invention of the SVM algorithm, there were a number of algorithms proposed to learn a hyperplane, the unique feature of SVM is that it is able to learn an *optimal-margin* hyperplane¹.

¹Here, we avoid the misleading notion of “maximum margin” and follow (Schölkopf and Smola, 2001) to use the notion of “optimal margin” instead. Technically speaking, “maximum margin” hyperplane does not really exist since, given any hyperplane, we can increase its margin by simply multiplying the scale of the input space. What is really matter is the ratio of the margin to the radius of the smallest ball containing all given examples, not the margin itself.

Unlike other hyperplanes, it has been shown that an optimal-margin hyperplane has a nice theoretical property to guarantee its prediction accuracy on unseen data (Schölkopf and Smola, 2001; Shawe-Taylor and Cristianini, 2004).

Given a constant $c > 0$, the objective function of SVM solving a binary classification problem can be stated as

$$f^{SVM}(\mathbf{w}, b) = c\|\mathbf{w}\|^2 + \sum_{i=1}^n [1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)]_+,$$

where $[\cdot]_+$ denotes the standard hinge loss: $[z]_+ = \max(z, 0)$. We can restate the optimization problem of SVM as shown in Figure 2.5.

<p>Minimize $c\ \mathbf{w}\ ^2 + \sum_{i=1}^n \xi_i$ \mathbf{w}, b, ξ_i</p> <p>Subject to: $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i,$ $\xi_i \geq 0, \mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}.$</p>

Figure 2.5: A QP formulation for the SVM algorithm.

This SVM formulation is an instance of the class of (positive definite) quadratic programs (QP) (Boyd and Vandenberghe, 2004) and can be solved in a polynomial running time. After an optimal vector (\mathbf{w}, b) is obtained from solving a QP, an unseen data \mathbf{x}' can be classified by using the following formula:

$$\text{sign}(\langle \mathbf{w}, \mathbf{x}' \rangle + b). \tag{2.3}$$

2.2.2 Principal Component Analysis (PCA)

PCA is an algorithm for discovering a low-rank projection which best preserves a certain geometrical structure of input data. A geometrical structure which PCA attempts to preserve is the pairwise distance among each pair of the data. It can be shown that preserving the pairwise distances is equivalent to preserving the statistical variance of the data. Therefore, PCA is usually described as an algorithm which searches for a maximum-variance projection.

Denote the PCA objective function as $f^{PCA}(A)$. To simplify the formula,

we assume that training data are centered at the origin, i.e. $\sum_{i=1}^n \mathbf{x}_i = 0$. Then

$$f^{PCA}(A) = - \sum_{i=1}^n \|A\mathbf{x}_i\|^2. \quad (2.4)$$

By adding the *orthonormal projection* constraint $AA^T = I$, the PCA optimization problem is

$$A^* = \arg \min_{AA^T=I} \sum_{i=1}^n -\|A\mathbf{x}_i\|^2 = \arg \max_{AA^T=I} \sum_{i=1}^n \|A\mathbf{x}_i\|^2. \quad (2.5)$$

It can be shown that $A^* = (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)})^T$ where $\{\mathbf{a}^{(j)}\}_{j=1}^d$ is the top d eigenvectors of the following eigenvalue problem with respect to the covariance matrix of the data XX^T (Fukunaga, 1990):

$$XX^T \mathbf{a}^{(j)} = \lambda_j \mathbf{a}^{(j)}, \quad j = 1, \dots, d. \quad (2.6)$$

2.2.3 Fisher Discriminant Analysis (FDA)

FDA (Fukunaga, 1990) is a dimensionality reduction algorithm attempting to preserve a discriminative structure of a given set of data. Denote c as the number of classes in a given training set. Provided that training examples of each class lie in a linear subspace and *do not* form several separate clusters, i.e. *do not* form multi-modality, FDA is able to discover a low-dimensional linear subspace (with at most $c - 1$ dimensionality) which is efficient for classification. The objection function of FDA can be stated as follows:

$$f^{FDA}(A) = -\text{trace}\left((AS_w A^T)^{-1} AS_b A^T\right).$$

where S_b and S_w are standard between-class and within-class scatter matrices, respectively:

$$S_w = \sum_{i=1}^c \sum_{j|y_j=i} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \quad \text{and} \quad S_b = \sum_{i=1}^c (\boldsymbol{\mu} - \boldsymbol{\mu}_i)(\boldsymbol{\mu} - \boldsymbol{\mu}_i)^T,$$

where $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$, $\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{i=1}^{n_i} \mathbf{x}_i$ and n_i is a number of examples in the i^{th} class.

It can be shown that $A^* = (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)})^T$ where $\{\mathbf{a}^{(j)}\}_{j=1}^d$ is the top d eigenvectors of the following eigenvalue problem with respect to the matrix $S_w^{-1}S_b$:

$$S_w^{-1}S_b\mathbf{a}^{(j)} = \lambda_j\mathbf{a}^{(j)}, \quad j = 1, \dots, d. \quad (2.7)$$

2.2.4 Neighborhood Component Analysis (NCA)

NCA (Goldberger et al., 2005) is an algorithm obtaining a full-rank linear map for using with kNN. NCA attempts to optimize the *leave-one-out* (LOO) performance on training data. However, as the actual LOO classification error of kNN is a non-smooth function of the matrix A , Goldberger et al. propose to minimize a stochastic variant of the LOO kNN score which is defined as follows:

$$f^{NCA}(A) = - \sum_i \sum_{y_j=c_i} p_{ij}, \quad (2.8)$$

where

$$p_{ij} = \frac{\exp(-\|A\mathbf{x}_i - A\mathbf{x}_j\|^2)}{\sum_{k \neq i} \exp(-\|A\mathbf{x}_i - A\mathbf{x}_k\|^2)}, \quad p_{ii} = 0.$$

Optimizing $f^{NCA}(\cdot)$ can be done by applying a gradient based method such as delta-bar-delta or conjugate gradients. The formula of $\partial f^{NCA}/\partial A$ can be obtained as follows:

$$\frac{\partial f^{NCA}}{\partial A} = -2A \sum_i \left(p_i \sum_k p_{ik} \mathbf{x}_{ik} \mathbf{x}_{ik}^T - \sum_{j \in c_i} p_{ij} \mathbf{x}_{ij} \mathbf{x}_{ij}^T \right)$$

where for brevity we denote $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. One major disadvantage of NCA, however, is that $f^{NCA}(\cdot)$ is not convex, and the gradient based methods are thus prone to local optima.

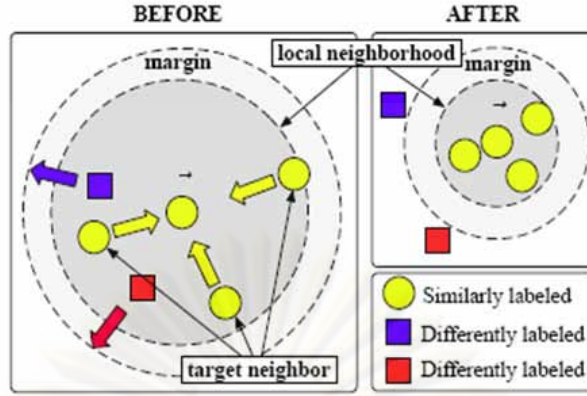


Figure 2.6: (Weinberger et al. (2006)). This figure illustrates the main intuition behind the LMNN algorithm. Before applying LMNN, data in the input space may be positioned randomly, but after applying LMNN and obtaining a linear transformation, data in the transformed space will be more ordered in the sense that *for each point, its k -nearest neighbors always belong to the same class while examples from different classes are separated by a large margin.*

2.2.5 Large Margin Nearest Neighbor (LMNN)

As NCA, LMNN (Weinberger et al., 2006) is an algorithm obtaining a full-rank linear map for using with kNN. In LMNN, the output Mahalanobis distance is optimized with the goal that *for each point, its k -nearest neighbors always belong to the same class while examples from different classes are separated by a large margin* (See Figure 2.6).

For each point \mathbf{x}_i , we define its k *target neighbors* as the k other inputs with the same label y_i that are closest to \mathbf{x}_i (with respect to the Euclidean distance in the input space). We use $w_{ij} \in \{0, 1\}$ to indicate whether an input \mathbf{x}_j is a target neighbor of an input \mathbf{x}_i . For convenience, we define $y_{ij} \in \{0, 1\}$ to indicate whether or not the labels y_i and y_j match. The objective function of LMNN is as follows:

$$f^{LMNN}(M) = \sum_{i,j} w_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|_M^2 + c \sum_{i,j,l} w_{ij} (1 - y_{il}) [1 + \|\mathbf{x}_i - \mathbf{x}_j\|_M^2 - \|\mathbf{x}_i - \mathbf{x}_l\|_M^2]_+.$$

The term $c > 0$ is a positive constant typically set by cross validation. The objective function above is convex and has two competing terms. The first term penalizes large distances between each input and its target neighbors, while the

second term penalizes small distances between each input and all other inputs that do not share the same label.

The objective function above can be reformulated as an instance of *semidefinite programs* (SDPs) (Boyd and Vandenberghe, 2004) as shown in Figure 2.7. Since SDP is an instance of convex programs, in contrast to $f^{NCA}(\cdot)$, the global optimum of $f^{LMNN}(\cdot)$ can be efficiently computed. A low-rank transformation $A \in \mathbb{R}^{d \times D}$ such that $d < D$ can be achieved by applying the orthogonal decomposition of M and retaining only the first d eigenvectors corresponding to the smallest d eigenvalues.

<p>Minimize $\sum_{i,j} w_{ij}(\mathbf{x}_i - \mathbf{x}_j)^T M(\mathbf{x}_i - \mathbf{x}_l) + c \sum_{i,j,l} w_{ij}(1 - y_{il})\xi_{ijl}$</p> <p>Subject to:</p> <p>(1) $(\mathbf{x}_i - \mathbf{x}_l)^T M(\mathbf{x}_i - \mathbf{x}_l) - (\mathbf{x}_i - \mathbf{x}_j)^T M(\mathbf{x}_i - \mathbf{x}_j) \geq 1 - \xi_{ijl}$.</p> <p>(2) $\xi_{ijl} \geq 0$.</p> <p>(3) $M \in \mathbb{S}_+^D$.</p>
--

Figure 2.7: An SDP formulation for the LMNN algorithm.

2.2.6 Discriminant Neighborhood Embedding (DNE)

As NCA and LMNN, DNE (Zhang et al., 2007b) is an algorithm obtaining a linear map for using with kNN. Nevertheless, the purpose of DNE is to obtain a low-rank linear map instead of a full-rank linear map. The main idea of DNE is quite similar to LMNN. DNE seeks a linear transformation such that neighborhood points in the same class are squeezed but those in different classes are separated as much as possible. However, DNE does not care about the notion of margin; in the case of LMNN, we want *every* point to stay far from points of other classes, but for DNE, we want the *average* distance between two neighborhood points of different classes to be large. Another difference is that LMNN can learn a full Mahalanobis distance, i.e. a general *weighted* linear projection, while DNE can learn only an *unweighted* linear projection.

Similar to LMNN, we define *two* sets of k target neighbors for each point \mathbf{x}_i based on the Euclidean distance in the input space. For each \mathbf{x}_i , let $Neig^I(i)$ be the set of k nearest neighbors having the same label y_i , and let $Neig^E(i)$ be the

set of k nearest neighbors having different labels from y_i . We define w_{ij} as follows:

$$w_{ij} = \begin{cases} +1, & \text{if } j \in \text{Neig}^I(i) \vee i \in \text{Neig}^I(j), \\ -1, & \text{if } j \in \text{Neig}^E(i) \vee i \in \text{Neig}^E(j), \\ 0, & \text{otherwise.} \end{cases}$$

The objective function of DNE is:

$$f^{DNE}(A) = \sum_{i,j} w_{i,j} \|A\mathbf{x}_i - A\mathbf{x}_j\|^2$$

which can be reformulated (up to a constant factor) to be

$$f^{DNE}(A) = \text{trace}(AX(D - W)X^T A^T),$$

where W is a symmetric matrix with elements w_{ij} , D is a diagonal matrix with $D_{ii} = \sum_j w_{ij}$ and X is the matrix of input points $(\mathbf{x}_1, \dots, \mathbf{x}_n)$. It is a well-known result from spectral graph theory (von Luxburg, 2007) that $D - W$ is symmetric but is not necessarily PSD. To solve the problem by eigen-decomposition, the constraint $AA^T = I$ is added (recall that $A \in \mathbb{R}^{d \times D}$ where $d \leq D$) so that we have the following optimization problem:

$$A^* = \arg \min_{AA^T=I} \text{trace}(AX(D - W)X^T A^T). \quad (2.9)$$

Then $A^* = (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)})^T$ where the optimal vectors $\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)}$ are the bottom eigenvectors of the following eigenvalue problem (Fukunaga, 1990, Chapter 10):

$$X(D - W)X^T \mathbf{a}^{(i)} = \lambda_i \mathbf{a}^{(i)}.$$

As LMNN, the global optimum of DNE can be efficiently computed.

One advantage of DNE over LMNN and NCA is that, for DNE, we have a deterministic rule to select the optimal dimensionality d of the transformed space; d will be the number of *negative* eigenvalues obtained from the above eigenvalue problem (Zhang et al., 2007b).

CHAPTER III

NON-LINEARIZATION USING A KPCA TRICK

As learning a Mahalanobis distance is equivalent to learning a linear map, a Mahalanobis distance may be still not appropriate for data lying in a manifold (a non-linear subspace) as shown in Figure 3.1. Non-linearization is a technique which allows a Mahalanobis distance learner to be able to learn an appropriate distance on a manifold.

A new framework introduced in this chapter is called a *KPCA trick*. We will show subsequently in the chapter the advantages of KPCA trick over existing frameworks.

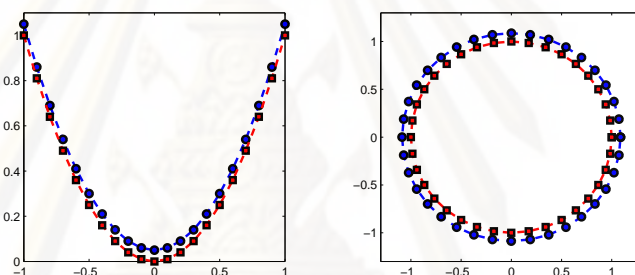


Figure 3.1: Two examples where data lie in non-linear manifold. Mahalanobis distance learners do not perform well on these examples. More discussions on these two datasets can be found in Chapter 4.

3.1 Existing Frameworks

Three popular non-linearization frameworks are the *basis expansion approach*, *neural networks*, and the *kernel trick* (Hastie et al., 2001; Schölkopf and Smola, 2001; Shawe-Taylor and Cristianini, 2004; Bishop, 2006). In this dissertation we are mainly interested in improving on the kernel trick framework, which is in fact an improvement over the basis expansion approach. Therefore, we explain here the two relevant frameworks. The framework of neural networks will not be discussed further in this dissertation.

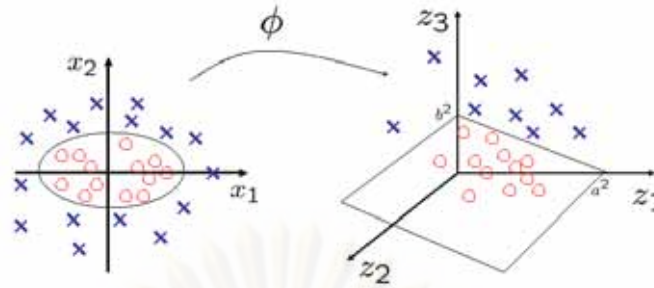


Figure 3.2: (Left) At the beginning, data lie in their original input space. Note that in this input space data are not linearly separable. (Right) After transforming using the basis functions denoted by $\phi(\cdot)$, data now lie in a new space where data become linearly separable. Therefore, applying an existing hyperplane learner in the transformed space can separate the training data. More concretely, by denoting each point by its coordinate in the input space $\mathbf{x} = (x_1, x_2)$, the ellipse formula which perfectly separates the data is in the form of $\frac{x_1^2}{a} + \frac{x_2^2}{b} = 1$. By expanding into a space of 2-degree polynomial of $\phi(\mathbf{x}) = (z_1, z_2, z_3)$ where $z_1 = x_1^2$, $z_2 = x_2^2$ and $z_3 = 2x_1x_2$, the ellipse formula becomes linear with respect to the new variables: $\frac{z_1}{a} + \frac{z_2}{b} = 1$.

3.1.1 The Basis-Expansion Framework

The basis-expansion framework is the simplest method for non-linearization. It can be described as a simple 2-step approach as follows:

- (1) select a set of basis functions and apply it to given data so that the data are transformed into a new space defined by the basis functions.
- (2) apply an existing algorithm (e.g. algorithms described in Section 2.2) in the new space.

This simple 2-step approach is illustrated in Figure 3.2. The main advantage of this approach lies in its simplicity: after applying the selected basis functions, any existing algorithm can be applied to the transformed data without the need of modification to the algorithm.

The choice of basis functions is crucial to this framework. The set of basis functions should be flexible enough so that data, forming complicated patterns

in an original space, have much more simpler sub-patterns in the transformed space. In those cases, simpler sub-patterns may be extracted by, e.g., a low-rank linear map in the transformed space. One common choice of basis functions is a set of polynomial basis functions (Schölkopf and Smola, 2001). Although a set of high-degree polynomial basis functions is powerful enough for any data, the number of basis functions grows exponentially with respect to the polynomial degree. As working in a high-dimensional space is computationally expensive, generally speaking, an application of high-degree polynomial basis functions is practically infeasible.

In the next subsection, we describe the kernel trick framework which brilliantly solves this computational issue.

3.1.2 The kernel trick Framework

The kernel trick framework was first applied to pattern recognition and machine learning by Vapnik and co-authors (Vapnik, 1999). The framework reduces the two steps of the basis-expansion framework into a magnificent single step. The essence of the kernel trick lies in the property of the positive semidefinite (PSD) kernel function and *representer theorems* (Schölkopf and Smola, 2001). Unfortunately, the existing representer theorems do not, in general, cover Mahalanobis distance learners. The representer theorem in the context of Mahalanobis distance learning will be discussed in Chapter 7.

Given a PSD kernel function $k(\cdot, \cdot)$, by the Mercer theorem (Schölkopf and Smola, 2001), we have

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \quad (3.1)$$

for some mapping $\phi(\cdot)$ which transforms data to a Hilbert space, often called a *feature space*. For simplicity, we shall denote ϕ , ϕ' and ϕ_i instead of $\phi(\mathbf{x})$, $\phi(\mathbf{x}')$ and $\phi(\mathbf{x}_i)$, respectively. A (squared) Mahalanobis distance under a matrix M in the feature space is

$$(\phi_i - \phi_j)^T M (\phi_i - \phi_j) = (\phi_i - \phi_j)^T A^T A (\phi_i - \phi_j). \quad (3.2)$$

As in Subsection 2.2.6, let $A = (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)})^T$. Denote a (possibly infinite-dimensional) matrix of the mapped training data $\Phi = (\phi_1, \dots, \phi_n)$. The main idea of the kernel trick framework in the context of Mahalanobis distance learning is to parameterize (see representer theorems Chapter 7)

$$A^T = \Phi U^T, \quad (3.3)$$

where $U = (\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(d)})^T$. Substituting A in Eq. (3.2) by using Eq. (3.3), we have

$$(\phi_i - \phi_j)^T M(\phi_i - \phi_j) = (\mathbf{k}_i - \mathbf{k}_j)^T U^T U (\mathbf{k}_i - \mathbf{k}_j),$$

where

$$\mathbf{k}_i = \Phi^T \phi_i = (\langle \phi_1, \phi_i \rangle, \dots, \langle \phi_n, \phi_i \rangle)^T. \quad (3.4)$$

Now our formula depends only on an inner-product $\langle \phi_i, \phi_j \rangle$, and thus the application of the Mercer theorem stating that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi_i, \phi_j \rangle$ can be applied. Therefore, the problem of learning the best Mahalanobis distance in the feature space is now reduced to a problem of learning the best (finite) linear transformation U of size $d \times n$.

Once we find the matrix U , the Mahalanobis distance from a new test point \mathbf{x}' to any input point \mathbf{x}_i in the feature space can be calculated as follows:

$$\|\phi' - \phi_i\|_M^2 = (\mathbf{k}' - \mathbf{k}_i)^T U^T U (\mathbf{k}' - \mathbf{k}_i), \quad (3.5)$$

where $\mathbf{k}' = (k(\mathbf{x}', \mathbf{x}_1), \dots, k(\mathbf{x}', \mathbf{x}_n))^T$. kNN classification in the feature space can be performed based on Eq. (3.5).

At this point, we note that, in contrast to the basis-expansion framework, there is no need to actually map the input data into a new high-dimensional space. By reformulating the problem using the kernel function, the kernel trick creates a computational shortcut for the basis-expansion framework by changing the optimized variable from A to U . There are many choices of valid PSD kernel functions (Vapnik, 1999; Schölkopf and Smola, 2001). Here, we note that applying

m -degree polynomial basis functions is equivalent to applying the kernel functions $k(\cdot, \cdot) = \langle \cdot, \cdot \rangle^m$.

Nonetheless, it often happens that an optimization with respect to a new variable U is much more complicated than the original optimization problem of with respect to a variable A in the input space, even their optimization problems look similar as shown in Section 3.5. In the next chapter, a new non-linearization framework which does not have this problem will be presented.

3.2 The KPCA trick Framework

In this section, we develop a *KPCA trick* framework which, compared to existing frameworks, can be much more conveniently applied to non-linearize the three learners. The KPCA trick is an improvement over the kernel trick and is also based on the Mercer theorem and representer theorems.

The KPCA trick framework, in fact, take us back to a 2-step approach as the basis-expansion framework. Nevertheless, unlike the basic-expansion framework where the number of basis functions may grow exponentially (if users uncarefully select a set of basis functions), the number of basis functions of the KPCA trick framework is always bounded by the number of data points. Hence, the KPCA trick framework has a computational running time in the same order as the kernel trick framework.

Denote $k(\cdot, \cdot)$, ϕ_i , ϕ and ϕ' as in Subsection 3.1.2. The central idea of the KPCA trick is to represent each ϕ_i and ϕ' in a new “finite”-dimensional space, without any loss of information. Within the framework, a new coordinate of each example is computed “explicitly”, and each example in the new coordinate is then used as the input of any existing Mahalanobis distance learner. As a result, by using the KPCA trick in place of the kernel trick, there is no need to derive new mathematical formulas and no need to implement new algorithms.

To simplify the discussion of KPCA, we assume that $\{\phi_i\}$ is linearly independent and has its center at the origin, i.e. $\sum_i \phi_i = 0$ (otherwise, $\{\phi_i\}$ can be centered by a simple pre-processing step (Shawe-Taylor and Cristianini, 2004, p.

115)). Since we have n total examples, the span of $\{\phi_i\}$ has dimensionality n . Here we claim that each example ϕ_i can be represented as $\varphi_i \in \mathbb{R}^n$ with respect to a new *orthonormal* basis $\{\psi_i\}_{i=1}^n$ such that $\text{span}(\{\psi_i\}_{i=1}^n)$ is the same as $\text{span}(\{\phi_i\}_{i=1}^n)$ without loss of any information. More precisely, we define

$$\varphi_i = \left(\langle \phi_i, \psi_1 \rangle, \dots, \langle \phi_i, \psi_n \rangle \right) = \Psi^T \phi_i. \quad (3.6)$$

where $\Psi = (\psi_1, \dots, \psi_n)$. Note that although we may be unable to numerically represent each ψ_i , an inner-product of $\langle \phi_i, \psi_j \rangle$ can be conveniently computed by KPCA (or kernel Gram-Schmidt (Shawe-Taylor and Cristianini, 2004)). Likewise, a new test point ϕ' can be mapped to $\varphi' = \Psi^T \phi'$. Consequently, the mapped data $\{\varphi_i\}$ and φ' are finite-dimensional and can be explicitly computed.

3.2.1 Literature Notes

Historically, the name *KPCA trick* was first appeared in the paper of Chapelle and Schölkopf (2001) who first applied this method to invariant support vector machines. Recently, Li et al. (2008a) invent similar trick in the context of data clustering. About the same time as our work, Zhang et al. (2009a) also propose the KPCA-trick in the context of finite-dimensionality reduction. Nevertheless, the applications of the KPCA-trick framework presented in Sect. 3.5 goes beyond what were shown in the previous works since these works constrain their method to the finite-dimensional cases. In contrast, learning a full Mahalanobis distance sometimes involves an infinite dimensional space. Thus, the new validation proof of the KPCA trick is needed in the context of Mahalanobis distance learning (see Theorem 1). Note that, parallel to our work, Jain et al. (2009) propose another kernelization framework which is complimentary to ours.

3.2.2 The KPCA trick Algorithm

The KPCA trick algorithm consisting of three simple steps is shown in Figure 3.3. In the algorithm, we denote a Mahalanobis distance learner by *maha* which performs the optimization process shown in Eq. (2.1) (or Eq. (2.2)) and outputs the best Mahalanobis distance M^* (or the best linear map A^*).

Input: 1. training examples: $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$,
 2. new example: \mathbf{x}' ,
 3. kernel function: $k(\cdot, \cdot)$
 4. Mahalanobis distance learning algorithm: maha

Algorithm:
 (1) Apply $\text{kpca}(k, \{\mathbf{x}_i\}, \mathbf{x}')$ such that $\{\mathbf{x}_i\} \mapsto \{\varphi_i\}$ and $\mathbf{x}' \mapsto \varphi'$.
 (2) Apply maha with new inputs $\{(\varphi_i, y_i)\}$ to achieve M^* or A^* .
 (3) Perform kNN based on the distance $\|\varphi_i - \varphi'\|_{M^*}$ or $\|A^*\varphi_i - A^*\varphi'\|$.

Figure 3.3: The KPCA trick algorithm.

NCA, LMNN and DNE described in Chapter 2 can all be kernelized by this simple algorithm as proved in Chapter 5. Besides NCA, LMNN and DNE, most Mahalanobis distance learners we know to date can be kernelized by this simple algorithm. For examples, Zhang et al. (2008, 2009b) recently proposed a general manifold learning framework called *patch alignment* containing many existing Mahalanobis distance learners (each of which is in fact a linear version of a manifold learner), including new learners such as *Local Coordinates Alignment* and *Discriminative Locality Alignment*. Although these manifold learners are able to learn non-linear subspaces without using the kernel-based frameworks, they do not provide a coordinate in a manifold for a new test data point (the so-called *out-of-sample* problem). Therefore, the KPCA trick can be applied to the patch alignment framework to provide a non-linear manifold subspace together with an out-of-sample mapping. Other learners whose kernel versions are previously undeveloped (Yang et al., 2006a; Xing et al., 2003; Li et al., 2008b) can easily apply the KPCA trick to get their non-linear versions as well.

3.3 Representer Theorems

Is it valid to represent an infinite-dimensional vector ϕ by a finite-dimensional vector φ ? In the context of SVMs (Chapelle and Schölkopf, 2001), this validity of the KPCA trick is easily achieved by straightforwardly extending a proof of classical representer theorems (Schölkopf et al., 2001; Kimeldorf and Wahba, 1971). In the context of Mahalanobis distance learning, however, proofs provided in previous works cannot be straightforwardly extended. The proof presented in Chapter 7 is a non-straightforward extension of Schölkopf and Smola's work. Note that, in the SVM case, what is learned is a hyperplane, a linear functional outputting a

1-dimensional value. In our case, what is learned is a linear map which, in general, outputs a *countably infinite dimensional* vector. Hence, to prove the validity of the KPCA trick in our case, we need some mathematical tools which can handle a countably infinite dimensionality. Below we state our versions of representer theorems which prove the validity of the KPCA trick in the current context. The proofs of the theorems, which also cover semi-supervised algorithms, will be given in Chapter 7.

By our representer theorems, it is the fact that, given an objective function $f(\cdot)$ (see Eq. (2.1)), the optimal value of $f(\cdot)$ based on the input $\{\phi_i\}$ is equal to the optimal value of $f(\cdot)$ based on the input $\{\varphi_i\}$. Hence, the representation of φ_i can be safely applied. We separate the problem of Mahalanobis distance learning into two different cases. The first theorem covers Mahalanobis distance learners (learning a full-rank linear transformation) while the second theorem covers dimensionality reduction algorithms (learning a low-rank linear transformation).

Theorem 1. (*Full-Rank Representer Theorem*) Let $\{\tilde{\psi}_i\}_{i=1}^n$ be a set of points in a feature space \mathcal{X} such that $\text{span}(\{\tilde{\psi}_i\}_{i=1}^n) = \text{span}(\{\phi_i\}_{i=1}^n)$, and \mathcal{X} and \mathcal{Y} be separable Hilbert spaces. For an objective function f depending only on $\{\langle A\phi_i, A\phi_j \rangle\}$, the optimization

$$\begin{aligned} \min_A : & f(\langle A\phi_1, A\phi_1 \rangle, \dots, \langle A\phi_i, A\phi_j \rangle, \dots, \langle A\phi_n, A\phi_n \rangle) \\ \text{s.t. } & A : \mathcal{X} \rightarrow \mathcal{Y} \text{ is a bounded linear map,} \end{aligned}$$

has the same optimal value as,

$$\min_{A' \in \mathbb{R}^{n \times n}} f(\tilde{\varphi}_1^T A'^T A' \tilde{\varphi}_1, \dots, \tilde{\varphi}_i^T A'^T A' \tilde{\varphi}_j, \dots, \tilde{\varphi}_n^T A'^T A' \tilde{\varphi}_n),$$

where $\tilde{\varphi}_i = \left(\langle \phi_i, \tilde{\psi}_1 \rangle, \dots, \langle \phi_i, \tilde{\psi}_n \rangle \right)^T \in \mathbb{R}^n$.

Theorem 2. (*Low-Rank Representer Theorem*) Define $\{\tilde{\psi}_i\}_{i=1}^n$ and $\tilde{\varphi}_i$ as in Theorem 1, f as an objective function depending only on $\{\langle A\phi_i, A\phi_j \rangle\}$. The optimiza-

tion

$$\begin{aligned} \min_A : & f(\langle A\phi_1, A\phi_1 \rangle, \dots, \langle A\phi_i, A\phi_j \rangle, \dots, \langle A\phi_n, A\phi_n \rangle) \\ \text{s.t. } & A : \mathcal{X} \rightarrow \mathbb{R}^d \text{ is a bounded linear map,} \end{aligned}$$

has the same optimal value as,

$$\min_{A' \in \mathbb{R}^{d \times n}} f(\tilde{\varphi}_1^T A'^T A' \tilde{\varphi}_1, \dots, \tilde{\varphi}_i^T A'^T A' \tilde{\varphi}_j, \dots, \tilde{\varphi}_n^T A'^T A' \tilde{\varphi}_n).$$

We note that Theorem 1 and Theorem 2 are more general than what is necessary for the KPCA trick. In fact, they justify both the kernel trick (by substituting $\tilde{\psi}_i = \phi_i$ and hence $\tilde{\varphi}_i = \mathbf{k}_i$) and the KPCA trick (by substituting $\tilde{\psi}_i = \psi_i$ and hence $\tilde{\varphi}_i = \varphi_i$).

3.4 Remarks

1. Note that by Mercer theorem (Schölkopf and Smola, 2001, pp. 37), we can either think of each $\phi_i \in \ell_2$ or $\phi_i \in \mathbb{R}^N$ for some positive integer N , and thus the assumption of Theorem 1 that \mathcal{X} , as well as \mathcal{Y} , is separable Hilbert space is then valid. Also, both theorems require that the objective function of a learning algorithm must depend only on $\{\langle A\phi_i, A\phi_j \rangle\}_{i,j=1}^n$ or equivalently $\{\langle \phi_i, M\phi_j \rangle\}_{i,j=1}^n$. This condition is, actually, not a strict condition since learners in literatures have their objective functions in this form (Chen et al., 2005; Goldberger et al., 2005; Globerson and Roweis, 2006; Weinberger et al., 2006; Yang et al., 2006a; Sugiyama, 2006; Yan et al., 2007; Zhang et al., 2007b; Torresani and Lee, 2007; Pang et al., 2006, 2008, 2009; Li et al., 2008b; Zhang et al., 2008, 2009b).

2. Note that the two theorems stated in this section do not require $\{\tilde{\psi}_i\}$ to be an orthonormal set. However, there is an advantage of the KPCA trick which restricts $\tilde{\psi}_i = \psi_i$ as in Eq. (3.6); this will be discussed in Section 3.5.

3. The statement of classical representer theorems deals with the representation of an optimal hyperplane (Schölkopf et al., 2001). In the same sense, our theorems also imply the representation of an optimal linear map A^* as shown in Chapter 7.

4. A running time of each learner strongly depends on the dimensionality of the input data. As recommended by Weinberger et al. (2006), it can be helpful to first apply a dimensionality reduction algorithm such as PCA before performing a learning process: the learning process can be tremendously speed up by retaining only, says, the 200 largest-variance principal components of the input data. In the KPCA trick framework illustrated in Figure 3.3, dimensionality reduction can be performed without any extra work as KPCA is already applied at the first place.

3.5 KPCA Trick versus Kernel Trick

To understand the advantages of the KPCA trick over the kernel trick, it is best to derive a kernel trick formula for each algorithm and see difficulties of implementing a kernel trick. Note that their original papers do not show how to kernelize these algorithms, and the material presented in this section is new. We denote KNCA, KLMNN and KDNE as the kernel versions of NCA, LMNN and DNE, respectively.

3.5.1 KNCA

As noted in Section 2.2.4, in order to minimize the objective of NCA and KNCA, we need to derive gradient formulas, and the formula of $\partial f^{KNCA}/\partial A$ is (Goldberger et al., 2005):

$$-2A \sum_i \left(p_i \sum_k p_{ik} \phi_{ik} \phi_{ik}^T - \sum_{j \in c_i} p_{ij} \phi_{ij} \phi_{ij}^T \right) \quad (3.7)$$

where for brevity we denote $\phi_{ij} = \phi_i - \phi_j$. Nevertheless, since ϕ_i may lie in an infinite dimensional space, the above formula cannot be always implemented in practice. In order to implement the kernel trick version of KNCA, users need to prove the following proposition which is not stated in the original work of Goldberger et al. (2005).

Proposition 1. $\partial f^{KNCA}/\partial A$ can be formulated as $V\Phi^T$ where V depends on $\{\phi_i\}$ only in the form of $\langle \phi_i, \phi_j \rangle = k(\mathbf{x}_i, \mathbf{x}_j)$, and thus we can compute all elements of V .

Proof. Define a matrix $B_i^\phi = (0, 0, \dots, \phi, \dots, 0, 0)$ as a matrix with its i^{th} column is ϕ and zero vectors otherwise. Denote $\mathbf{k}_{ij} = \mathbf{k}_i - \mathbf{k}_j$. Substitute $A = U\Phi^T$ to Eq. (3.7) we have

$$\begin{aligned} \frac{\partial f^{KNCA}}{\partial A} &= -2U \sum_i \left(p_i \sum_k p_{ik} \mathbf{k}_{ik} \phi_{ik}^T - \sum_{j \in c_i} p_{ij} \mathbf{k}_{ij} \phi_{ij}^T \right) \\ &= -2U \sum_i \left(p_i \sum_k p_{ik} (B_i^{\mathbf{k}_{ik}} - B_k^{\mathbf{k}_{ik}}) - \right. \\ &\quad \left. \sum_{j \in c_i} p_{ij} (B_i^{\mathbf{k}_{ij}} - B_j^{\mathbf{k}_{ij}}) \right) \Phi^T \\ &= V\Phi^T, \end{aligned}$$

which completes the proof. \square

Therefore, at the i^{th} iteration of an optimization step of a gradient optimizer, we needs to update the current best linear map as follows:

$$\begin{aligned} A^{(i)} &= A^{(i-1)} + \epsilon \frac{\partial f^{KNCA}}{\partial A} = (U^{(i-1)} + \epsilon V^{(i-1)}) \Phi^T \\ &= U^{(i)} \Phi^T, \end{aligned} \tag{3.8}$$

where ϵ is a step size. The kernel trick formulas of KNCA are thus finally achieved. However, we emphasize that the process of proving Proposition 1 and Eq. (3.8) is not trivial and may be tedious and difficult for non-experts as well as practitioners who focus their tasks on applications rather than theories. Moreover, since the formula of $\partial f^{KNCA}/\partial A$ is significantly different from $\partial f^{NCA}/\partial A$, users are required to re-implement KNCA (even they already possess an NCA implementation) which is again not at all convenient. In contrast, we note that all these difficulties are disappeared if the KPCA trick algorithm consisting of three simple steps shown in Fig. 3.3 is applied instead of the kernel trick.

There is another advantage of using the KPCA trick on KNCA¹. By the nature of a gradient optimizer, it takes a large amount of time for NCA and KNCA to converge to a local solution, and thus a method of speeding up the algorithms is needed. As remarked in Section 3.4, the learning process can be

¹We slightly modify the code of Charless Fowlkes: <http://www.cs.berkeley.edu/~fowlkes/software/nca/>

tremendously speed up by retaining only, says, the 100 largest-variance principal components of the input data. In the KPCA trick framework, no extra work is required for this speed-up task as KPCA is already applied at the first place.

3.5.2 KLMNN

Similar to KNCA, the online-available code of LMNN² employs a gradient based optimization, and thus new gradient formulas in the feature space has to be derived and new implementation has to be done in order to apply the kernel trick³. On the other hand, by applying the KPCA trick, the original LMNN code can be immediately used.

There is another advantage of the KPCA trick on LMNN: LMNN requires a specification of w_{ij} which is usually based on the quantity $\|\mathbf{x}_i - \mathbf{x}_j\|$. Thus, it makes sense that w_{ij} should be based on $\|\phi_i - \phi_j\| = \sqrt{k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j)}$ with respect to the feature space of KLMNN, and hence, with the kernel trick, users have to modify the original code in order to appropriately specify w_{ij} . In contrast, by applying the KPCA trick which restricts $\{\psi_i\}$ to be an orthonormal set as in Eq. (3.6), we have the following proposition.

Proposition 2. *Let $\{\psi_i\}_{i=1}^n$ be an orthonormal set such that $\text{span}(\{\psi_i\}_{i=1}^n) = \text{span}(\{\phi_i\}_{i=1}^n)$ and $\varphi_i = (\langle \phi_i, \psi_1 \rangle, \dots, \langle \phi_i, \psi_n \rangle)^T \in \mathbb{R}^n$, then $\|\varphi_i - \varphi_j\|^2 = \|\phi_i - \phi_j\|^2$ for each $1 \leq i, j \leq n$.*

Proof. Since we work on a separable Hilbert space \mathcal{X} , we can extend the orthonormal set $\{\psi_i\}_{i=1}^n$ to $\{\psi_i\}_{i=1}^\infty$ such that $\overline{\text{span}(\{\psi_i\}_{i=1}^\infty)}$ is \mathcal{X} and $\langle \phi_i, \psi_j \rangle = 0$ for each $i = 1, \dots, n$ and $j > n$. Then, by an application of the Parseval identity (Lewkeeratiyutkul, 2006),

$$\begin{aligned} \|\phi_i - \phi_j\|^2 &= \sum_{k=1}^{\infty} \langle \phi_i - \phi_j, \psi_k \rangle^2 = \sum_{k=1}^n \langle \phi_i - \phi_j, \psi_k \rangle^2 \\ &= \|\varphi_i - \varphi_j\|^2. \end{aligned}$$

²<http://www.weinbergerweb.net/Downloads/LMNN.html>

³There is a variation on LMNN called “large margin component analysis” (LMCA) (Torresani and Lee, 2007) which proposes to optimize A instead of M ; however, LMCA does not preserve some desirable properties, such as convexity, of LMNN, and therefore the algorithm “Kernel LMCA” presented there is different from “Kernel LMNN” presented in this chapter.

The last equality comes from Eq.(3.6) and the fact that $\Psi^T(\phi_i - \phi_j) = (\varphi_i - \varphi_j)$. \square

Therefore, with the KPCA trick, the target neighbors w_{ij} of each point is computed based on $\|\varphi_i - \varphi_j\| = \|\phi_i - \phi_j\|$ without any modification of the original code.

3.5.3 KDNE

By applying $A = U\Phi^T$ and defining the gram matrix $K = \Phi^T\Phi$, we have the following proposition.

Proposition 3. *The kernel trick formula of KDNE is the following minimization problem:*

$$U^* = \arg \min_{UKU^T=I} \text{trace}(UK(D - W)KU^T). \quad (3.9)$$

Note that this kernel trick formula of KDNE involves a *generalized eigenvalue problem* instead of a plain eigenvalue problem involved in DNE. As a consequence, we face a *singularity* problem, i.e. if K is not full-rank, the constraint $UKU^T = I$ cannot be satisfied. Using elementary linear algebra, it can be shown that K is not full-rank if and only if $\{\phi_i\}$ is not linearly independent, and this condition is not highly improbable. Sugiyama (2006), Yu and Yang (2001), and Yang and Yang (2003) suggest methods to cope with the singularity problem in the context of Fisher discriminant analysis which may be applicable to KDNE. Sugiyama (2006) recommends to use the constraint $U(K + \epsilon I)U^T = I$ instead of the original constraint; however, an appropriate value of ϵ has to be tuned by cross validation which is time-consuming. Alternatively, Yu and Yang (2001) and Yang and Yang (2003) propose more complicated methods of directly minimizing an objective function in the null space of the constraint matrix so that the singularity problem is explicitly avoided.

We note that a KPCA trick implementation of KDNE does not have this singularity problem as only a plain eigenvalue problem has to be solved. Moreover, as in KLMNN, applying the KPCA trick instead of the kernel trick to KDNE avoids the tedious task of modifying the original code to appropriately specify w_{ij}

in the feature space.

3.5.4 Practical Investigation

The representer theorems state that, in each learning problem, optimal points of the two frameworks must have the same objective value. Nevertheless, for learners where their objective functions are not convex (e.g. KNCA) or not strictly-convex (e.g. KLMNN⁴), it is not surprising that the two frameworks may not result in an identical Mahalanobis distance. In the case of non-convexity a learner itself does not guarantee to find a global optimal solution, and in the case of non-strict-convexity there are plenty of global optimal solutions so that an obtained Mahalanobis distance does not depend on the KPCA trick or the kernel trick, but on an initial condition and on an optimizer's mechanism.

It is interesting to note that, in practice, even KDNE, which has a strictly convex (quadratic) objective function, can sometimes have different results obtained from the two kernelization frameworks. This is mainly because KDNE's involved matrices, K and $K(D - W)K$, can be *ill-conditioned*, i.e. their *condition numbers* (Demmel, 1997; Ng et al., 2001) are very high. See also Section 3.3 of Bach and Jordan (2006). The ill-conditioned of the two matrices may lead to inaccurate numerical computations on eigen-decomposition and generalized eigen-decomposition which are employed by the KPCA trick and the kernel trick, respectively.

Here, we show experimental results on standard datasets obtained from the UCI repository (Asuncion and Newman, 2007) and a high-dimensional dataset called M-USPS which is a modified version of the famous USPS dataset (Chapelle et al., 2006, Chapter 21). Figure 3.4 illustrates a situation where the two frameworks result in very-slightly different subspaces where the condition number of $K(D - W)K$ is in the order of 10^{18} . The difference seems to be indistinguishable by human, and their accuracies on the test dataset are different by only one test data. This example shows that matrices having condition numbers in the order of 10^{18} are still not ill-conditioned with respect to the MATLAB's eig function. Figure 3.5 illustrates more examples about this usual indistinguishable-difference

⁴Because of the $[\cdot]_+$ function in its objective function.

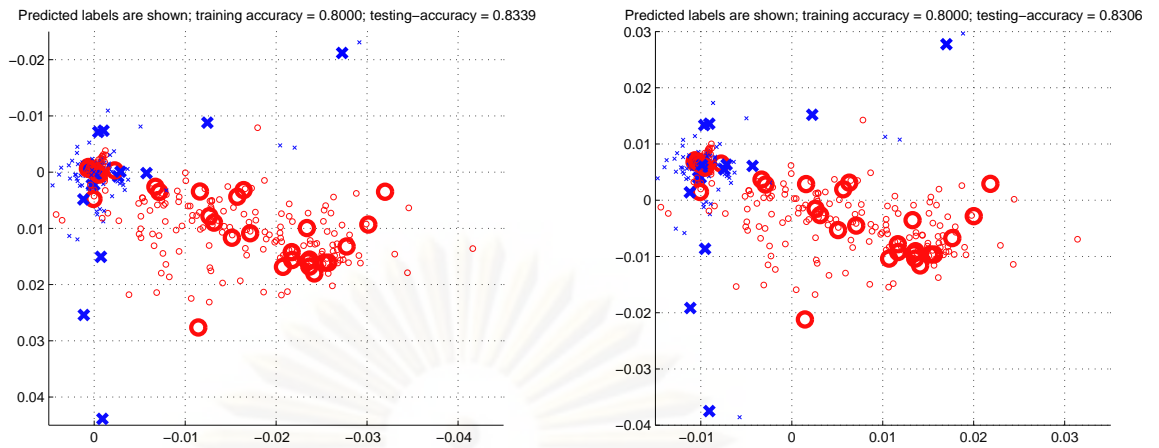


Figure 3.4: An example of usual experimental results on IONOSPHERE where the KPCA trick (Left) and the kernel trick (Right) result in insignificantly-different subspaces (only one test data is differently classified). Big points denote training data and little points denote testing data where their predicted labels are shown.

situation. Nevertheless, when condition numbers are extremely high, differences become distinguishable. Figure 3.6 illustrates examples about this distinguishable-difference situation. There are few cases on Figure 3.6 where the two frameworks provide totally different accuracies. Condition numbers of K and $K(D - W)K$ of these cases are as high as 10^{32} for IONOSPHERE and 10^{35} for GLASS. Investigations of various numerical eigen-decomposition approaches and of methods to improve the condition number of a matrix (e.g. Pan et al. (2009)) are beyond the scope of this paper and are potential future works.

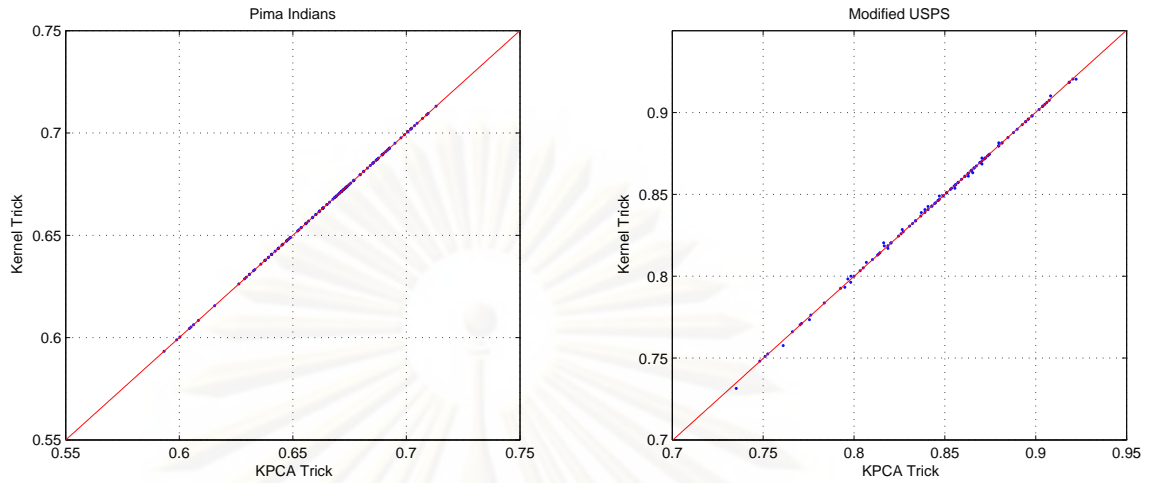


Figure 3.5: Examples of datasets, PIMA and M-USPS, which are not ill-conditioned with respect to KDNE. Therefore, the two frameworks result in usual cases of indifferent or insignificantly-different Mahalanobis distances similar to that of Figure 3.4. Points illustrate accuracies for the two frameworks on experiment settings varying from two different polynomial-kernel degrees, $d \in \{2, \dots, 5\}$, $n \in \{50, 100, 150\}$ and $k \in \{1, \dots, 5\}$ (notations are introduced Chapter 2).

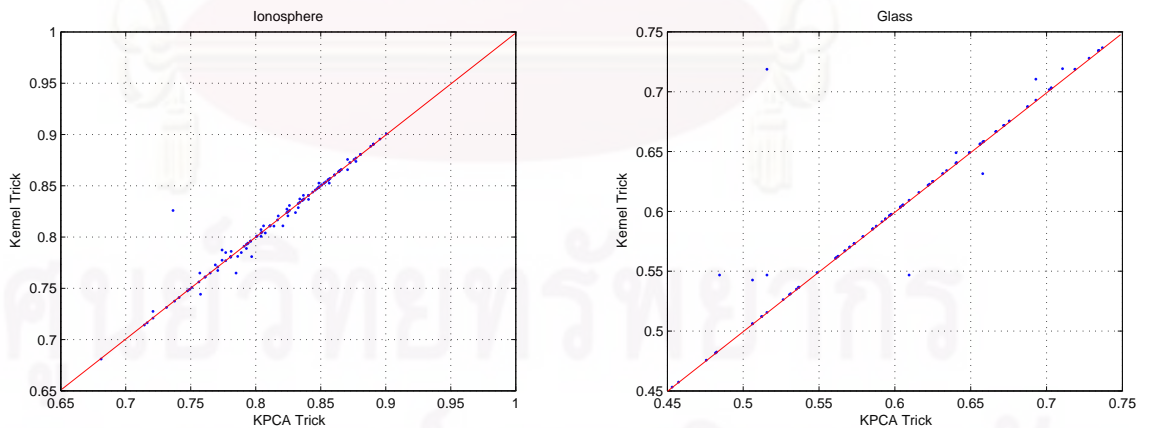


Figure 3.6: Examples of datasets, IONOSPHERE and GLASS, which are, in a few cases, ill-conditioned with respect to KDNE (setting is the same as Figure 3.5): the two frameworks occasionally give significantly different Mahalanobis distances.

CHAPTER IV

KERNEL SELECTION

The problem of selecting an efficient kernel function is central to all kernel machines. All previous works on Mahalanobis distance learners use exhaustive methods such as cross validation to select a kernel function. In this chapter, we investigate a possibility to automatically construct a kernel which is appropriate for a Mahalanobis distance learner.

In the first part of this chapter, we consider a popular method called *kernel alignment* (Lanckriet et al., 2004; Zhu et al., 2005) which is able to learn, from a training set, a kernel in the form of $k(\cdot, \cdot) = \sum_i \alpha_i k_i(\cdot, \cdot)$ where $k_1(\cdot, \cdot), \dots, k_m(\cdot, \cdot)$ are pre-chosen base kernels. New kernel alignment formulas based on quadratic programming (QP) and linear programming (LP) are derived. As the previous formulas are based on semidefinite programming (SDP) and quadratically constraint quadratic programming (QCQP), our formulas should be preferred. In the second part, we investigate a simple method which constructs an unweighted combination of base kernels, $\sum_i k_i(\cdot, \cdot)$ (henceforth referred to as an *unweighted kernel*). A theoretical result is provided to support this simple approach.

While accuracy performance is comparable, kernel constructions based on our two approaches require much shorter running time when compared to the standard cross validation approach.

4.1 Kernel Alignment

Kernel alignment is one of a popular technique for kernel selection (Goen and Alpaydm, 2010). Our kernel alignment formulation presented in this section belongs to the class of quadratic programs (QPs) which can be solved more efficiently than the formulations proposed by Lanckriet et al. (2004) and Zhu et al. (2005) which belong to the class of semidefinite programs (SDPs) and quadratically constrained quadratic programs (QCQPs), respectively (Boyd and Vandenberghe, 2004).

To use kernel alignment in classification problems, the following assumption is central: for each couple of examples $\mathbf{x}_i, \mathbf{x}_j$, the ideal kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ is Y_{ij} (Guermeur et al., 2004) where

$$Y_{ij} = \begin{cases} +1, & \text{if } y_i = y_j, \\ \frac{-1}{p-1}, & \text{otherwise,} \end{cases}$$

and p is the number of classes in the training data. Denoting Y as the matrix having elements of Y_{ij} , we then define the *alignment* between the kernel matrix K and the ideal kernel matrix Y as follows:

$$\text{align}(K, Y) = \frac{\langle K, Y \rangle_F}{\|K\|_F \|Y\|_F}, \quad (4.1)$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner-product such that $\langle K, Y \rangle_F = \text{trace}(K^T Y)$ and $\|\cdot\|_F$ is the Frobenius norm induced by the Frobenius inner-product.

Assume that we have m kernel functions, $k_1(\cdot, \cdot), \dots, k_m(\cdot, \cdot)$, and K_1, \dots, K_m are their corresponding Gram matrices with respect to the training data. Here, the kernel function obtained from the alignment method is parameterized in the form of $k(\cdot, \cdot) = \sum_i \alpha_i k_i(\cdot, \cdot)$ where $\alpha_i \geq 0$. Note that the obtained kernel function is guaranteed to be positive semidefinite. In order to learn the best coefficients $\alpha_1, \dots, \alpha_m$, we solve the following optimization problem:

$$\{\alpha_1, \dots, \alpha_m\} = \arg \max_{\alpha_i \geq 0} \text{align}(K, Y), \quad (4.2)$$

where $K = \sum_i \alpha_i K_i$. Note that as K and Y are PSD, $\langle K, Y \rangle_F \geq 0$. Since both the numerator and denominator terms in the alignment equation can be arbitrary large, we can simply fix the numerator to 1. We then reformulate the problem as follows:

$$\begin{aligned} \arg \max_{\alpha_i \geq 0, \langle K, Y \rangle_F = 1} \text{align}(K, Y) &= \arg \min_{\alpha_i \geq 0, \langle K, Y \rangle_F = 1} \|K\|_F \|Y\|_F \\ &= \arg \min_{\alpha_i \geq 0, \langle K, Y \rangle_F = 1} \|K\|_F^2 \\ &= \arg \min_{\alpha_i \geq 0, \sum_i \alpha_i \langle K_i, Y \rangle_F = 1} \sum_{i,j} \alpha_i \alpha_j \langle K_i, K_j \rangle_F. \end{aligned}$$

Defining a vector $\mathbf{b} = (\langle K_1, Y \rangle_F, \dots, \langle K_m, Y \rangle_F)^T$, a PSD matrix S whose elements $S_{ij} = \langle K_i, K_j \rangle_F$, and a vector $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)^T$, we then reformulate Eq. (4.2) as follows:

$$\boldsymbol{\alpha} = \arg \min_{\alpha_i \geq 0, \boldsymbol{\alpha}^T \mathbf{b} = 1} \boldsymbol{\alpha}^T S \boldsymbol{\alpha}. \quad (4.3)$$

This optimization problem is a QP and can be efficiently solved (Boyd and Vandenberghe, 2004); hence, we are able to learn the best kernel function $k(\cdot, \cdot) = \sum_i \alpha_i k_i(\cdot, \cdot)$ efficiently.

Since the magnitudes of the optimal α_i are varied due to $\|K_i\|_F$, it is convenient to use $k'_i(\cdot, \cdot) = k_i(\cdot, \cdot) / \|K_i\|_F$ and hence $K'_i = K_i / \|K_i\|_F$ in the derivation of Eq. (4.3). We define S' and \mathbf{b}' similar to S and \mathbf{b} except that they are based on K'_i instead of K_i . Let

$$\boldsymbol{\gamma} = \arg \min_{\gamma_i \geq 0, \boldsymbol{\gamma}^T \mathbf{b}' = 1} \boldsymbol{\gamma}^T S' \boldsymbol{\gamma}. \quad (4.4)$$

It is easy to see that the final kernel function $k(\cdot, \cdot) = \sum_i \gamma_i k'_i(\cdot, \cdot)$ achieved from Eq. (4.4) is not changed from the kernel achieved from Eq. (4.3).

Note that we can further modify Eq. (4.3) to enforce sparseness of $\boldsymbol{\alpha}$ and improve a speed of an algorithm by minimizing an upper bound of $\|K\|_F$ instead of minimizing the exact quantity so that the optimization formula belongs to the class of linear programs (LPs) instead of QPs.

$$\min_{\alpha_i \geq 0, \langle K, Y \rangle_F = 1} \|K\|_F \leq \min_{\alpha_i \geq 0, \langle K, Y \rangle_F = 1} \|\text{vec}(K)\|_1 \quad (4.5)$$

where $\text{vec}(\cdot)$ denotes a standard “vec” operator converting a matrix to a vector (Minka, 1997). By using a standard trick for an absolute-valued objective function (Boyd and Vandenberghe, 2004), Eq. (4.5) can be solved by linear programming. Note that the above optimization algorithm of minimizing the upper bound of a desired objective function is similar to the popular support vector machines where the hinge loss is minimized instead of the 0/1 loss.

4.2 Unweighted Kernels

In this section, we show that a very simple kernel $k'(\cdot, \cdot) = \sum_i k_i(\cdot, \cdot)$ is theoretically efficient (based on the value of a given objective function), no less than a kernel obtained from the alignment method. Denote ϕ_i^k as a mapped vector of an original example \mathbf{x}_i by a map associated with a kernel $k(\cdot, \cdot)$. The main idea of the contents presented in this section is the following simple but useful result.

Proposition 4. *Let $\{\alpha_i\}$ be a set of positive coefficients, $\alpha_i > 0$ for each i , and let $k_1(\cdot, \cdot), \dots, k_m(\cdot, \cdot)$ be base PSD kernels and $k(\cdot, \cdot) = \sum_i \alpha_i k_i(\cdot, \cdot)$ and $k'(\cdot, \cdot) = \sum_i k_i(\cdot, \cdot)$. Then, there exists an invertible linear map B such that $B : \phi_i^{k'} \rightarrow \phi_i^k$ for each i .*

Proof. Without loss of generality, we will concern here only the case of $m = 2$; the cases such that $m > 2$ can be proven by induction. Let $\mathcal{H}_i \oplus \mathcal{H}_j$ be a direct sum of \mathcal{H}_i and \mathcal{H}_j where its inner product is defined by $\langle \cdot, \cdot \rangle_{\mathcal{H}_i} + \langle \cdot, \cdot \rangle_{\mathcal{H}_j}$ and let $\{\phi_i^{(j)}\} \subset \mathcal{H}_j$ denote a mapped training set associated with the j^{th} base kernel. Then we can view $\phi_i^k = (\sqrt{\alpha_1} \phi_i^{(1)}, \sqrt{\alpha_2} \phi_i^{(2)}) \in \mathcal{H}_1 \oplus \mathcal{H}_2$ since

$$\begin{aligned} \langle \phi_i^k, \phi_j^k \rangle &= k(\mathbf{x}_i, \mathbf{x}_j) = \alpha_1 k_1(\mathbf{x}_i, \mathbf{x}_j) + \alpha_2 k_2(\mathbf{x}_i, \mathbf{x}_j) \\ &= \langle \sqrt{\alpha_1} \phi_i^{(1)}, \sqrt{\alpha_1} \phi_j^{(1)} \rangle_{\mathcal{H}_1} + \langle \sqrt{\alpha_2} \phi_i^{(2)}, \sqrt{\alpha_2} \phi_j^{(2)} \rangle_{\mathcal{H}_2} \\ &= \left\langle \left(\sqrt{\alpha_1} \phi_i^{(1)}, \sqrt{\alpha_2} \phi_i^{(2)} \right), \left(\sqrt{\alpha_1} \phi_j^{(1)}, \sqrt{\alpha_2} \phi_j^{(2)} \right) \right\rangle_{\mathcal{H}_1 \oplus \mathcal{H}_2}. \end{aligned}$$

Similarly, we can also view $\phi_i^{k'} = (\phi_i^{(1)}, \phi_i^{(2)}) \in \mathcal{H}_1 \oplus \mathcal{H}_2$. Let I_j be the identity map in \mathcal{H}_j . Then,

$$B = \begin{bmatrix} \sqrt{\alpha_1} I_1 & 0 \\ 0 & \sqrt{\alpha_2} I_2 \end{bmatrix}.$$

Since $\infty > \alpha_1, \alpha_2 > 0$ and B is bounded (the operator norm of B is $\max(\sqrt{\alpha_1}, \sqrt{\alpha_2})$), B is invertible. \square

Now suppose we apply the kernel $k(\cdot, \cdot) = \sum_i \alpha_i k_i(\cdot, \cdot)$ obtained from the kernel alignment method to a Mahalanobis distance learner and an optimal transformation A^* is returned. Let $f(\cdot)$ be an objective function which depends only

on an inner product $\langle A\phi_i, A\phi_j \rangle$ (as assumed in Theorems 1 and 2). Since, from Proposition 4, $\langle A^*\phi_i^k, A^*\phi_j^k \rangle = \langle A^*B\phi_i^{k'}, A^*B\phi_j^{k'} \rangle$, we have

$$f^* \equiv f(\{\langle A^*\phi_i^k, A^*\phi_j^k \rangle\}) = f(\{\langle A^*B\phi_i^{k'}, A^*B\phi_j^{k'} \rangle\}).$$

Thus, by applying a training set $\{\phi_i^{k'}\}$ to a learner who tries to minimize $f(\cdot)$, a learner will return a linear map with the objective value less than or equal to f^* (because the learner can at least return A^*B). Notice that because B is invertible, the value f^* is in fact optimal. Consequently, the following claim can be stated: “there is no need to apply the methods which learn $\{\alpha_i\}$, e.g. the kernel alignment method, *at least in theory*, because learning with a simple kernel $k'(\cdot, \cdot)$ also results in a linear map having the same optimal objective value”. However, *in practice*, there can be some differences between using the two kernels $k(\cdot, \cdot)$ and $k'(\cdot, \cdot)$ due to the following reasons.

- **Existence of a local solution.** As some optimization problems are not convex, there is no guarantee that a solver is able to discover a global solution within a reasonable time. Usually, a learner discovers only a local solution, and hence two learners based on $k(\cdot, \cdot)$ and $k'(\cdot, \cdot)$ will not give the same solution. KNCA belongs to this case.

- **Non-existence of the unique global solution.** In some optimization problems, there can be many different linear maps having the same optimal values f^* , and hence there is no guarantee that two learners based on $k(\cdot, \cdot)$ and $k'(\cdot, \cdot)$ will give the same solution. KLMNN is an example of this case.

- **Size constraints.** Because of a size constraint such as $AA^T = I$ used in KDNE, our arguments used in the previous subsection cannot be applied, i.e., given that $A^*A^{*T} = I$, there is no guaranteed that $(A^*B)(A^*B)^T = I$. Hence, A^*B may not be an optimal solution of a learner based on $k'(\cdot, \cdot)$.

- **Preprocessing of target neighbors.** The behavior of some learners depends on their preprocesses. For example, before learning takes place, the KLMNN and KDNE algorithms have to specify the target neighbors of each point (by specifying a value of w_{ij}). In a case of using the KPCA trick, this specification is based

on the Euclidean distance with respect to a selected kernel (see Subsection 3.5.2 and Proposition 2). In this case, the Euclidean distance with respect to an aligned kernel $k(\cdot, \cdot)$ (which already employs some information of a training set) is more appropriate than the Euclidean distance with respect to an unweighted kernel $k'(\cdot, \cdot)$.

- **Zero coefficients.** In the above proposition we assume $\alpha_i > 0$ for all i . Often, the alignment algorithm returns $\alpha_i = 0$ for some i . Define A^* and f^* as above. Following the same line of the proof of Proposition 4, in the cases that the alignment method gives $\alpha_i = 0$ for some i , it can be easily shown that a learner with a kernel $k'(\cdot, \cdot)$ will return a linear map with its objective value better than or equal to f^* . Nevertheless, note that sometimes a better value of an objective function can lead to overfitting.

Since constructing $k'(\cdot, \cdot)$ is extremely easy, $k'(\cdot, \cdot)$ is a very attractive choice to be used in kernelized algorithms.

4.3 Numerical Experiments

On page 8 of the LMNN paper (Weinberger et al., 2006), Weinberger et al. gave a comment about KLMNN: ‘*as LMNN [with a further application of kNN] already yields highly nonlinear decision boundaries in the original input space, however, it is not obvious that “kernelizing” the algorithm will lead to significant further improvement*’. Here, before giving experimental results, we explain why “kernelizing” the algorithm can lead to significant improvements. The main intuition behind the kernelization of “Mahalanobis distance learners for the kNN classification algorithm” lies in the fact that non-linear boundaries produced by kNN (with or without Mahalanobis distance) is usually helpful for problems with multi-modalities; however, the non-linear boundaries of kNN is sometimes not helpful when data of the same class stay on a low-dimensional non-linear manifold as shown in Figure 4.1.

In this section, we conduct experiments on NCA, LMNN, DNE and their kernel versions on nine real-world datasets to show that (1) it is really the case that the kernelized algorithms usually outperform their original versions on real-world

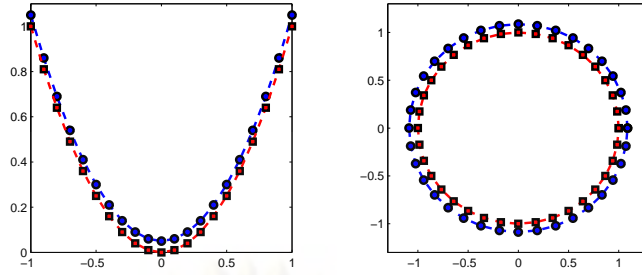


Figure 4.1: Two synthetic examples where NCA, LMNN and DNE cannot learn any efficient Mahalanobis distances for kNN. Note that in each example, data in each class lie on a simple non-linear 1-dimensional subspace (which, however, cannot be discovered by the three learners). In contrast, the kernel versions of the three algorithms (using the 2^{nd} -order polynomial kernel) can learn very efficient distances, i.e., the non-linear subspaces can be discovered by the kernelized algorithms.

Table 4.1: The average accuracy with standard deviation of NCA and their kernel versions. On the bottom row, the win/draw/lose statistics of each kernelized algorithm compared to its original version is drawn.

NAME	NCA	KNCA	AKNCA	UKNCA
BALANCE	0.89 \pm 0.03	0.92 \pm 0.01	0.92 \pm 0.01	0.91 \pm 0.03
BREAST CANCER	0.95 \pm 0.01	0.97 \pm 0.01	0.96 \pm 0.01	0.96 \pm 0.02
GLASS	0.61 \pm 0.05	0.69 \pm 0.02	0.69 \pm 0.04	0.68 \pm 0.04
IONOSPHERE	0.83 \pm 0.04	0.94 \pm 0.03	0.92 \pm 0.02	0.90 \pm 0.03
IRIS	0.96 \pm 0.03	0.96 \pm 0.01	0.95 \pm 0.03	0.96 \pm 0.02
MUSK2	0.87 \pm 0.02	0.90 \pm 0.01	0.88 \pm 0.02	0.87 \pm 0.02
PIMA	0.68 \pm 0.02	0.71 \pm 0.02	0.67 \pm 0.03	0.69 \pm 0.01
SATELLITE	0.82 \pm 0.02	0.84 \pm 0.01	0.84 \pm 0.01	0.82 \pm 0.02
YEAST	0.47 \pm 0.02	0.50 \pm 0.01	0.49 \pm 0.02	0.47 \pm 0.02
WIN/DRAW/LOSE	-	8/1/0	7/0/2	5/4/0

datasets, and (2) the performance of linearly combined kernels achieved by the two methods presented in this chapter are acceptable compared to kernels which are exhaustively selected, but the exhaustive selection method requires much more running time.

To measure the generalization performance of each algorithm, we use the nine real-world datasets obtained from the UCI repository (Asuncion and Newman, 2007): BALANCE, BREAST CANCER, GLASS, IONOSPHERE, IRIS, MUSK2, PIMA, SATELLITE and YEAST. Following previous works, we randomly divide each dataset into training and testing sets. By repeating the process 40 times, we have 40 training and testing sets for each dataset. The generalization performance

Table 4.2: The average accuracy with standard deviation of LMNN and their kernel versions.

NAME	LMNN	KLMNN	AKLMNN	UKLMNN
BALANCE	0.84 ± 0.04	0.87 ± 0.01	0.88 ± 0.02	0.85 ± 0.01
BREAST CANCER	0.95 ± 0.01	0.97 ± 0.01	0.97 ± 0.00	0.97 ± 0.00
GLASS	0.63 ± 0.05	0.69 ± 0.04	0.69 ± 0.04	0.66 ± 0.05
IONOSPHERE	0.88 ± 0.02	0.95 ± 0.02	0.94 ± 0.02	0.94 ± 0.02
IRIS	0.95 ± 0.02	0.96 ± 0.02	0.95 ± 0.02	0.97 ± 0.01
MUSK2	0.80 ± 0.03	0.93 ± 0.01	0.88 ± 0.02	0.86 ± 0.02
PIMA	0.68 ± 0.02	0.71 ± 0.02	0.72 ± 0.02	0.67 ± 0.03
SATELLITE	0.81 ± 0.01	0.85 ± 0.01	0.84 ± 0.01	0.83 ± 0.02
YEAST	0.47 ± 0.02	0.48 ± 0.02	0.54 ± 0.02	0.50 ± 0.02
WIN/DRAW/LOSE	-	9/0/0	8/1/0	8/0/1

Table 4.3: The average accuracy with standard deviation of DNE and their kernel versions.

NAME	DNE	KDNE	AKDNE	UKDNE
BALANCE	0.79 ± 0.02	0.90 ± 0.01	0.83 ± 0.02	0.85 ± 0.03
BREAST CANCER	0.96 ± 0.01	0.97 ± 0.01	0.96 ± 0.01	0.96 ± 0.02
GLASS	0.65 ± 0.04	0.70 ± 0.03	0.69 ± 0.04	0.65 ± 0.03
IONOSPHERE	0.87 ± 0.02	0.95 ± 0.02	0.95 ± 0.02	0.93 ± 0.03
IRIS	0.95 ± 0.02	0.97 ± 0.02	0.96 ± 0.02	0.96 ± 0.03
MUSK2	0.89 ± 0.02	0.91 ± 0.01	0.89 ± 0.02	0.84 ± 0.03
PIMA	0.67 ± 0.02	0.69 ± 0.02	0.70 ± 0.03	0.70 ± 0.02
SATELLITE	0.84 ± 0.01	0.85 ± 0.01	0.85 ± 0.01	0.81 ± 0.02
YEAST	0.40 ± 0.05	0.48 ± 0.01	0.47 ± 0.04	0.52 ± 0.02
WIN/DRAW/LOSE	-	9/0/0	7/2/0	5/2/2

of each algorithm is then measured by the average test accuracy over the 40 testing sets of each dataset. The number of training data is 200 except for GLASS and IRIS where we use 100 examples because these two datasets contain only 214 and 150 total examples, respectively.

Following previous works, we use the 1NN classifier in all experiments. In order to kernelize the algorithms, three approaches are applied to select appropriate kernels:

- Cross validation (KNCA, KLMNN and KDNE).
- Kernel alignment (AKNCA, AKLMNN and AKDNE).
- Unweighted combination of base kernels (UKNCA, UKLMNN and UKDNE).

For all three methods, we consider scaled RBF base kernels (Schölkopf and Smola, 2001, p. 216), $k(x, y) = \exp(-\frac{\|x-y\|^2}{2D\sigma^2})$ where D is the dimensionality of input data. Twenty one based kernels specified by the following values of σ are considered: 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1, 2.5, 5, 7.5, 10, 25, 50, 75, 100, 250, 500, 750, 1000. All kernelized algorithms are implemented by the KPCA trick illustrated in Figure 3.3. As noted in Subsection 4.2, the main problem of using the unweighted kernel to algorithms such as UKLMNN and UKDNE is that the Euclidean distance with respect to the unweighted kernel is not informative and thus should not be used to specify target neighbors of each point. Therefore, in cases of UKLMNN and UKDNE, we employ the Euclidean distance with respect to the input space to specify target neighbors. We slightly modify the original codes of LMNN and DNE to fulfill this desired specification. YALMIP toolbox is applied for implementing convex programs (Loefberg, 2004).

The experimental results are shown in Tables 4.1, 4.2 and 4.3. From the results, it is clear that the kernelized algorithms usually improve the performance of their original algorithms. The kernelized algorithms applying cross validation obtain the best performance. They outperform the original methods in 26 out of 27 datasets. The other two kernel versions of the three original algorithms also have satisfiable performance. The kernelized algorithms applying kernel alignment outperform the original algorithms in 22 datasets and obtain an equal performance in 3 datasets. Only 2 out of 27 datasets where the original algorithms outperform the kernel algorithms applying kernel alignment. Similarly, the kernelized algorithms applying the unweighted kernel outperform the original algorithms in 18 datasets and obtain an equal performance in 6 datasets. Only 3 out of 27 datasets where the original algorithms outperform the kernel algorithms applying the unweighted kernel.

We note that although the cross validation method usually gives the best performance, the other two kernel construction methods provide acceptable performance in much shorter running time. In fact, the alignment method provides comparable performance. For each dataset, a run-time overhead of the kernelized algorithms applying cross validation is of several hours (on Pentium IV 1.5GHz,

Ram 1 GB) while run-time overheads of the kernelized algorithms applying aligned kernels and the unweighted kernel are about minutes and seconds, respectively, for each dataset. Therefore, in time-limited circumstance, it is attractive to apply an aligned kernel or an unweighted kernel.

Note that the kernel alignment method is not appropriate for a multi-modal dataset in which there may be several clusters of data points for each class since, from Eq. (4.1), the function $\text{align}(K, Y)$ will attain the maximum value if and only if all points of the same class are collapsed into a single point. This may be one reason which explains why cross validated kernels give better results than results of aligned kernels in our experiments. Developing a new kernel alignment algorithm which is suitable for multi-modality is currently an open problem.

Comparing generalization performance induced by aligned kernels and the unweighted kernel, we found that algorithms applying aligned kernels perform slightly better than algorithms applying the unweighted kernel. With little overhead and satisfiable performance, however, the unweighted kernel is still attractive for algorithms, like NCA (in contrast to LMNN and DNE), which are not required a specification of target neighbors w_{ij} . Since Euclidean distance with respect to the unweighted kernel is usually not appropriate for specifying w_{ij} , a KPCA trick application of algorithms like LMNN and DNE may still require some re-programming.

As noted in the previous section, aligned kernels usually do not use all base kernels ($\alpha_i = 0$ for some i); in contrast, the unweighted kernel uses all base kernels ($\alpha_i = 1$ for all i). Hence, as described in Section 4.2, the feature space corresponding to the unweighted kernel usually contains the feature space corresponding to aligned kernels. Therefore, we may informally say that the feature space induced by the unweighted kernel is “larger” than one induced by the aligned kernel.

Since a feature space which is too large can lead to overfitting, one may wonder whether or not using the unweighted kernel leads to overfitting. Figure 4.2 shows that overfitting indeed does not occur. For compactness, we show only the results of UKDNE. In the experiments shown in this figure, base kernels are adding in the following order: 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1,

2.5, 5, 7.5, 10, 25, 50, 75, 100, 250, 500, 750, 1000. It can be observed from the figure that the generalization performance of UKDNE is consistently non-decreasing and will be eventually stable as we add more and more base kernels. Also, It can be observed that 10 - 14 base kernels are enough to obtain stable performance. It is interesting to further investigate an overfitting behavior of a learner by applying methods such as a *bias-variance analysis* (James, 2003) or *zone analysis* (Chatpatanasiri et al., 2009) and investigate whether it is appropriate or not to apply an “adaptive resampling and combining” method (Breiman, 1998) to improve the classification performance of a supervised mahalanobis distance learner.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

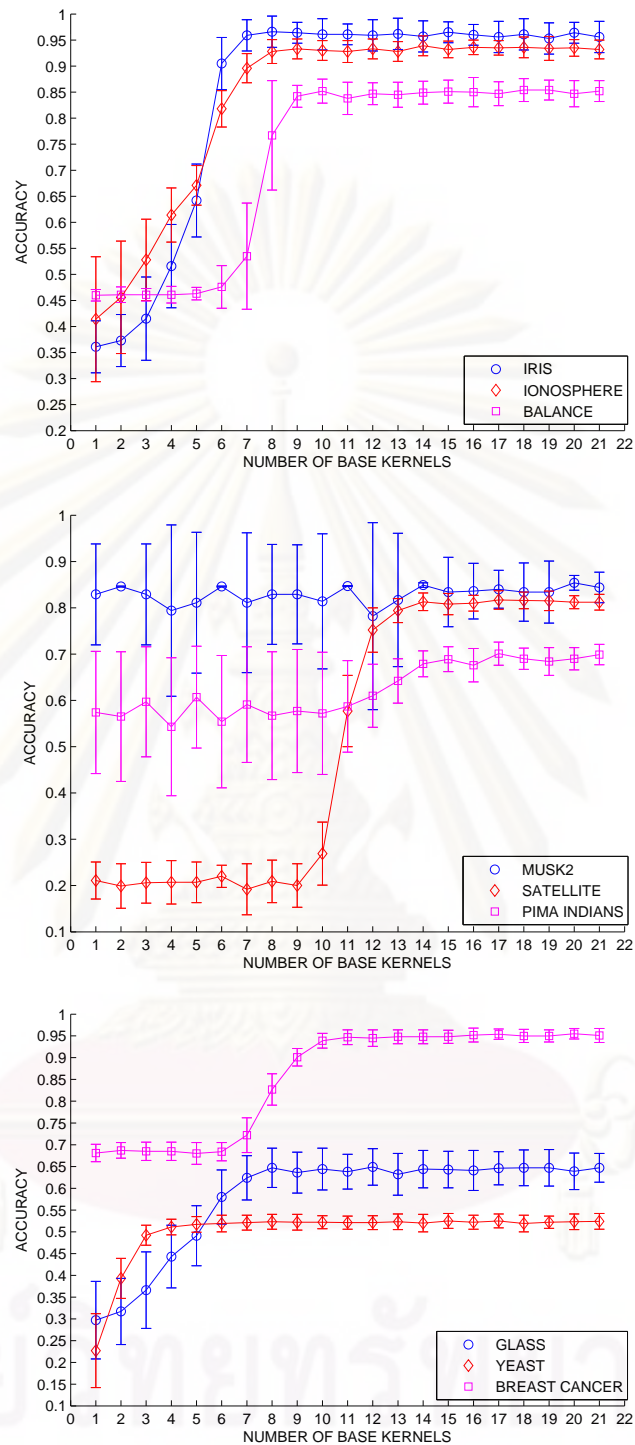


Figure 4.2: This figure illustrates performance of UKDNE with different numbers of base kernels. It can be observed from the figure that the generalization performance of UKDNE is consistently non-decreasing and will be eventually stable as we add more and more base kernels.

CHAPTER V

SPECTRAL SEMI-SUPERVISED LEARNING FRAMEWORK: THEORY

In this chapter, a semi-supervised learning framework is developed for a specific class of Mahalanobis distance learners, namely, the class of spectral linear dimensionality reduction algorithms. The framework naturally generalizes existing supervised, unsupervised and semi-supervised learning frameworks which apply the spectral decomposition. Algorithms derived under our framework are able to employ both labeled and unlabeled examples and are able to handle complex problems where data form separate clusters of manifolds. Our framework offers simple views, explains relationships among existing frameworks and provides further extensions which can improve existing algorithms. The KPCA trick extended to semi-supervised learning frameworks is also presented.

5.1 Introduction

In many real-world applications, high-dimensional data indeed lie on (or near) a low-dimensional subspace. The goal of *dimensionality reduction* is to reduce complexity of input data while some desired intrinsic information of the data is preserved. The desired information can be discriminative (Yan et al., 2007; Zhang et al., 2007b; Cai et al., 2007; Hoi et al., 2006; Chen et al., 2005; Cheng et al., 2004), geometrical (Tenenbaum et al., 2000; Roweis and Saul, 2000; He and Niyogi, 2004; Saul et al., 2006) or both (Sugiyama, 2007). There are several advantages of reducing the dimensionality of input data. First, working on a low-dimensional space significantly saves both time and storage. Second, an intuitive visualization is possible for low-dimensional data. Finally, and most importantly, working on a low-dimensional subspace secures us from the *curse of dimensionality* (Bishop, 2006); “learning” is possible even if we have only a relatively few number of input data. Note that, as described in Chapter 2, linear dimensionality reduction is a special case of Mahalanobis distance learning.

Fisher discriminant analysis (FDA), introduced in Chapter 2, is the most

popular method among all supervised dimensionality reduction algorithms. Denote c as the number of classes in a given training set. Provided that training examples of each class lie in a linear subspace and *do not* form several separate clusters, i.e. *do not* form multi-modality, FDA is able to discover a low-dimensional linear subspace (with at most $c - 1$ dimensionality) which is efficient for classification. Recently, many works have improved the FDA algorithm in several aspects (Sugiyama, 2007; Yan et al., 2007; Zhang et al., 2007b; Cai et al., 2007; Hoi et al., 2006; Chen et al., 2005; Cheng et al., 2004). These *extended FDA algorithms* are able to discover a nice low-dimensional subspace even when training examples of each class lie in separate clusters of complicated non-linear manifolds. Moreover, a subspace discovered by these algorithms has no limitation of $c - 1$ dimensionality.

Although the extended FDA algorithms work reasonably well, a considerable number of labeled examples is required to achieve satisfiable performance. In many real-world applications such as image classification, web page classification and protein function prediction, a labeling process is costly and time consuming; in contrast, unlabeled examples can be easily obtained. Therefore, in such situations, it can be beneficial to incorporate the information which is contained in unlabeled examples into a learning problem, i.e., semi-supervised learning (SSL) should be applied instead of supervised learning (Chapelle et al., 2006).

One may have a question: why are unlabeled examples useful in supervised learning? In fact, the whole research on clustering devotes itself to the problem of constructing a good classifier from unlabeled examples. As the research on clustering have shown promising results, we can expect that unlabeled examples can help a learner improve the quality of a constructed hypothesis. Figure 5.1 illustrates an example of clustering.

In this chapter, we present a general semi-supervised dimensionality reduction framework which is able to employ information from both labeled and unlabeled examples. As the extended FDA algorithms, algorithms developed in our framework are able to discover a nice low-dimensional subspace even when training examples of each class form separate clusters of complicated non-linear manifolds. In fact, those previous supervised algorithms can be casted as instances in our

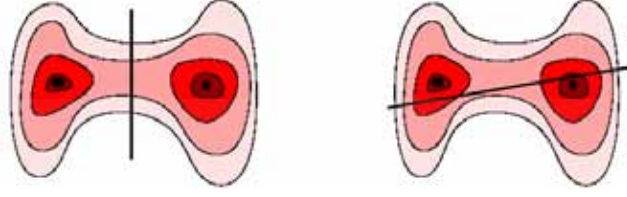


Figure 5.1: Given a contour of data (higher a number of data, darker a color), a rationale human can easily find a likely partition as one shown in the left and an unlikely partition similar to one shown in the right.

framework. Moreover, recent existing semi-supervised frameworks known to us (Li et al., 2007; Sugiyama et al., 2008; Song et al., 2008) can be viewed as special cases of our framework as well.

5.2 Spectral Semi-Supervised Learning Framework

Let $\{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$ denote a training set of ℓ labeled examples, with inputs $\mathbf{x}_i \in \mathbb{R}^{d_0}$ generated from a fixed but unknown probability distribution $\mathcal{P}_{\mathbf{x}}$, and corresponding class labels $y_i \in \{1, \dots, c\}$ generated from $\mathcal{P}_{y|\mathbf{x}}$. In addition to the labeled examples, let $\{\mathbf{x}_i\}_{i=\ell+1}^{\ell+u}$ denote a set of u unlabeled examples also generated from $\mathcal{P}_{\mathbf{x}}$. We define the following goal of SSL dimensionality reduction.

Goal. Using the information of both labeled and unlabeled examples, we want to embed an input space into a low-dimensional space, i.e. we want to map $(\mathbf{x} \in \mathbb{R}^{d_0}) \mapsto (\mathbf{z} \in \mathbb{R}^d)$ where $d < d_0$, such that in the embedded space $\mathcal{P}_{y|\mathbf{z}}$ can be accurately estimated (i.e., unknown labels are easy to predict) by a *simple classifier*.

Here, following the previous works in the supervised setting (Sugiyama, 2007; Yan et al., 2007; Zhang et al., 2007b), the *nearest neighbor* algorithm is used for representing a simple classifier mentioned in the goal. Note that important special cases of SSL problems are *transductive* problems where we only want to predict the labels $\{y_i\}_{i=\ell+1}^{\ell+u}$ of the given unlabeled examples.

In order to make use of unlabeled examples in the learning process, we make

the following so-called *manifold assumption*, which is proven to be applicable in many real-world data (Chapelle et al., 2006):

Semi-Supervised Manifold Assumption. The support of $\mathcal{P}_{\mathbf{x}}$ is on a low-dimensional manifold. Furthermore, $\mathcal{P}_{y|\mathbf{x}}$ is smooth, as a function of \mathbf{x} , with respect to the underlying structure of the manifold.

In words, this assumption states that two nearby points on a high-density region of $\mathcal{P}_{\mathbf{x}}$ are likely to be in the same class where a high-density region is of a low-dimensional manifold structure. Since unlabeled examples can be used to estimate a high-density region of $\mathcal{P}_{\mathbf{x}}$, they are also useful to predict the label of an example¹.

5.2.1 The Framework

Let $Z^* = (\mathbf{z}_1, \dots, \mathbf{z}_n) \in \mathbb{R}^{d \times n}$, where $n = \ell + u$, be the matrix of desired embedded points. In our framework, we propose to cast the problem as a constrained optimization problem:

$$Z^* = \arg \min_{Z \in \mathcal{Z}} f^\ell(Z) + \gamma f^u(Z), \quad (5.1)$$

where $f^\ell(\cdot)$ is an objective function which is based on label information, $f^u(\cdot)$ is an objective function based on unlabel information, γ is a parameter which controls the weights between the two objective functions and \mathcal{Z} is a constraint set in $\mathbb{R}^{d \times n}$. Up to *orthogonal and translational transformations*, we can identify embedded points via their pairwise distances instead of their individual locations². Therefore, we can base the objective functions on pairwise distances of embedded examples. Here, we define the objective functions to be *linear* with respect to

¹In fact, beside the manifold assumption, here we make an additional assumption that available unlabeled examples are generated from a high-density region of $\mathcal{P}_{\mathbf{x}}$. Nevertheless, this additional assumption is not too strong as it is implied by the law of large number.

²As noted above, for all Mahalanobis distance learners considered in this dissertation, kNN will be applied in the embedded space. kNN is translational and orthogonal invariance and thus depends only on pairwise distances of examples.

pairwise distances:

$$f^\ell(Z) = \sum_{i,j=1}^n c_{ij}^\ell \text{dist}(\mathbf{z}_i, \mathbf{z}_j), \quad f^u(Z) = \sum_{i,j=1}^n c_{ij}^u \text{dist}(\mathbf{z}_i, \mathbf{z}_j),$$

where $\text{dist}(\cdot, \cdot)$ is an arbitrary distance function between two embedded points, c_{ij}^ℓ and c_{ij}^u are costs which penalize an embedded distance between two points i and j . A specification of c_{ij}^ℓ is based on the label information and a specification of c_{ij}^u is based on the unlabel information as described and interpreted later in Section 5.2.3.

If we restrict ourselves to consider only the cases that (I) $\text{dist}(\cdot, \cdot)$ is a squared Euclidean distance function, i.e. $\text{dist}(\mathbf{z}_i, \mathbf{z}_j) = \|\mathbf{z}_i - \mathbf{z}_j\|^2$, (II) c_{ij}^ℓ and c_{ij}^u are symmetric, and (III) $Z \in \mathcal{Z}$ is in the form of $ZBZ^T = I$ where B is a positive semidefinite (PSD) matrix, Eq.(5.1) will result in a general framework which indeed generalizes previous spectral-method frameworks as shown later in Section 5.3. Define $c_{ij} = c_{ij}^\ell + \gamma c_{ij}^u$. We then can rewrite the weighted combination of the objective functions in Eq. (5.1) as follows:

$$\begin{aligned} f^\ell(Z) + \gamma f^u(Z) &= \sum_{i,j=1}^n c_{ij}^\ell \text{dist}(\mathbf{z}_i, \mathbf{z}_j) + \gamma \sum_{i,j=1}^n c_{ij}^u \text{dist}(\mathbf{z}_i, \mathbf{z}_j) \\ &= \sum_{i,j=1}^n (c_{ij}^\ell + \gamma c_{ij}^u) \text{dist}(\mathbf{z}_i, \mathbf{z}_j) = \sum_{i,j=1}^n c_{ij} \text{dist}(\mathbf{z}_i, \mathbf{z}_j) \\ &= \sum_{i,j=1}^n c_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|^2 = 2 \sum_{i,j=1}^n c_{ij} (\mathbf{z}_i^T \mathbf{z}_i - \mathbf{z}_i^T \mathbf{z}_j) \\ &= 2 \text{trace} \left(\sum_{i,j=1}^n (\mathbf{z}_i c_{ij} \mathbf{z}_i^T) - \sum_{i,j=1}^n (\mathbf{z}_i c_{ij} \mathbf{z}_j^T) \right) \\ &= 2 \text{trace} \left(\sum_{i=1}^n \mathbf{z}_i \left(\sum_{j=1}^n c_{ij} \right) \mathbf{z}_i^T - \sum_{i,j=1}^n (\mathbf{z}_i c_{ij} \mathbf{z}_j^T) \right) \\ &= 2 \text{trace}(Z(D - C)Z^T), \end{aligned}$$

where C is a symmetric cost matrix with elements c_{ij} and D is a diagonal matrix

with $D_{ii} = \sum_j c_{ij}$ ³. Thus, the optimization problem (5.1) can be restated as

$$Z^* = \arg \min_{ZBZ^T=I} \text{trace}(Z(D - C)Z^T). \quad (5.2)$$

Note that the constraint $ZBZ^T = I$ prevents trivial solutions such as every \mathbf{z}_i is a zero vector. If B is a positive definite (PD) matrix, a solution of the above problem is given by the bottom d eigenvectors of the following generalized eigenvalue problem (Fukunaga, 1990; von Luxburg, 2007)

$$(D - C)\mathbf{z}^{(j)} = \lambda_j B\mathbf{z}^{(j)}, \quad j = 1, \dots, d. \quad (5.3)$$

Then we have the optimal embedded points represented by

$$Z^* = (\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(d)})^T. \quad (5.4)$$

Note that, in terms of solutions of Eq.(5.3), it is more convenient to represent Z by its rows $\mathbf{z}^{(i)} \in \mathbb{R}^n$ than its columns $\mathbf{z}_i \in \mathbb{R}^d$.

5.2.2 Linear Parameterization

Notice that, in fact, the optimal solution Z^* obtained from Eq.(5.4) does not completely solve our goal because although the map $\mathbf{x}_i \mapsto \mathbf{z}_i$ is given for each training point, a map $\mathbf{x}' \mapsto \mathbf{z}'$ for an unseen point \mathbf{x}' is unknown. Hence, $\|\mathbf{z}' - \mathbf{z}_i\|$ cannot be computed, and kNN classification in the obtained subspace cannot be performed⁴. We propose to resolve this difficulty by parameterizing $\{\mathbf{z}_i\}$. One of the simplest parameterization approaches is the “linear” restriction of a map $\mathbf{x}_i \mapsto \mathbf{z}_i$ such that $\mathbf{z}_i = A\mathbf{x}_i$ where $A \in \mathbb{R}^{d \times d_0}$. A “non-linear” extension of this idea will be later presented in Section 5.2.4. From the linear parameterization, we have $Z = AX$ where $X \in \mathbb{R}^{d_0 \times n}$ is a matrix of the input examples $(\mathbf{x}_1, \dots, \mathbf{x}_n)$. Now, the original problem in Eq.(5.1) is changed to a problem of finding a *linear transformation* which minimizes a cost function. This new problem can be formally

³To simplify our notations, in this chapter whenever we define a cost matrix C' having elements c'_{ij} , we always define its associated diagonal matrix D' with elements $D'_{ii} = \sum_j c'_{ij}$.

⁴If we do not concern about an unseen example, e.g. in cases that we work in the transductive setting, the optimal solution Z^* is sufficient.

stated as follows:

$$A^* = \arg \min_{ABA^T=I} \sum_{i,j=1}^n c_{ij} \|A\mathbf{x}_i - A\mathbf{x}_j\|^2 \quad (5.5)$$

which, following the same lines of derivation of Eq.(5.2), can be restated as

$$A^* = \arg \min_{ABA^T=I} \text{trace}(AX(D - C)X^T A^T). \quad (5.6)$$

If B is PD (see Remark 2. in Section 5.2.5), its solution is provided by solving

$$X(D - C)X^T \mathbf{a}^{(j)} = \lambda_j B \mathbf{a}^{(j)}, \quad j = 1, \dots, d \quad (5.7)$$

where $A^* = (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)})^T$. Moreover,

$$\|\mathbf{z} - \mathbf{z}'\| = \|A^* \mathbf{x} - A^* \mathbf{x}'\|, \quad (5.8)$$

so that kNN in the embedded space can be performed. Consequently, an algorithm implemented under our framework consists of three steps as shown in Figure 5.2.

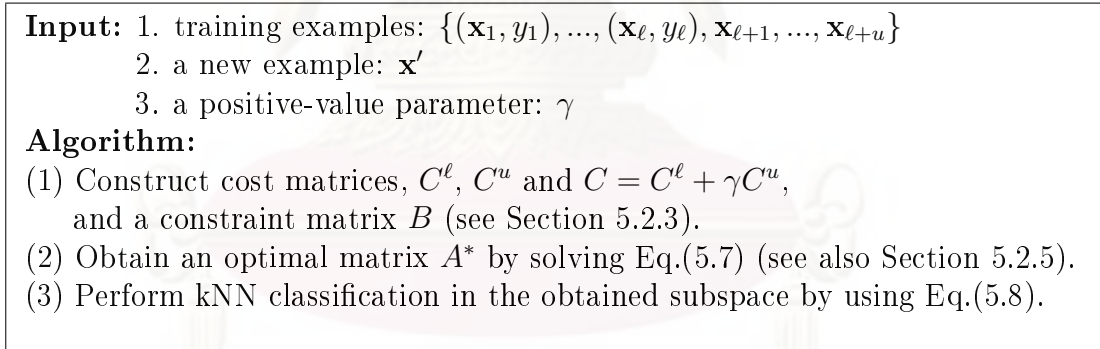


Figure 5.2: Our semi-supervised learning framework.

5.2.3 Specification of the Cost and Constraint Matrices

In this section, we present various reasonable approaches of specifying the two cost matrices, C^ℓ and C^u , and the constraint matrix, B , by using the label and unlabel information. There are two important types of unlabel information (Zhang, 2003), *neighborhood information* and *side information*. Here, we are not interested in the side information, i.e. the information of a similarity between each pair of examples, since the side information is usually not provided in real-world

problems. Thus, we use the two words “unlabel information” and “neighborhood information” interchangeably in this chapter.

5.2.3.1 The Cost Matrix C^ℓ and the Constraint Matrix B

Normally, based on the label information, classical supervised algorithms usually require an embedded space to have the following two desirable conditions:

- (1) two examples of the same class stay close to one another, and
- (2) two examples of different classes stay far apart.

The two conditions are imposed in classical works such as FDA. However, the first condition is too restrictive to capture manifold and multi-modal structures of data which naturally arise in some applications. Thus, the first condition should be relaxed as follows.

- (1*) two *nearby examples* of the same class stay close to one another

where “nearby examples”, defined by using the neighborhood information, are examples which should stay close to each other in both original and embedded spaces. The specification of “nearby examples” has been proven to be successful in discovering manifold and multi-modal structure (Sugiyama, 2007; Yan et al., 2007; Zhang et al., 2007b; Cai et al., 2007; Hoi et al., 2006; Chen et al., 2005; Cheng et al., 2004; Goldberger et al., 2005; Globerson and Roweis, 2006; Weinberger et al., 2006; Yang et al., 2006a; Torresani and Lee, 2007). See Figure 5.3 for explanations. In some cases, it is also appropriate to relax the second condition to

- (2*) two *nearby examples* of different classes stay far apart.

In this section, we give three examples of cost matrices which satisfy the

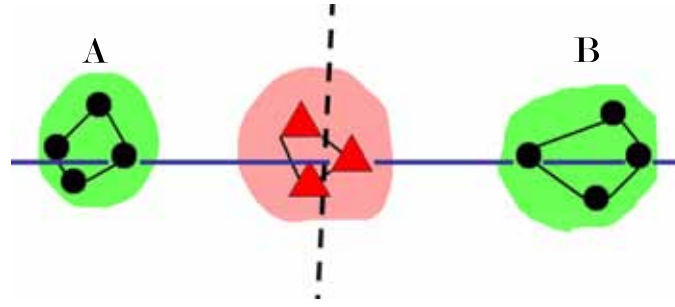


Figure 5.3: An example when data form a multi-modal structure. An algorithm, e.g. FDA, which imposes the condition (1) will try to discover a new subspace (a dashed line) which merges two clusters **A** and **B** altogether. An obtained space is undesirable as data of the two classes are mixed together. In contrast, an algorithm which imposes the condition (1*) (instead of (1)) will discover a subspace (a thick line) which does not merge the two clusters **A** and **B** as there are no nearby examples (indicated by a link between a pair of examples) between the two clusters.

conditions (1*) and (2) (or (2*)). These three examples are recently introduced in previous works, namely, Discriminant Neighborhood Embedding (DNE) (Zhang et al., 2007b), Marginal Fisher Analysis (MFA) (Yan et al., 2007) and Local Fisher Discriminant Analysis (LFDA) (Sugiyama, 2007), with different presentations and motivations but they can be unified under our general framework.

Firstly, to utilize neighborhood information, we construct two matrices C^I and C^E based on Euclidean distance⁵. For each \mathbf{x}_i , let $Neig^I(i)$ be the set of k nearest neighbors having the *same* label y_i , and let $Neig^E(i)$ be the set of k nearest neighbors having *different* labels from y_i . Define C^I and C^E as follows⁶: let $c_{ij}^I = c_{ij}^E = 0$ if points \mathbf{x}_i and/or \mathbf{x}_j are unlabeled, and

$$c_{ij}^I = \begin{cases} 1, & \text{if } j \in Neig^I(i) \vee i \in Neig^I(j), \\ 0, & \text{otherwise, and} \end{cases}$$

$$c_{ij}^E = \begin{cases} 1, & \text{if } j \in Neig^E(i) \vee i \in Neig^E(j), \\ 0, & \text{otherwise.} \end{cases}$$

Often, $k \ll \ell$ is applied in order to make C^ℓ sparse so that, hopefully, all involved computations are efficient. By utilizing the neighborhood information, the speci-

⁵Any distance functions which are sensible to a given problem can be applied in place of Euclidean distance.

⁶In fact, specifications of C^I and C^E presented here are one of the simplest possibilities and can be combined with the heat kernel described in the next subsection.

fication $c_{ij}^I = 1$ and $c_{ij}^E = 1$ represent nearby examples in the conditions (1*) and (2*). Next, C^ℓ and B of existing algorithms (Eq. (5.6) and Eq. (5.7)) are:

Discriminant Neighborhood Embedding (DNE)

$$C^\ell = C^I - C^E \quad B = I \text{ (an identity matrix)}$$

Marginal Fisher Analysis (MFA)

$$C^\ell = -C^E \quad B = X(D^I - C^I)X^T$$

Local Fisher Discriminant Analysis (LFDA)

Let n_1, \dots, n_c be the numbers of examples of classes $1, \dots, c$, respectively. Define matrices C^{bet} and C^{wit} as:

$$c_{ij}^{bet} = \begin{cases} c_{ij}^I \left(\frac{1}{n_k} - \frac{1}{n} \right), & \text{if } y_i = y_j = k, \\ -\frac{1}{n}, & \text{otherwise,} \end{cases} \quad \text{and}$$

$$c_{ij}^{wit} = \begin{cases} \frac{1}{n_k} c_{ij}^I, & \text{if } y_i = y_j = k, \\ 0, & \text{otherwise,} \end{cases}$$

$$C^\ell = C^{bet} \quad B = X(D^{wit} - C^{wit})X^T$$

Within our framework, relationships among the three previous works can be explained. The three methods exploit different ideas in specifying matrices C^ℓ and B to satisfy two desirable conditions in an embedded space. It is easy to see in the cases of DNE and MFA. In DNE, C^ℓ is designed to penalize an embedded space which does not satisfy the condition (1*) and (2*). In MFA, the constraint matrix B is designed to satisfy the condition (1*) and C^ℓ is designed to penalize an embedded space which does not satisfy the condition (2*).

Things are not quite obvious in the case of LFDA. In LFDA, the constraint matrix B is designed to satisfy the condition (1*) since elements C^{wit} are proportional to C^I ; nevertheless, since weights are inversely proportional to n_k , elements in a small class have larger weights than elements in a bigger class, i.e. a pair in

a small class is more likely to satisfy the condition (1*) than a pair in a bigger class. To understand C^ℓ , we recall that

$$\begin{aligned} \text{trace}(AX(D^\ell - C^\ell)X^T A^T) &= \sum_{i,j} c_{ij}^\ell \|A\mathbf{x}_i - A\mathbf{x}_j\| \\ &= \sum_{y_i=y_j} c_{ij}^I \left(\frac{1}{n_k} - \frac{1}{n}\right) \|A\mathbf{x}_i - A\mathbf{x}_j\| - \sum_{y_i \neq y_j} \frac{1}{n} \|A\mathbf{x}_i - A\mathbf{x}_j\| \\ &= d - \frac{1}{n} \left(\sum_{y_i=y_j} c_{ij}^I \|A\mathbf{x}_i - A\mathbf{x}_j\| + \sum_{y_i \neq y_j} \|A\mathbf{x}_i - A\mathbf{x}_j\| \right), \end{aligned}$$

where at the third equality we use the constraint $AXBX^T A^T = I$ and hence

$$\begin{aligned} \text{trace}(AX(D^{wit} - C^{wit})X^T A^T) &= \sum_{y_i=y_j} \frac{c_{ij}^I}{n_k} \|A\mathbf{x}_i - A\mathbf{x}_j\| \\ &= \text{trace}(I) = d. \end{aligned}$$

Hence, we observe that *every* pair of labeled examples coming from different classes has a corresponding cost of $-\frac{1}{n}$. Therefore, C^ℓ is designed to penalize an embedded space which does not satisfy the condition (2). Surprisingly, in LFDA, nearby examples of the *same* class (having $c_{ij}^I = 1$) also has a cost of $-\frac{1}{n}$. As a cost proportional to $-\frac{1}{n}$ is meant to preserve a pairwise distance between each pair of examples (see Section 5.3.1). Thus, in contrast to DNE and MFA which try to squeeze nearby examples of the same class to a single point, LFDA tries to preserve a local geometrical structure between each pair of nearby examples of the same class.

We note that other recent supervised methods for manifold learning can also be presented and interpreted in our framework with different specifications of C^ℓ , for examples, *Discriminant Locally Linear Embedding* of Li et al. (2008b) *Local Discriminant Embedding* of Chen et al. (2005) and *Supervised Nonlinear Local Embedding* of Cheng et al. (2004).

5.2.3.2 The Cost Matrix C^u and the Hadamard Power Operator

One important implication of the manifold assumption is that “nearby examples are likely to belong to a same class”. Hence, by the assumption, it makes

sense to design C^u such that *it prevents any pairs of nearby examples to stay far apart in an embedded space.*

Among methods of extracting the neighborhood information to define C^u , methods based on *the heat kernel* (or *the gaussian function*) are most popular. Beside using the heat kernel, other methods of defining C^u are invented, see (Chapelle et al., 2006, Chap. 15) and (von Luxburg, 2007) for more details. The simplest specifications of nearby examples based on the heat kernel are:

$$c_{ij}^u = \exp\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right). \quad (5.9)$$

Each pair of nearby examples will be penalized with different costs depended on their similarity, and a similarity between two points is based on the Euclidean distance between them in the input space. Incidentally, with this specification of C^u , the term $f^u(Z)$ in Eq. (5.1) can be interpreted as an approximation of the *Laplace-Beltrami operator* on a data manifold. A learner which employs $C = C^u$ (i.e. $C^\ell = 0$) is named *Locally Preserving Projection* (LPP) (He and Niyogi, 2004).

The parameter σ is crucial as it controls the scale of a cost c_{ij}^u . Hence, the choice of σ must be sensible. Moreover, an appropriate choice of σ may vary across the support of $\mathcal{P}_\mathbf{x}$. Hence, the *local scale* σ_i for each point \mathbf{x}_i should be used. Let \mathbf{x}'_i be the k^{th} nearest neighbor of \mathbf{x}_i . A local scale is defined as

$$\sigma_i = \|\mathbf{x}'_i - \mathbf{x}_i\|,$$

and a weight of each edge is then defined as

$$c_{ij}^u = \exp\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_i \sigma_j}\right). \quad (5.10)$$

Using this local scaling method is proven to be efficient in previous experiments (Zelnik-Manor and Perona, 2004) on clustering. A specification of k to define the local scale of each point is usually more convenient than a specification of σ since a space of possible choices of k is considerably smaller than that of σ .

Instead of proposing yet another method to specify a cost matrix, here we

present a novel method which can be used to modify any existing cost matrix. Let Q and R be two matrices of equal size and have q_{ij} and r_{ij} as their elements. Recall that the *Hadamard product* P (Schott, 2005) between Q and R , $P = Q \odot R$, has elements $p_{ij} = q_{ij}r_{ij}$. In words, the Hadamard product is a pointwise product between two matrices. Here, we define the Hadamard α^{th} power operator as

$$\overset{\alpha}{\odot} Q = \overbrace{Q \odot Q \odot \dots \odot Q}^{\alpha \text{ times}}. \quad (5.11)$$

Given a cost matrix C^u and a positive integer α , we define a new cost matrix C^{u^α} as

$$C^{u^\alpha} = \overset{\alpha}{\odot} C^u \frac{\|C^u\|_F}{\|\overset{\alpha}{\odot} C^u\|_F}, \quad (5.12)$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The multiplication of $\frac{\|C^u\|_F}{\|\overset{\alpha}{\odot} C^u\|_F}$ make $\|C^{u^\alpha}\|_F = \|C^u\|_F$. Note that if C^u is symmetric and non-negative, C^{u^α} still has these properties.

The intuition behind the Hadamard operator is that, while preserving the norm of the original cost matrix, the operator relatively strengthens high-cost elements and weakens low-cost elements of the original cost matrix. It is beneficial to use the operator provided that C^u is roughly accurate in the sense that if $c_{ij}^u > c_{ik}^u$ then examples i and j are more significant nearby examples than those of i and k . Experiments in the next chapter show that C^{u^α} can further improve the quality of C^u constructed by the local scaling method so that the classification performance of a semi-supervised learner is increased.

Any combinations of a label cost matrix C^ℓ of those in previous works such as DNE, MFA and LFDA with an unlabel cost matrix C^u result in new SSL algorithms, and we will call the new algorithms SS-DNE, SS-MFA and SS-LFDA.

5.2.4 Non-Linear Parameterization Using the KPCA Trick

By the linear parameterization described in Section 5.2.2, however, we can only obtain a linear subspace of the original space. As described in Chapter 3, learning a non-linear subspace can be accomplished by first *non-linearly trans-*

forming examples $\{\mathbf{x}_i\}$ into a feature space of $\{\phi(\mathbf{x}_i)\}$ and then learning a linear subspace in the feature space. Nevertheless, a target feature space usually has a high or even an infinite dimensionality so that straightforward learning in the feature space can be intractable. To resolve the computational problem, the kernel trick can be applied. However, applying the kernel trick can be inconvenient since, usually, new mathematical formulas have to be derived and new implementation have to be done separately from the existing linear implementation, described in Section 5.2.2. By the same arguments as in Chapter 3, the KPCA trick can be applied to the SSL framework instead of the kernel trick. We will formally prove this fact in the next chapter.

5.2.4.1 The KPCA trick Algorithm

In this section, the KPCA trick framework is extended to cover learners implemented under our semi-supervised learning framework presented in Section 5.2.2. Let $k(\cdot, \cdot)$ be a PSD kernel function associated with a non-linear function $\phi(\cdot) : \mathbb{R}^{d_0} \mapsto \mathcal{H}$ such that $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ (Schölkopf and Smola, 2001). Denote ϕ_i for $\phi(\mathbf{x}_i)$ for $i = 1, \dots, \ell + u$ and ϕ' for $\phi(\mathbf{x}')$. As in Chapter 3, The central idea of the KPCA trick is to represent each ϕ_i and ϕ' in a new “finite”-dimensional space, with dimensionality bounded by $\ell + u$, without any loss of information. Within the framework, a new coordinate of each example is computed “explicitly”, and each example in the new coordinate is then used as the input of any existing semi-supervised learner without any re-implementations.

As before, for simplicity, we assume that $\{\phi_i\}$ is linearly independent and has its center at the origin, i.e. $\sum_i \phi_i = 0$. Since we have $n = \ell + u$ total examples, the span of $\{\phi_i\}$ has dimensionality⁷ n . Here we claim that each example ϕ_i can be represented as $\varphi_i \in \mathbb{R}^n$ with respect to a new *orthonormal* basis $\{\psi_i\}_{i=1}^n$ such that $\text{span}(\{\psi_i\}_{i=1}^n)$ is the same as $\text{span}(\{\phi_i\}_{i=1}^n)$ without loss of any information. More precisely, we define

$$\varphi_i = \left(\langle \phi_i, \psi_1 \rangle, \dots, \langle \phi_i, \psi_n \rangle \right) = \Psi^T \phi_i. \quad (5.13)$$

where $\Psi = (\psi_1, \dots, \psi_n)$. An inner-product of $\langle \phi_i, \psi_j \rangle$ can be conveniently computed

⁷In cases that $\{\phi_i\}$ is not linearly independent, this dimensionality is less than $\ell + u$.

by KPCA where each ψ_i is a principal component in the feature space. Likewise, a new test point ϕ' can be mapped to $\varphi' = \Psi^T \phi'$.

The KPCA trick algorithm for semi-supervised learning consisting of three simple steps is shown in Figure 5.4. All semi-supervised learners can be kernelized by this simple algorithm. In the algorithm, we denote a semi-supervised learner by ssl which outputs the best linear map A^* (or the best embedded set of points Z^*).

Input: 1. training examples: $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+u}\}$
 2. a new example: \mathbf{x}'
 3. a kernel function: $k(\cdot, \cdot)$
 4. a linear semi-supervised learning algorithm: ssl (see Figure 5.2)

Algorithm:
 (1) Apply $\text{kpca}(k, \{\mathbf{x}_i\}_{i=1}^{\ell+u}, \mathbf{x}')$ such that $\{\mathbf{x}_i\} \mapsto \{\varphi_i\}$ and $\mathbf{x}' \mapsto \varphi'$.
 (2) Apply ssl with new inputs $\{(\varphi_1, y_1), \dots, (\varphi_\ell, y_\ell), \varphi_{\ell+1}, \dots, \varphi_{\ell+u}\}$ to achieve A^* .
 (3) Perform kNN based on the distance $\|A^* \varphi_i - A^* \varphi'\|$.

Figure 5.4: The KPCA trick algorithm for semi-supervised learning.

5.2.4.2 Representer Theorems

In this section, two semi-supervised learning versions of the representer theorem Schölkopf et al. (2001) are stated to validate the KPCA trick algorithm shown in Figure 5.4. They will be proven in the next chapter. The first theorem states that, for a non-regularized learner, there exists A'^* such that $A'^* \varphi_i = A^* \phi_i$ for all i . Therefore, as an alternative to some optimal A^* (which can have huge or infinitely many number of rows), we can obtain an equally optimal A'^* (whose number of rows is bounded by $\ell + u$). In contrast, the second theorem implies a stronger result for a learner which employs a regularizer, e.g. SS-DNE, that *every* optimal A^* attaining the best objective value, its number of rows is bounded by $\ell + u$. Section 5.2.5 illustrates how the three algorithms, SS-MFA, SS-LFDA and SS-DNE, obey the representer theorems.

We write a function f with inputs x_1, \dots, x_n as $f(\{x_i\}_{i=1}^n)$.

Theorem 3. (*Weak Representer Theorem for SSL*) For arbitrary objective function f which depends solely on inner products of linearly transformed examples,

the optimization,

$$\begin{aligned} \min_A f\left(\{\langle A\phi_i, A\phi_j \rangle\}_{i,j=1}^{\ell+u}\right) \\ \text{s.t. } A : \mathcal{H} \rightarrow \mathbb{R}^d \text{ is a bounded linear map,} \end{aligned}$$

has the same optimal value as

$$\min_{A' \in \mathbb{R}^{d \times (\ell+u)}} f(\{\varphi_i^T A'^T A' \varphi_j\}_{i,j=1}^{\ell+u}).$$

In the statement of the next theorem, we use the fact, which will be proven in Lemma 1 of Chapter 7, that $A = \sum_{i=1}^d \langle \cdot, \tau_i \rangle e_i$ for some $\{\tau_i\}_{i=1}^d \subseteq \mathcal{H}$ and orthonormal set $\{e_i\}_{i=1}^d$ of \mathbb{R}^d . Thus, A is representable by $\{\tau_i\}$.

Theorem 4. (*Strong Representer Theorem for SSL*) For arbitrary monotonically increasing functions g and objective functions f which depend solely on inner products between an example ϕ_i and a linear functional τ_j . Let

$$h(\tau_1, \dots, \tau_d, \phi_1, \dots, \phi_{\ell+u}) = f(\langle \tau_1, \phi_1 \rangle, \dots, \langle \tau_i, \phi_j \rangle, \dots, \langle \tau_d, \phi_{\ell+u} \rangle) + g\left(\sum_{i=1}^d \|\tau_i\|\right).$$

Any optimal set of linear functionals

$$\begin{aligned} \arg \min_{\{\tau_i\}} h(\tau_1, \dots, \tau_d, \phi_1, \dots, \phi_{\ell+u}) \\ \text{s.t. } \forall i \tau_i : \mathcal{H} \rightarrow \mathbb{R} \text{ is a bounded linear functional} \end{aligned}$$

must admit the representation of $\tau_i = \sum_{j=1}^n u_{ij} \tilde{\psi}_j$ ($i = 1, \dots, d$).

Although the statement of Theorem 4 is quite different from the statement of Theorem 3, it is the fact that we can achieve the result of Theorem 3 from Theorem 4. Note that from Theorem 4, we can write

$$A^* = (\tau_1, \dots, \tau_d)^T = U\Psi^T,$$

where U is a matrix having elements u_{ij} . Hence,

$$\langle A^* \phi_i, A^* \phi_j \rangle = \langle U\Psi^T \phi_i, U\Psi^T \phi_j \rangle = \langle U\varphi_i, U\varphi_j \rangle,$$

and the result of Theorem 3 is obtained by renaming U to A^* .

The kernel versions of algorithms such as DNE, MFA and KLFDA are called KDNE, KMFA and KLFDA, and their SSL kernel versions are called SS-KDNE, SS-KMFA and SS-KLFDA.

5.2.5 Remarks

1. The main optimization problem shown in Eq.(5.6) can be restated as follows: Fukunaga (1990)

$$\arg \min_{A \in \mathbb{R}^{d \times d_0}} \text{trace} \left((ABA^T)^{-1} AX(D-C)X^T A^T \right).$$

Within this formulation, the corresponding optimal solution is invariant under a non-singular linear transformation; i.e., if A^* is an optimal solution, then TA^* is also an optimal solution for any non-singular $T \in \mathbb{R}^{d \times d}$ (Fukunaga, 1990, pp.447). We note that four choices of T which assign a weight to each new axis are natural: (1) $T = I$, (2) T is a diagonal matrix with $T_{ii} = \frac{1}{\|\mathbf{a}^{(i)}\|}$, i.e. T normalizes each axis to be equally important, (3) T is a diagonal matrix with $T_{ii} = \sqrt{\lambda_i}$ as $\sqrt{\lambda_i}$ determines how well each axis $\mathbf{a}^{(i)}$ fits the objective function $\mathbf{a}^{(i)T} X(D-C)X^T \mathbf{a}^{(i)}$, and (4) T is a diagonal matrix with $T_{ii} = \frac{\sqrt{\lambda_i}}{\|\mathbf{a}^{(i)}\|}$, i.e. a combination of (2) and (3). Note that these four choices of T can also be applied to the solution of Eq.(5.2) (applied to Z^* instead of A^*).

2. The matrix B defined in Subsection 5.2.3 of the two algorithms, SS-MFA and SS-LFDA, is guaranteed to be positive semidefinite (PSD) but may not be positive definite (PD), i.e., B may not be of full rank. In this case, B is singular and we cannot immediately apply Eq.(5.3) and Eq.(5.7) to solve the optimization problems. One common way to solve this difficulty is, instead of B , by using $(B + \epsilon I)$, for some value of $\epsilon > 0$, which is now guaranteed to be of full rank. Since ϵ acts in a role of regularizer, it makes sense to set $\epsilon = \gamma$, the regularization parameter specified in Section 5.2.3. Similar settings of ϵ has also been used by some existing algorithms, e.g. (Friedman, 1989; Sugiyama et al., 2008).

Also, in a small sample size problem where $X(D-C)X^T$ is not full-rank, the obtained matrix A^* (or some columns of A^*) lie in the null space of $X(D-C)X^T$. Although this matrix does optimize our optimization problem, it usually overfits the given data. One possible solution to this problem is to apply PCA to the given data in the first place (Belhumeur et al., 1997) so that the resulted data have dimensionality less than or equal to the rank of $X(D-C)X^T$. Note that in our KPCA trick framework this pre-process is automatically accomplished as KPCA has to be applied to a learner as shown in Figure 3.3.

3. Recall that the purpose of parameterization (using Eq. (5.6) instead of Eq. (5.2)) is to handle unseen data. However, in transductive problems where we already know all examples to be tested $\{\mathbf{x}_i\}_{i=\ell+1}^{\ell+u}$, we can directly use Eq. (5.2). Nevertheless, it turns out that, in algorithms such as SS-LFDA and SS-MFA, the same solution is obtained from both the direct method (Eq.(5.2)) and the kernel method (Figure 5.4) provided that $P = (\varphi_1, \dots, \varphi_n)$ is full-rank. For example, to see this for the case of SS-MFA, we have to solve the following equation:

$$P(D-C)P^T \mathbf{a}^{(j)} = \lambda_j P(D^I - C^I)P^T \mathbf{a}^{(j)}.$$

Since P is invertible by assumption, we have

$$(D-C)P^T \mathbf{a}^{(j)} = \lambda_j (D^I - C^I)P^T \mathbf{a}^{(j)}.$$

Finally, by changing the variable $\mathbf{z}^{(j)} = P^T \mathbf{a}^{(j)}$ (this is valid because P^T is full-rank), we obtain Eq.(5.3).

4. Here, we show that the representer theorems do validate the three algorithms presented in Section 5.2.3. Note that, by the method of lagrange multiplier for equality constraint, under a non-linear transform $\phi(\cdot)$, the optimization problem shown in Eq.(5.6) can be restated as

$$\arg \min_{A \in \mathbb{R}^{d \times d_0}} \text{trace} \left(A \Phi (D-C) \Phi^T A^T \right) + \text{trace} \left(\Lambda (A B A^T - I) \right),$$

where Λ is a diagonal matrix of lagrange multipliers λ_i . Therefore, now the optimization has no constraint. Note that

$$\text{trace}\left(\Lambda(ABA^T - I)\right) = \sum_i \lambda_i \left(\mathbf{a}^{(i)T} B \mathbf{a}^{(i)} - 1\right)$$

With this formulation, it is possible to show that Theorem 3 validates the use of the KPCA trick applying to the SS-MFA and SS-LFDA algorithms. By substituting B and C of the two algorithms, we have that the objective function depends only on the quantities of $\|\mathbf{a}^{(i)T} \phi_i - \mathbf{a}^{(i)T} \phi_j\|^2$ for $i, j = 1, \dots, \ell + u$. Since $\|\mathbf{a}^{(i)T} \phi_i - \mathbf{a}^{(i)T} \phi_j\|^2 = \langle \mathbf{a}^{(i)T} \phi_i - \mathbf{a}^{(i)T} \phi_j, \mathbf{a}^{(i)T} \phi_i - \mathbf{a}^{(i)T} \phi_j \rangle$, the objective function then depends only on the inner-product of transformed examples. Hence, it is showed that the applications of KPCA trick with respect to SS-MFA and SS-LFDA are valid by Theorem 3.

To validate the application of KPCA trick with respect to SS-DNE, it is necessary to apply Theorem 4. Since we can calculate a solution of SS-DNE iteratively, for each $\mathbf{a}^{(i)}$, the SS-DNE optimization can be restated as follows:

$$\begin{aligned} \mathbf{a}^{(i)} &= \arg \min_{\mathbf{a} \in \mathcal{H}^{(i)}} \frac{\mathbf{a}^T \Phi(D - C)\Phi^T \mathbf{a}}{\|\mathbf{a}\|^2} \\ &= \arg \max_{\mathbf{a} \in \mathcal{H}^{(i)}} \frac{-\mathbf{a}^T \Phi(D - C)\Phi^T \mathbf{a}}{\|\mathbf{a}\|^2} \end{aligned}$$

where $\mathcal{H}^{(i)} = \{\phi \in \mathcal{H} | \phi \perp \mathbf{a}^{(j)}, j < i\}$ and $A = (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)})^T$.

Since it is recommended for the original version of DNE to consider only eigenvectors with *negative* eigenvalues of matrices $\Phi(D - C)\Phi^T$ (Zhang et al., 2007b) and discard all other eigenvectors, without loss of generality, we can regard $\Phi(D - C)\Phi^T$ as negative definite and hence $-\Phi(D - C)\Phi^T$ as positive definite. Then, the optimization can be further restated as follows:

$$\mathbf{a}^{(i)} = \arg \min_{\mathbf{a} \in \mathcal{H}^{(i)}} \frac{\|\mathbf{a}\|^2}{-\mathbf{a}^T \Phi(D - C)\Phi^T \mathbf{a}}$$

or equivalently,

$$\mathbf{a}^{(i)} = \arg \min_{\mathbf{a} \in \mathcal{H}^{(i)}, \mathbf{a}^T \Phi(D-C)\Phi^T \mathbf{a} = -1} \|\mathbf{a}\|^2.$$

Finally, we have

$$\mathbf{a}^{(i)} = \arg \min_{\mathbf{a} \in \mathcal{H}^{(i)}} f(\mathbf{a}^T \Phi(D-C)\Phi^T \mathbf{a}) + \|\mathbf{a}\|^2, \quad (5.14)$$

where

$$f(\mathbf{a}^T \Phi(D-C)\Phi^T \mathbf{a}) = \begin{cases} 0, & \text{if } \mathbf{a}^T \Phi(D-C)\Phi^T \mathbf{a} = -1, \\ \infty, & \text{otherwise.} \end{cases}$$

This formulation of SS-DNE obeys the condition of Theorem 4 (with $g(\|\mathbf{a}\|) = \|\mathbf{a}\|^2$), and hence the application of KPCA trick with respect to SS-DNE is valid.

5.3 Connection to Related Work

As we already described in Section 5.2.3, our framework generalizes various existing supervised and unsupervised manifold learners (Sugiyama, 2007; Yan et al., 2007; Zhang et al., 2007b; Cai et al., 2007; Hoi et al., 2006; Chen et al., 2005; Cheng et al., 2004; von Luxburg, 2007; He and Niyogi, 2004; Zelnik-Manor and Perona, 2004). The KPCA trick and the two representer theorems are new in the field of semi-supervised learning.

There are some supervised manifold learners which cannot be represented in our framework (Goldberger et al., 2005; Globerson and Roweis, 2006; Weinberger et al., 2006; Yang et al., 2006a; Torresani and Lee, 2007; Tao et al., 2009) because cost functions of these algorithms are not linear with respect to distances among examples. Extension of these algorithms to handle semi-supervised learning problems is an interesting future work.

Yang et al. (2006b) present another semi-supervised learning framework which solves entirely different problems to problems considered in this paper. They propose to extend unsupervised algorithms such as ISOMAP Tenenbaum et al. (2000) and Laplacian Eigenmap (Chapelle et al., 2006, Chapter 16) to cases

in which information about exact locations of some points is available. Xu et al. (2009) proposed a beautiful framework based on *optimal reverse prediction* which unifies many existing algorithms. Nevertheless, given a new learner, a systematic method to find a corresponding reverse formula is not yet shown. Therefore, practical uses of this framework are now limited.

To the best of our knowledge, there are currently four existing semi-supervised dimensionality reduction frameworks in literatures which have similar goal to ours; all of them are very recently proposed. Here, we subsequently show that these frameworks can be restated as *special cases* of our framework.

5.3.1 Sugiyama et al. (2008)

Sugiyama et al. (2008) extends the LFDA algorithm to handle a semi-supervised learning problem by adding the PCA objective function $f^{PCA}(A)$ (see Chapter 2) into the objective function $f^\ell(A)$ of LFDA described in Section 5.2.3. To describe Sugiyama et al.'s algorithm, namely 'SELF', without loss of generality, we assume that training data are centered at the origin, i.e. $\sum_{i=1}^n \mathbf{x}_i = 0$, and then we can write $f^{PCA}(A) = -\sum_{i=1}^n \|A\mathbf{x}_i\|^2$. Sugiyama et al. propose to solve the following problem:

$$A^* = \arg \min_{ABA^T=I} \left(\sum_{i,j=1}^{\ell} c_{ij}^\ell \|A\mathbf{x}_i - A\mathbf{x}_j\|^2 - \gamma \sum_{i=1}^n \|A\mathbf{x}_i\|^2 \right) \quad (5.15)$$

Interestingly, it can be shown that this formulation can be formulated in our framework with unlabel cost c_{ij}^u being *negative*, and hence our framework subsumes SELF. To see this, let $c_{ij}^u = -1/2n$, for all $i, j = 1, \dots, n$. Then, the objective $f^u(A)$ is equivalent to $f^{PCA}(A)$:

$$\begin{aligned} f^u(A) &= \sum_{i,j=1}^n -\frac{1}{2n} \|A\mathbf{x}_i - A\mathbf{x}_j\|^2 = -\frac{1}{2n} \sum_{i,j=1}^n \langle A\mathbf{x}_i - A\mathbf{x}_j, A\mathbf{x}_i - A\mathbf{x}_j \rangle \\ &= -\frac{1}{2n} \left(2 \sum_{i,j=1}^n \langle A\mathbf{x}_i, A\mathbf{x}_i \rangle - 2 \sum_{i,j=1}^n \langle A\mathbf{x}_i, A\mathbf{x}_j \rangle \right) \\ &= -\frac{1}{2n} \left(2n \sum_{i=1}^n \|A\mathbf{x}_i\|^2 - 2 \langle A \sum_{i=1}^n \mathbf{x}_i, A \sum_{j=1}^n \mathbf{x}_j \rangle \right) \\ &= f^{PCA}(A), \end{aligned}$$

where we use the fact that $\sum_{i=1}^n \mathbf{x}_i = 0$. This proves that SELF is a special case of our framework.

Note that the use of negative unlabel costs $c_{ij}^u = -1/2n$ results in an algorithm which attempts to preserve a *global* structure of the input data and does not convey the manifold assumption where only a *local* structure should be preserved. Therefore, when the input unlabeled data lie in a complicated manifold, it is not appropriate to apply $f^u(A) = f^{PCA}(A)$.

5.3.2 Song et al. (2008)

Song et al. propose to extend FDA and another algorithm named *maximum margin criterion* (MMC) (Li et al., 2006) to handle a semi-supervised learning problem. Their idea of semi-supervised learning extension is similar to ours as they add the term $f^u(\cdot)$ into the objective of FDA and MMC (hence, we call them, SS-FDA and SS-MMC, respectively). However, SS-FDA and SS-MMC cannot handle problems where data of each class form a manifold or several clusters as shown in Figure 5.3 because SS-FDA and SS-MMC satisfy the condition (1) but not (1*). In fact, SS-FDA and SS-MMC can both be restated as instances of our framework. To see this, we note that the optimization problem of SS-MMC can be stated as

$$A^* = \arg \min_{AA^T=I} \gamma' \text{trace}(AS_w A^T) - \text{trace}(AS_b A^T) + \gamma f^u(A), \quad (5.16)$$

where S_b and S_w are the standard between-class and within-class scatter matrices, respectively (Fukunaga, 1990):

$$S_w = \sum_{i=1}^c \sum_{j|y_j=i} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \quad \text{and} \quad S_b = \sum_{i=1}^c (\boldsymbol{\mu} - \boldsymbol{\mu}_i)(\boldsymbol{\mu} - \boldsymbol{\mu}_i)^T,$$

where $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$, $\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{i=1}^{n_i} \mathbf{x}_i$ and n_i is the number of examples in the i^{th} class. It can be verified that $\text{trace}(AS_w A^T) = \sum_{i,j=1}^{\ell} c_{ij}^w \|A\mathbf{x}_i - A\mathbf{x}_j\|^2$ and $\text{trace}(AS_b A^T) = \sum_{i,j=1}^{\ell} c_{ij}^b \|A\mathbf{x}_i - A\mathbf{x}_j\|^2$ where

$$c_{ij}^b = \begin{cases} (\frac{1}{n_k} - \frac{1}{n}), & \text{if } y_i = y_j = k, \\ -\frac{1}{n}, & \text{otherwise,} \end{cases} \quad \text{and} \quad c_{ij}^w = \begin{cases} \frac{1}{n_k}, & \text{if } y_i = y_j = k, \\ 0, & \text{otherwise.} \end{cases}$$

Hence, by setting $c_{ij}^\ell = \gamma' c_{ij}^w - c_{ij}^b$ we finish our proof that SS-MMC is a special case of our framework. The proof that SS-FDA is in our framework is similar to that of SS-MMC.

5.3.3 Zhang et al. (2007a)

Zhang et al. (2007a) also recently proposed a learner called *Semi Supervised Dimensionality Reduction* which is also a special case of ours. A proof showing that it is a special case of our framework is similar to those of SELF and SS-MMC, and therefore we omit the details.

5.3.4 Li et al. (2007)

We found that the transductive framework proposed by Li et al. (2007) is similar to ours even though their framework has totally different representations compared to our framework. Similar to ours, Li et al.'s framework also has the main goal to solve an optimization of the form of Eq. (5.1). However, our framework is more general than theirs in three important aspects.

First, Li et al.'s framework does not generalize some existing manifold learning algorithms such as LFDA and MFA since, in contrast to our framework, their framework does not allow a constraint of the form $ZBZ^T = I$ (see Eq. (5.2)). Therefore, the relations among existing algorithms which are characterized by the conditions (1), (2), (1*) and (2*) presented in Section 5.2.3 cannot be established under their framework.

Second, since Li et al.'s framework has been focused only on transductive problems, it is not clear how their framework can be used in a general semi-supervised learning problem where a new example is given. In contrast, we dedicated Section 5.2.2 to explain about linear parameterization, and in Section 5.2.4 we developed the novel KPCA trick framework for non-linear parameterization which can be used for handling a general semi-supervised learning problem.

Third, the representer theorem proven in Li et al.'s work (Li et al., 2007, Section 2.3) is more restrictive than ours, i.e. in order to use their framework, C^ℓ and C^u must satisfy some strict conditions as stated in Theorem 4 of their paper,

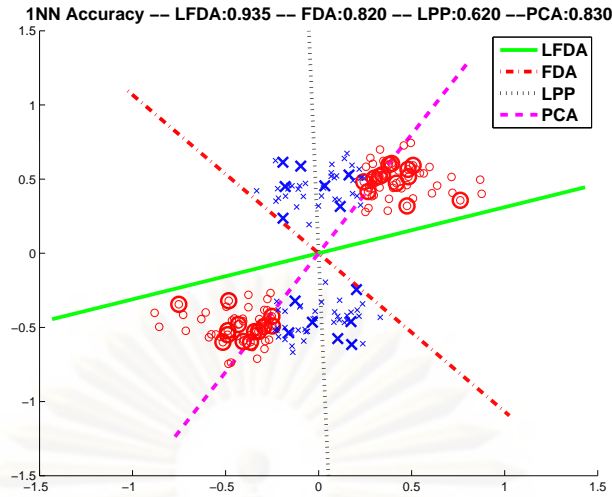


Figure 5.5: The first toy example. The projection axes of three algorithms, namely FDA, LFDA, LPP and PCA, are presented. Big circles and big crosses denote labeled examples while small circles and small crosses denote unlabeled examples. Their percentage accuracy over the unlabeled examples are shown on the top.

and this prohibits a general use of their framework. Moreover, they prove their representer theorem only in cases that the target dimensionality d is 1. On the other hand, Theorem 3 and Theorem 4 proven in this paper allow d to be any value which is not more than the input dimensionality d_0 .

Since any cost matrices which can be used in Li et al.'s framework can also be used in our framework, it can be viewed that our framework generalizes their framework. Note that as Li et al.'s framework generalizes unsupervised manifold learning framework such as *Isometric Mapping* (Tenenbaum et al., 2000), *Locally Linear Embedding* (Roweis and Saul, 2000) and *Laplacian Eigenmap* (Belkin and Niyogi, 2003), our framework generalizes these unsupervised frameworks as well, i.e. there exists cost matrices C^u which make a learner of our framework behaves exactly like these unsupervised learners.

5.3.5 Improvement over Previous Frameworks

In this section, we explain why SELF and SS-FDA proposed by Sugiyama et al. (2008) and Song et al. (2008) described above are *not* enough to solve some semi-supervised learning problems, even simple ones shown in Figure 5.5 and Figure 5.6.

In Figure 5.5, four dimensionality reduction algorithms, FDA, LFDA, LPP

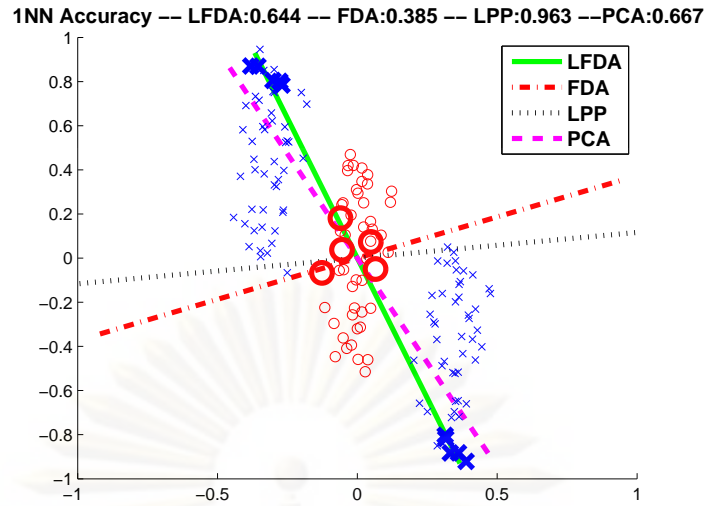


Figure 5.6: The second toy example consisting of three clusters of two classes.

and PCA are performed on this dataset. Because of multi-modality, FDA cannot find an appropriate projection. Since the two clusters do not contain data of the same class, LPP which attempts to preserve the structure of the two clusters also fails. Likewise, PCA fails because it does not take labeled data into account. In this case, only LFDA can find a proper projection since it can cope with multi-modality and can take into account the labeled examples. Note that since SS-FDA is a linearly combined algorithm of FDA and LPP, it can only find a projection lying in between the projections discovered by FDA and LPP, and in this case SS-FDA cannot find an efficient projection, unlike LFDA and, of course, SS-LFDA derived from our framework.

A similar argument can be given to warn an uncaredful use of SELF in some situations. In Figure 5.6, four dimensionality reduction algorithms, FDA, PCA, LFDA and LPP are performed on this dataset. Because of multi-modality, FDA and PCA cannot find an appropriate projection. Also, since there are only a few labeled examples, LFDA fails to find a good projection as well. In this case, only LPP can find a proper projection since it can cope with multi-modality and can take the unlabeled examples into account. Note that since SELF is a linearly combined algorithm of LFDA and PCA, it can only find a projection lying in between the projections discovered by LFDA and PCA, and in this case SELF cannot find a correct projection, unlike a semi-supervised learner like SS-LFDA derived from our framework which, as explained in Section 5.2.3, employs the LPP cost function as its C^u .

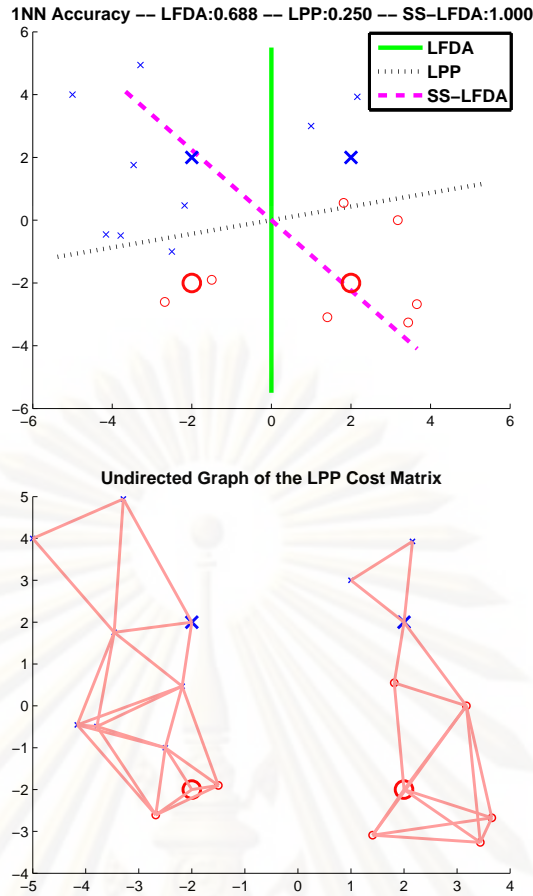


Figure 5.7: (Top) The third toy example where only a semi-supervised learner is able to find a good projection. (Bottom) An undirected graph corresponding to the values of C^u used by LPP and SS-LFDA. In this figure, a pair of examples i and j has a link if and only if $c_{ij}^u > 0.1$. This graph explains why LPP projects the data in the axis shown in the top figure; LPP, which does not apply the label information, tries to choose a projection axis which squeezes the two clusters as much as possible. Note that we apply a local-scaling method, Eq.(5.10), to specify C^u .

Since a semi-supervised manifold learner derived from our framework can be intuitively thought of as a combination of a supervised learner and an unsupervised learner. One may misunderstand that a semi-supervised learner cannot discover a good subspace if neither is a supervised learner nor an unsupervised learner able to discover a good subspace. The above two toy examples may also mislead the readers to think in that way. In fact, that intuition is incorrect. Here, we give another toy example shown in Figure 5.7 where only a semi-supervised learner is able to discover a good subspace but neither is its supervised and unsupervised counterparts. Intuitively, a semi-supervised learner is able to exploit useful information from both labeled and unlabeled examples.

CHAPTER VI

SPECTRAL SEMI-SUPERVISED LEARNING FRAMEWORK: PRACTICE

In this chapter, classification performance of each algorithm derived from our framework is demonstrated. A similar experimental setting as those in previous works (Sugiyama et al., 2008; Chapelle et al., 2006, Chapter 21) is employed so that our results can be compared to them.

6.1 Experimental Setting

In all experiments, two semi-supervised learners, SS-LFDA and SS-DNE, derived from our framework are compared to relevant existing algorithms, PCA, LPP*, LFDA, DNE and SELF (Sugiyama et al., 2008). In contrast to the standard LPP which does not apply the Hadamard power operator explained in Section 5.2.3, we denote LPP* as a variant of LPP applying the Hadamard power operator.

Non-linear semi-supervised manifold learning is also experimented by applying the KPCA trick algorithm illustrated in Figure 5.4. Since it is not our intention to apply the “best” kernel *but* to compare efficiency between a “semi-supervised” kernel learner and its base “supervised” (and “unsupervised”) kernel learners, we simply apply the 2^{nd} -degree polynomial kernel $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^2$ to the kernel algorithms in all experiments.

By using the nearest neighbor algorithm on their discovered subspaces, classification accuracies of the experimented learners are measured on five standard datasets shown on Table 6.1, the first two datasets are obtained from the UCI repository (Asuncion and Newman, 2007), the next two datasets mainly designed for testing a semi-supervised learner (Chapelle et al., 2006, Chapter 21). The final dataset, *extended Yale B* (Georghiades et al., 2001), is a standard dataset of a face recognition task. The classification performance of each algorithm on each dataset is measured by the average test accuracy over 25 realizations of randomly

Table 6.1: Details of each dataset: d_0, c, ℓ, u and t denote the numbers of input features, classes, a number of labeled examples, a number of unlabeled examples and a number of testing examples, respectively. ‘*’ denotes the transductive setting used in small datasets, where all examples which are not labeled are given as unlabeled examples and used as testing examples as well. d , determined by using prior knowledge, denotes the target dimensionality for each dataset. ‘GOOD NEIGHBORS’ denotes a quantity which measures a goodness of unlabeled data for each dataset.

NAME	d_0	c	$\ell + u + t$	ℓ	u	d	GOOD NEIGHBORS	
							LINEAR	KERNEL
IONOSPHERE	34	2	351	10/100	*	2	0.866	0.843
BALANCE	4	3	625	10/100	300	1	0.780	0.760
BCI	117	2	400	10/100	*	2	0.575	0.593
USPS	241	2	1500	10/100	300	10	0.969	0.971
M-EYALE	504	5	320	20/100	*	10	0.878	0.850

splitting each dataset into training and testing subsets.

Three parameters are needed to be tuned in order to apply a semi-supervised learner derived from our framework (see Section 5.2.3): γ , the regularizer, α , the degree of the Hadamard power operator and k , the k^{th} -nearest neighbor parameter needed to construct the cost matrices. To make our learners satisfy the condition (1*) described in Section 5.2.3, it is clear that k should be small compared to n_c , the number of training examples of class c . From experience, we found that semi-supervised learners are quite insensitive to various small values of k . Therefore, in all our experiments, we simply set $k = \min(3, n_c)$ so that only two parameters, γ and α , are needed to be tuned. We tune these two parameters via cross validation. Note that only α is needed to be tuned for LPP* and only γ is needed to be tuned for SELF.

The ‘GOOD NEIGHBORS’ score shown in Table 6.1 is due to Sugiyama et al. (2008). The score is simply defined as a training accuracy of the nearest neighbor algorithm when *all available data are labeled and are given to the algorithm*. Intuitively, if a dataset gets a high score, unlabeled examples should be useful since it indicates that each pair of examples having a high penalty cost c_{ij}^u should belong to the same class. Note that on Table 6.1 there are two scores for each dataset: LINEAR is a score on a given input space while KERNEL measures a score on a

feature space corresponding to the 2^{nd} -degree polynomial kernel.

6.2 Numerical Results

Numerical results are shown in Table 6.2 for the case of $\ell = 10$ (except M-EYALE where $\ell = 20$) and Table 6.3 for the case of $\ell = 100$. In experiments, SS-DNE and SS-LFDA are compared their classification accuracies to their unsupervised and supervised counterparts: LPP* and DNE for SS-DNE, and LPP* and LFDA for SS-LFDA. SELF is also compared to SS-LFDA as they are related semi-supervised learners originated from LFDA. Our two algorithms will be highlighted if they are superior to their counterpart opponents. For comparison, accuracies of a learner from a popular framework of Sindhwani et al. (Chapelle et al., 2006, Chapter 12), namely *Laplacian least-square* (LapLS), is also demonstrated on binary-class datasets¹.

From the results, our two algorithms, SS-LFDA and SS-DNE, outperform all their opponents in 32 out of 40 comparisons: in the first setting of small ℓ (Table 6.2), our algorithms outperform the opponents in 18 out of 20 comparisons while in the second setting of large ℓ (Table 6.3), our algorithms outperform the opponents in 14 out of 20 comparisons. Consequently, our framework offers a semi-supervised learner which consistently improves its base supervised and unsupervised learners.

Note that as the number of labeled examples increases, usefulness of unlabeled examples decreases. We will subsequently discuss and analyze the results of each dataset in details in the next subsections.

6.2.1 Ionosphere

IONOSPHERE is a real-world dataset of radar pulses passing through the ionosphere which were collected by a system in Goose Bay, Labrador. The targets were free electrons in the ionosphere. “Good” radar returns are those showing evidence of some type of structure in the ionosphere. “Bad” returns are those

¹LapLS does not naturally support multi-class problems. Although there are a number of methods to wrap a binary classifier for multi-class problems, the results will be biased due to the choice of a wrapping method. Selection of the best multi-class warping method is beyond the scope of this dissertation.

Table 6.2: Percentage accuracies of SS-DNE and SS-LFDA derived from our framework compared to existing algorithms ($\ell = 10$, except M-EYALE where $\ell = 20$). SS-LFDA and SS-DNE are highlighted when they outperform their opponents (LPP* and DNE for SS-DNE, and LPP*, LFDA and SELF for SS-LFDA). Superscripts indicate %-confidence levels of the one-tailed paired t-test for differences in accuracies between our algorithms and their best opponents. No superscripts denote confidence levels which below 80%. The accuracy of LapLS is also shown for comparison.

LINEAR	PCA	LPP*	DNE	LFDA	SELF	SS-DNE	SS-LFDA	LAPLS
IONOSPHERE	71±1.2	82±1.3	70±1.2	71±1.1	70±1.5	75±1.0	78.1±.9	68±1.9
BALANCE	49±1.9	61±1.9	63±2.2	70±2.2	69±2.3	71±1.8⁹⁹	73±2.3⁸⁰	-
BCI	49.8±.6	53.4±.3	51.3±.6	52.6±.5	52.1±.5	57.1±.6⁹⁹	55.2±.3⁹⁹	53.1±.6
USPS	79±1.2	74±1.0	79.6±.6	80.6±.9	81.7±.8	81.8±.5⁹⁹	83.0±.5⁹⁰	60±1.1
M-EYALE	44.6±.7	67±1.1	66±1.2	71.6±1.0	67.2±.8	76.9±.8⁹⁹	75.7±.9⁹⁹	-
KERNEL	PCA	LPP*	DNE	LFDA	SELF	SS-DNE	SS-LFDA	LAPLS
IONOSPHERE	70±1.8	83.2±.9	70±1.6	71±1.3	74±1.5	87.2±.9⁹⁹	88±1.0⁹⁹	71.4±.7
BALANCE	41.7±.8	47.9±.9	62±2.5	66±2.0	60±2.8	66±1.8⁸⁰	69±1.9⁸⁰	-
BCI	49.7±.3	53.7±.3	50.1±.4	50.3±.6	50.5±.4	53.8±.3	54.1±.3⁸⁰	52.1±.4
USPS	77±1.1	76±1.1	79.9±.5	80.3±.8	80.9±.8	82.0±.4⁹⁹	83.7±.6⁹⁹	61±1.4
M-EYALE	42.1±.9	63.2±.7	58.0±.9	60.3±.8	58.8±.7	69.9±.7⁹⁹	73.2±.8⁹⁹	-

Table 6.3: Percentage accuracies of SS-DNE and SS-LFDA compared to existing algorithms ($\ell = 100$).

LINEAR	PCA	LPP*	DNE	LFDA	SELF	SS-DNE	SS-LFDA	LAPLS
IONOSPHERE	72.8±.6	83.7±.6	77.9±.7	74±1.0	77.8±.5	84.5±.6⁸⁰	84.9±.4⁹⁵	82.3±.7
BALANCE	57±2.2	80±1.3	86.4±.5	87.9±.3	87.2±.4	88.2±.5⁹⁹	86.3±.6	-
BCI	49.5±.5	54.9±.5	53.1±.7	67.9±.5	67.6±.6	63.1±.5⁹⁹	67.5±.6	61.7±.8
USPS	91.4±.3	75.7±.3	91.1±.3	89.3±.4	92.2±.3	92.2±.4⁹⁵	91.6±.3	71.2±.6
M-EYALE	69.4±.4	84.1±.4	92.3±.4	95.4±.3	94.3±.2	93.5±.4⁹⁵	95.7±.2	-
KERNEL	PCA	LPP*	DNE	LFDA	SELF	SS-DNE	SS-LFDA	LAPLS
IONOSPHERE	79.8±.4	89.7±.5	78.7±.9	81.3±.7	81.1±.5	93.6±.2⁹⁹	93.7±.3⁹⁹	76.2±.6
BALANCE	42.5±.3	46.9±.5	84.0±.7	87.8±.7	79±1.6	86.5±.7⁹⁹	87.7±.9	-
BCI	49.7±.5	54.5±.4	51.6±.6	51.0±.8	52.4±.6	57.6±.2⁹⁹	57.0±.4⁹⁹	53.8±.5
USPS	91.1±.3	81.5±.6	91.4±.4	91.2±.4	92.7±.3	92.3±.3⁹⁵	91.9±.3	76±1.1
M-EYALE	66.3±.3	81.9±.5	91.2±.3	89.1±.5	85.8±.6	91.2±.3	94.3±.3⁹⁹	-

that do not. Since we do not know the true decision boundary of IONOSPHERE, we simply set the target dimensionality $d = c = 2$. It can be observed that non-linearization does improve the classification performance of all algorithms.

It can be observed that LPP* is much better than PCA on this dataset, and therefore, unlike SELF, SS-LFDA much improves LFDA. In fact, the main reason that SS-LFDA, SS-DNE and LPP* have good classification performances are because of the Hadamard power operator. This is explained in Figures 6.1, 6.2 and 6.3. From Figures 6.1 and 6.2, defining “nearby examples” be a pair of examples with a link (having $c_{ij}^u \geq 0.36$), we see that *almost every link connects nearby examples of the same class* (i.e. connects good nearby examples). This indicates that our unlabel cost matrix C^u is quite accurate as bad nearby examples rarely

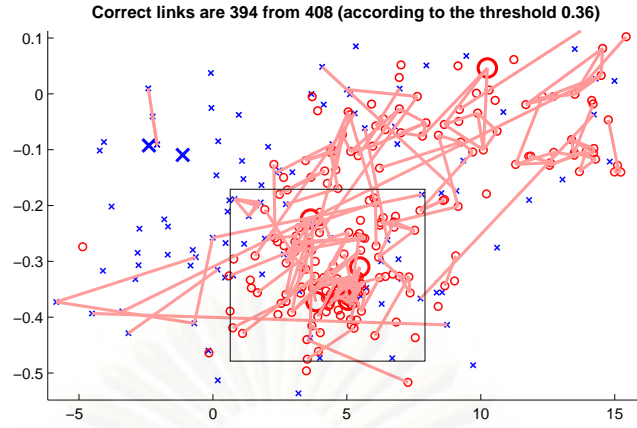


Figure 6.1: The undirected graph corresponding to C^u constructed on IONOSPHERE. Each link corresponds to a pair of nearby examples having $c_{ij}^u \geq 0.36$. The number ‘0.36’ is just chosen for visualizability.

have links. In fact, the *ratio of good nearby examples per total nearby examples* (shortly, the good-nearby-examples ratio) is $394/408 \approx 0.966$. Nevertheless, if we re-define “nearby examples” be a pairs of examples having, e.g., $c_{ij}^u \geq 0.01$, the same ratio then reduces to 0.75 as shown in Figure 6.3 (Left). This indicates that many pairs of examples having small values of c_{ij}^u are of different classes (i.e. bad nearby examples).

Since an algorithm derived from our framework minimizes the *cost-weighted average* distances of every pair of examples (see Eq. (5.6) and its derivation), it is beneficial to further increases the cost of a pair having large c_{ij}^u (since it usually corresponds to a pair of the same class) and decreases the cost of of a pair having small c_{ij}^u . From Eq. (5.12), it can be easily seen that the effect of the Hadamard power operator is exactly what we need. The good-nearby-examples ratios after applying the Hadamard power operator with $\alpha = 8$ are illustrated in Figure 6.3 (Bottom). Notice that, after applying the operator, even pairs with small values of c_{ij}^u are usually of the same class.

6.2.2 Balance

BALANCE is an artificial dataset which was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The 4 attributes containing integer value from 1 to 5 are LEFT_WEIGHT, LEFT_DISTANCE, RIGHT_WEIGHT, and

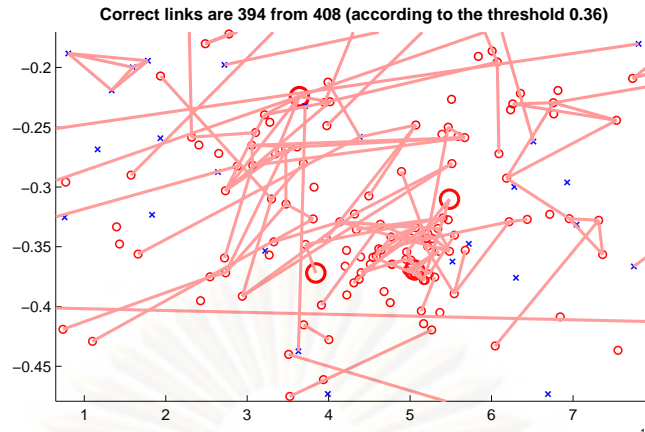


Figure 6.2: Zoom-in on the square area of Figure 6.2.

RIGHT_DISTANCE. The correct way to find the class is the greater of $(\text{LEFT_DISTANCE} \times \text{LEFT_WEIGHT})$ and $(\text{RIGHT_DISTANCE} \times \text{RIGHT_WEIGHT})$. If they are equal, it is balanced. Therefore, there are $5^4 = 625$ total examples and 3 classes in this dataset. Moreover, the correct decision surface is 1-dimensional manifold lying in the feature space corresponding to the $\langle \cdot, \cdot \rangle^2$ kernel so that we set the target dimensionality $d = 1$.

This dataset illustrates another flaw of using PCA in a classification task. After centering, the covariance matrix of the 625 examples is just a multiple of I , the identity matrix. Therefore, any direction is a principal component with largest variance, and PCA is just return a random direction! Hence, we cannot expect much about the classification performance of PCA in this dataset. Thus, PCA cannot help SELF improves much the performance on LFDA, and sometimes SELF degrades the performance of LFDA due to overfitting. In contrast, SS-LFDA often improves the performance of LFDA. Also, SS-DNE is able to improve the classification performance of DNE and LPP* in all settings.

6.2.3 BCI

This dataset originates from the development of a Brain-Computer Interface where a single person performed 400 trials in each of which he imagined movements with either the left hand (the 1st class) or the right hand (the 2nd class). In each trial, electroencephalography (EEG) was recorded from 39 electrodes. An autoregressive model of order 3 was fitted to each of the resulting 39 time series.

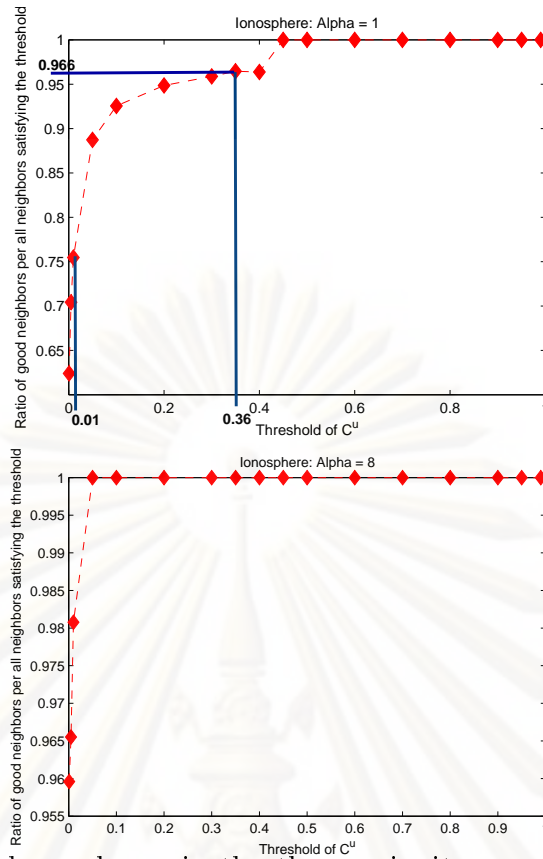


Figure 6.3: For each number x in the the x-axis, its corresponding value on the y-axis is the *ratio between the number of good nearby examples (having $c_{ij}^u > x$ and belonging to the same class) and the number of nearby examples (having $c_{ij}^u > x$)*. The ratios with respect to C^{u^α} are demonstrated where (Top) $\alpha = 1$ (the standard LPP), and where (Bottom) $\alpha = 8$ (LPP*).

The trial was represented by the total of $117 = 39 \times 3$ fitted parameters. The target dimensionality is set to the number of classes, $d = c = 2$. Similar to the previous datasets, SS-LFDA and SS-DNE are usually able to outperform their opponents. Again, PCA is not appropriate for this real-world dataset, and hence SELF is inferior to SS-LFDA.

6.2.4 USPS

This benchmark is derived from the famous USPS dataset of handwritten digit recognition. For each digit, 150 images are randomly drawn. The digits '2' and '5' are assigned to the first class, and all others form the second class. To prevent a use of a domain knowledge, each example is rescaled, noise added, dimension masked and pixel shuffled (Chapelle et al., 2006, Chapter 21). Although there are only 2 classes in this dataset, the original data presumably form 10



Figure 6.4: Extended Yale B Face dataset. 21 examples images from various illumination conditions.

clusters, one for each digit. Therefore, the target dimension d is set to 10.

Often, SS-LFDA and SS-DNE outperform their opponents. Nevertheless, note that SS-LFDA and SS-DNE do not improve much on LFDA and DNE when $\ell = 100$ because 100 labeled examples are quite enough to discriminating the data and therefore unlabeled examples offer relatively small information to semi-supervised learners.

6.2.5 M-Eyale

This face recognition dataset is derived from *extended Yale B* Georghiades et al. (2001), see Figure 6.4. There are 28 human subjects under 9 poses and 64 illumination conditions. In our M-EYALE (Modified Extended Yale B), we randomly chose ten subjects, 32 images per each subject, from the original dataset and down-sampling each example to be of size 21×24 pixels.

M-EYALE consists of 5 classes where each class consists of images of two randomly-chosen subjects. Hence, there should be two separated clusters for each class, and we should be able to see the advantage of algorithms employing the conditions (1*) and (2*) explained in Section 5.2.3. In this dataset, the number of labeled examples of each class is fixed to $\frac{\ell}{c}$ so that examples of all classes are observed. Since this dataset should consist of ten clusters, the target dimensionality is set to $d = 10$.

It is clear that LPP* performs much better than PCA in this dataset. Recall that PCA captures maximum-variance directions; nevertheless, in this face recognition task, maximum-variance directions are not discriminant directions but directions of lighting and posing (Belhumeur et al., 1997). Therefore, PCA captures

totally wrong directions, and hence PCA degrades the performance of SELF from LFDA. In contrast, LPP* much better captures local structures in the dataset and discover much better subspaces. Thus, by cooperating LPP* with LFDA and DNE, SS-LFDA and SS-DNE are able to obtain very good classification accuracies.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

CHAPTER VII

REPRESENTER THEOREM

In this chapter, three versions of the representer theorems will be proven to validate all kernelized learners presented in this dissertation. This chapter is technical. Readers can skip this chapter without sacrificing the main ideas of the dissertation.

7.1 Introduction

A *representer theorem*, along with Mercer theorem, is a key ingredient for validating the *kernel trick* widely used in pattern recognition and machine learning (Schölkopf and Smola, 2001). However, classical representer theorems which cover only an algorithm learning a finite-dimensional linear map cannot be applied to a general *metric learning* algorithm discovering a *countably-infinite* dimensional linear map. In this chapter, we generalize the representer theorems given by Kimeldorf and Wahba (1971) and Schölkopf et al. (2001) to cover the cases of general metric learning.

7.2 Representer Theorem for Metric Learning

Let $\{\mathbf{x}_i\}_{i=1}^n$ denote n examples in an input space¹, $\mathbf{x}_i \in \mathbb{R}^D$. Given a positive semidefinite (PSD) kernel function $k(\cdot, \cdot)$, Mercer theorem implies that there exists a mapping $\phi(\cdot) : \mathbf{x}_i \in \mathbb{R}^D \mapsto \phi(\mathbf{x}_i) \in \mathcal{H}$ where \mathcal{H} is a separable Hilbert space such that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. In this chapter, we simply denote $\phi(\mathbf{x}_i)$ as ϕ_i for brevity.

In machine learning literatures (Chen et al., 2005; Goldberger et al., 2005; Globerson and Roweis, 2006; Weinberger et al., 2006; Yang et al., 2006a; Sugiyama, 2006; Yan et al., 2007; Zhang et al., 2007b; Torresani and Lee, 2007; Li et al., 2007), the task of metric learning is referred to as the task of learning a symmetric PSD matrix M where the metric induced by M in the feature space is $\|\phi_i - \phi_j\|_M = \sqrt{(\phi_i - \phi_j)^T M (\phi_i - \phi_j)}$. Since M can be decomposed to $A^T A$, we also

¹In the context of semi-supervised learning, $n = \ell + u$ as defined in Chapter 5

have $\|\phi_i - \phi_j\|_M = \|A\phi_i - A\phi_j\| = \sqrt{(A\phi_i - A\phi_j)^T(A\phi_i - A\phi_j)}$ where $\|\cdot\|$ denotes the standard (Euclidean) norm of a separable Hilbert space, and we can consider learning a bounded linear map A instead of M . Note that, according to Mercer theorem, in general A maps from a countably-infinite dimensional input space to a countably-infinite dimensional output space; however, the classical representer theorem (Kimeldorf and Wahba, 1971; Schölkopf et al., 2001) cannot be applied to this case of countably-infinite dimensional output. The essence of representer theorem proved in this chapter is to show that the best linear map A can be represented in terms of $\{\phi_i\}_{i=1}^n$ so that computational shortcuts of the kernel trick are possible for metric learning. To be precise, we will show that $M = A^T A = \Phi G \Phi^T$ where $\Phi = (\phi_1, \dots, \phi_n)$ and $G \in \mathbb{S}_+^n$ where \mathbb{S}_+^n denotes a space of $n \times n$ PSD matrices; a practical learning process can now take place by obtaining the best finite-dimensional matrix G instead of the best metric M or the best linear map A .

The following lemma is useful for proving the theorem.

Lemma 1. *Let \mathcal{X}, \mathcal{Y} be two Hilbert spaces and \mathcal{Y} is separable, i.e. \mathcal{Y} has a countable orthonormal basis $\{e_i\}_{i \in \mathbb{N}}$. Any bounded linear map $A : \mathcal{X} \rightarrow \mathcal{Y}$ can be uniquely decomposed as $\sum_{i=1}^{\infty} \langle \cdot, \tau_i \rangle_{\mathcal{X}} e_i$ for some $\{\tau_i\}_{i \in \mathbb{N}} \subseteq \mathcal{X}$.*

Proof. As A is bounded, the linear functional $\phi \mapsto \langle A\phi, e_i \rangle_{\mathcal{Y}}$ is bounded for every i since, by Cauchy-Schwarz inequality, $|\langle A\phi, e_i \rangle_{\mathcal{Y}}| \leq \|A\phi\| \|e_i\| \leq \|A\| \|\phi\|$. By Riesz representation theorem, the map $\langle A\cdot, e_i \rangle_{\mathcal{Y}}$ can be written as $\langle \cdot, \tau_i \rangle_{\mathcal{X}}$ for a unique $\tau_i \in \mathcal{X}$. Since $\{e_i\}_{i \in \mathbb{N}}$ is an orthonormal basis of \mathcal{Y} , for every $\phi \in \mathcal{X}$, $A\phi = \sum_{i=1}^{\infty} \langle A\phi, e_i \rangle_{\mathcal{Y}} e_i = \sum_{i=1}^{\infty} \langle \phi, \tau_i \rangle_{\mathcal{X}} e_i$. \square

For convenience, in our proof below we assume that $\{\phi_i\}_{i=1}^n$ is linearly independent². Our main theorem can be stated as follows³.

Theorem 5. *(Representer Theorem for Full-Rank Mahalanobis Distance Learners)*

²With more cumbersome notations, the proof can be straightforwardly extend to handle the cases where $\{\phi_i\}_{i=1}^n$ is not linearly independent.

³The theorem is more general than what we have discussed; see remark below.

Let $\{\tilde{\psi}_i\}_{i=1}^n$ be a set of points in a feature space \mathcal{X} such that $\text{span}(\{\tilde{\psi}_i\}_{i=1}^n) = \text{span}(\{\phi_i\}_{i=1}^n)$, and \mathcal{X} and \mathcal{Y} be separable Hilbert spaces. For an objective function f depending only on $\{\langle A\phi_i, A\phi_j \rangle\}$, the optimization

$$\begin{aligned} \min_A : & f(\langle A\phi_1, A\phi_1 \rangle, \dots, \langle A\phi_i, A\phi_j \rangle, \dots, \langle A\phi_n, A\phi_n \rangle) \\ \text{s.t. } & A : \mathcal{X} \rightarrow \mathcal{Y} \text{ is a bounded linear map,} \end{aligned}$$

has the same optimal value as,

$$\min_{G \in \mathbb{S}_+^n} f(\tilde{\varphi}_1^T G \tilde{\varphi}_1, \dots, \tilde{\varphi}_i^T G \tilde{\varphi}_j, \dots, \tilde{\varphi}_n^T G \tilde{\varphi}_n),$$

where $\tilde{\varphi}_i = (\langle \phi_i, \tilde{\psi}_1 \rangle, \dots, \langle \phi_i, \tilde{\psi}_n \rangle)^T \in \mathbb{R}^n$.

Proof. To avoid complicated notations, we omit subscripts such as \mathcal{X}, \mathcal{Y} of inner products. The proof will consist of two steps. In the first step, we will prove the theorem by assuming that $\{\tilde{\psi}_i\}_{i=1}^n$ is an orthonormal set. In the second step, we prove the theorem in general cases where $\{\tilde{\psi}_i\}_{i=1}^n$ is not necessarily orthonormal. The proof of the first step requires an application of Fubini theorem (Lewkeeratiyutkul, 2006).

Step 1. Assume that $\{\tilde{\psi}_i\}_{i=1}^n$ is an orthonormal set. Let $\{e_i\}_{i=1}^\infty$ be an orthonormal basis of \mathcal{Y} . For any $\phi' \in \mathcal{X}$, we have, by Lemma 1, $A\phi' = \sum_{k=1}^\infty \langle \phi', \tau_k \rangle e_k$. Hence, for each bounded linear map $A : \mathcal{X} \rightarrow \mathcal{Y}$, and $\phi, \phi' \in \text{span}(\{\tilde{\psi}_i\}_{i=1}^n)$, we have $\langle A\phi, A\phi' \rangle = \sum_{k=1}^\infty \langle \phi, \tau_k \rangle \langle \phi', \tau_k \rangle$.

Note that Each τ_k can be decomposed as $\tau'_k + \tau_k^\perp$ such that τ'_k lies in $\text{span}(\{\tilde{\psi}_i\}_{i=1}^n)$ and τ_k^\perp is orthogonal to the span. These facts make $\langle \phi', \tau_k \rangle = \langle \phi', \tau'_k \rangle$ for every k . Moreover, $\tau'_k = \sum_{j=1}^n u_{kj} \tilde{\psi}_j$, for some $\{u_{k1}, \dots, u_{kn}\} \subset \mathbb{R}^n$.

จุฬาลงกรณ์มหาวิทยาลัย

Hence, we have

$$\begin{aligned}
\langle A\phi, A\phi' \rangle &= \sum_{k=1}^{\infty} \langle \phi, \tau_k \rangle \langle \phi', \tau_k \rangle = \sum_{k=1}^{\infty} \langle \phi, \tau'_k \rangle \langle \phi', \tau'_k \rangle \\
&= \sum_{k=1}^{\infty} \langle \phi, \sum_{i=1}^n u_{ki} \tilde{\psi}_i \rangle \langle \phi', \sum_{i=1}^n u_{ki} \tilde{\psi}_i \rangle \\
&= \sum_{k=1}^{\infty} \sum_{i,j=1}^n u_{ki} u_{kj} \langle \phi, \tilde{\psi}_i \rangle \langle \phi', \tilde{\psi}_j \rangle \\
\text{(Fubini theorem: explained below)} &= \sum_{i,j=1}^n \left(\sum_{k=1}^{\infty} u_{ki} u_{kj} \right) \langle \phi, \tilde{\psi}_i \rangle \langle \phi', \tilde{\psi}_j \rangle \\
&= \sum_{i,j=1}^n G_{ij} \langle \phi, \tilde{\psi}_i \rangle \langle \phi', \tilde{\psi}_j \rangle \\
&= \tilde{\varphi}^T G \tilde{\varphi}'.
\end{aligned}$$

At the fourth equality, we apply Fubini theorem to swap the two summations. To see that Fubini theorem can be applied at the fourth equality, we first note that $\sum_{k=1}^{\infty} u_{ki}^2$ is finite for each $i \in \{1 \dots n\}$ since

$$\sum_{k=1}^{\infty} u_{ki}^2 = \sum_{k=1}^{\infty} \langle \tilde{\psi}_i, \sum_{j=1}^n u_{kj} \tilde{\psi}_j \rangle \langle \tilde{\psi}_i, \sum_{j=1}^n u_{kj} \tilde{\psi}_j \rangle = \|A\tilde{\psi}_i\|^2 < \infty.$$

Applying the above result together with Cauchy-Schwarz inequality and Fubini theorem for non-negative summation, we have

$$\begin{aligned}
\sum_{k=1}^{\infty} \sum_{i,j=1}^n |u_{ki} u_{kj} \langle \phi, \tilde{\psi}_i \rangle \langle \phi', \tilde{\psi}_j \rangle| &= \sum_{i,j=1}^n \sum_{k=1}^{\infty} |u_{ki} u_{kj} \langle \phi, \tilde{\psi}_i \rangle \langle \phi', \tilde{\psi}_j \rangle| \\
&= \sum_{i,j=1}^n |\langle \phi, \tilde{\psi}_i \rangle \langle \phi', \tilde{\psi}_j \rangle| \left(\sum_{k=1}^{\infty} |u_{ki} u_{kj}| \right) \\
&\leq \sum_{i,j=1}^n |\langle \phi, \tilde{\psi}_i \rangle \langle \phi', \tilde{\psi}_j \rangle| \sqrt{\left(\sum_{k=1}^{\infty} u_{ki}^2 \right) \left(\sum_{k=1}^{\infty} u_{kj}^2 \right)} \\
&< \infty.
\end{aligned}$$

Hence, the summation converges absolutely and thus Fubini theorem can be applied as claimed above. Again, using the fact that $\sum_{k=1}^{\infty} u_{ki}^2 < \infty$, we have that each element of G , $G_{ij} = \sum_{k=1}^{\infty} u_{ki} u_{kj}$, is finite. Furthermore, the matrix G is PSD since each of its elements can be regarded as an inner product of two vectors in

ℓ_2 .

Hence, we finally have that $\langle A\phi_i, A\phi_j \rangle = \tilde{\varphi}_i^T G \tilde{\varphi}_j$, for each $1 \leq i, j \leq n$, and whenever a map A is given, we can construct G such that it results in the same objective function value. By reversing the proof, it is easy to see that the converse is also true. The first step of the proof is finished.

Step 2. We now prove the theorem without assuming that $\{\tilde{\psi}_i\}_{i=1}^n$ is an orthonormal set. Let all notations be the same as in Step 1. Let Ψ' be the matrix $(\tilde{\psi}_1, \dots, \tilde{\psi}_n)$. Define $\{\psi_i\}_{i=1}^n$ as an orthonormal set such that $\text{span}(\{\psi_i\}_{i=1}^n) = \text{span}(\{\tilde{\psi}_i\}_{i=1}^n)$ and $\Psi = (\psi_1, \dots, \psi_n)$ and $\varphi_i = \Psi^T \phi_i$. Then, we have that $\tilde{\psi}_i = \Psi \mathbf{c}_i$ for some $\mathbf{c}_i \in \mathbb{R}^n$ and $\Psi' = \Psi C$ where $C = (\mathbf{c}_1, \dots, \mathbf{c}_n)$. Moreover, since C map from an independent set $\{\psi_i\}$ to another independent set $\{\tilde{\psi}_i\}$, C is invertible. Denote $F = B^T B$ as an optimal matrix obtained by the proof of Step 1. We then have, for any F ,

$$\begin{aligned} \varphi_i^T F \varphi_j &= \varphi_i^T B^T B \varphi_j = \varphi_i^T C A^T A' C^T \varphi_j \\ &= \phi_i^T \Psi C A^T A' C^T \Psi^T \phi_j \\ &= \phi_i^T \Psi' A^T A' \Psi'^T \phi_j \\ &= \tilde{\varphi}_i^T A^T A' \tilde{\varphi}_j = \tilde{\varphi}_i^T G \tilde{\varphi}_j. \end{aligned}$$

Note that we can write $B^T = C A^T$ since C is invertible. Hence, for any F we have the matrix G which gives $\tilde{\varphi}_i^T G \tilde{\varphi}_j = \varphi_i^T F \varphi_j$. Using the same arguments as in Step 1, we finish the proof of Step 2 and of Theorem 1. \square

Theorem 6. (*Representer Theorem for Dimensionality Reduction*) For arbitrary objective function f which depends only on inner products of linearly transformed examples, the optimization,

$$\begin{aligned} \min_A f\left(\{\langle A\phi_i, A\phi_j \rangle\}_{i,j=1}^n\right) \\ \text{s.t. } A : \mathcal{H} \rightarrow \mathbb{R}^d \text{ is a bounded linear map,} \end{aligned}$$

has the same optimal value as

$$\min_{A' \in \mathbb{R}^{d \times (\ell+u)}} f(\{\varphi_i^T A'^T A' \varphi_j\}_{i,j=1}^n).$$

Proof. Let $\{e_i\}_{i=1}^d$ be the canonical basis of \mathbb{R}^d . By Lemma 1, $A\phi_k = \sum_{i=1}^d \langle \phi_k, \tau_i \rangle e_i$ for some $\tau_1, \dots, \tau_d \in \mathcal{X}$. Each τ_i can be decomposed as $\tau_i' + \tau_i^\perp$ such that τ_i' lies in $\text{span}\{\tilde{\psi}_1, \dots, \tilde{\psi}_n\}$ and τ_i^\perp is orthogonal to the span. These facts make $\langle \phi_k, \tau_i \rangle = \langle \phi_k, \tau_i' \rangle$ for every i . We then have, for some $u_{ij} \in \mathbb{R}$, $1 \leq i \leq d$, $1 \leq j \leq n$,

$$\begin{aligned} A\phi_k &= \sum_{i=1}^d \langle \phi_k, \sum_{j=1}^n u_{ij} \tilde{\psi}_j \rangle e_i = \sum_{i=1}^d e_i \sum_{j=1}^n u_{ij} \langle \phi_k, \tilde{\psi}_j \rangle \\ &= \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{d1} & \cdots & u_{dn} \end{bmatrix} \begin{bmatrix} \langle \phi_k, \tilde{\psi}_1 \rangle \\ \vdots \\ \langle \phi_k, \tilde{\psi}_n \rangle \end{bmatrix} = U\varphi_k. \end{aligned}$$

Now, one can easily check that $\langle A\phi_i, A\phi_j \rangle = \varphi_i^T U^T U \varphi_j$ for $i, j = 1, \dots, n$. Hence, whenever a map A is given, we can construct U such that it results in the same objective function value. By reversing the proof, the converse is also true, and thus the theorem is proven (by renaming U to A'). \square

The stronger version of Theorem 6 can be achieved by inserting a regularizer into the objective function of a (kernelized) Mahalanobis distance learner as stated in Theorem 7. For compact notations, we use the fact that A is representable by $\{\tau_i\}$ as shown in Lemma 1.

Theorem 7. (*Representer Theorem for Regularized Learners*) Define $\{\tilde{\psi}_i\}_{i=1}^n$ and f be as in Theorem 1. For monotonically increasing functions g_i , let

$$h(\tau_1, \dots, \tau_d, \phi_1, \dots, \phi_n) = f(\langle \tau_1, \phi_1 \rangle, \dots, \langle \tau_i, \phi_j \rangle, \dots, \langle \tau_n, \phi_d \rangle) + \sum_{i=1}^d g_i(\|\tau_i\|).$$

Any optimal set of linear functionals

$$\arg \min_{\{\tau_i\}} h(\tau_1, \dots, \tau_d, \phi_1, \dots, \phi_n)$$

s.t. $\forall i \tau_i : \mathcal{X} \rightarrow \mathbb{R}$ is a bounded linear functional

must admit the representation of $\tau_i = \sum_{j=1}^n u_{ij} \tilde{\psi}_j$ ($i = 1, \dots, d$).

Proof. We prove by contrapositive. Consider a set of linear functionals $\{\tau_i\}$ such that it is not in the span of ψ_j s, i.e. $\tau_i = \tau'_i + \tau_i^\perp$ where $\tau'_i = \sum_{j=1}^n u_{ij} \psi_j$ and $\langle \tau_i^\perp, \psi_j \rangle = 0$ for all $j = 1, \dots, \ell + u$ and $\exists_i \tau_i^\perp \neq 0$. Then $\langle \tau_i, \psi_j \rangle = \langle \tau'_i, \psi_j \rangle$. Thus,

$$f(\langle \phi_1, \tau_1 \rangle, \dots, \langle \phi_i, \tau_j \rangle, \dots, \langle \phi_{\ell+u}, \tau_d \rangle) = f(\langle \phi_1, \tau'_1 \rangle, \dots, \langle \phi_i, \tau'_j \rangle, \dots, \langle \phi_{\ell+u}, \tau'_d \rangle).$$

However,

$$g\left(\sum_{i=1}^d \|\tau_i\|\right) = g\left(\sum_{i=1}^d \sqrt{\|\tau'_i\|^2 + \|\tau_i^\perp\|^2}\right) > g\left(\sum_{i=1}^d \|\tau'_i\|\right),$$

using the fact that $\exists_i \tau_i^\perp \neq 0$. Hence,

$$h(\tau_1, \dots, \tau_d, \phi_1, \dots, \phi_n) > h(\tau'_1, \dots, \tau'_d, \phi_1, \dots, \phi_n),$$

and thus $\{\tau_i\}$ cannot be an optimal solution. The proof is completed. \square

To apply Theorem 7 to our framework, we can simply view $A = (\tau_1, \dots, \tau_d)^T$. If each g_i is the square function, then regularizer becomes $\sum_{i=1}^d \|\tau_i\|^2 = \|A\|_{HS}$ where $\|\cdot\|_{HS}$ is the Hilbert-Schmidt (HS) norm of an operator. If each τ_i is finite-dimensional, the HS norm is reduced to the Frobenius norm $\|\cdot\|_F$. Here, we allow the HS norm of a bounded linear operator to take a value of ∞ . For the kernel trick (by substituting $\tilde{\psi}_i = \phi_i$), the result above states that any optimal $\{\tau_i\}$ must be represented by $\{\Phi \mathbf{u}_i\}$. Therefore, we have

$$\sum_{i=1}^d g_i(\|\tau_i\|) = \sum_{i=1}^d \|\tau_i\|^2 = \sum_{i=1}^d \mathbf{u}_i^T \Phi^T \Phi \mathbf{u}_i = \sum_{i=1}^d \mathbf{u}_i^T K \mathbf{u}_i = \text{trace}(UKU^T).$$

This regularizer is first appeared in the work of Globerson and Roweis (2006). Similarly, for the KPCA trick (by substituting $\tilde{\psi}_i = \psi_i$), any optimal $\{\tau_i\}$ must be represented by $\{\Psi \mathbf{u}_i\}$ and, using the fact that $\Psi^T \Psi = I$, we have $\sum_{i=1}^d \|\tau_i\|^2 = \text{trace}(UU^T) = \|U\|_F^2$.

By adding the regularizer, $\text{trace}(UKU^T)$ or $\|U\|_F^2$, into existing objective

functions, we have a new class of learners, namely, *regularized Mahalanobis distance learners* such as regularized KNCA (RKNCA), regularized KLMNN (RKLMMN) and regularized KDNE (RKDNE). We plan to investigate effects of using various types of regularizers in the near future.

Remarks. The three theorems require that the objective function of a learning algorithm must depend only on $\{\langle A\phi_i, A\phi_j \rangle\}_{i,j=1}^n$ or equivalently $\{\langle \phi_i, M\phi_j \rangle\}_{i,j=1}^n$. This actually is not a strict condition since metric learners in literatures have their objective functions in this form (Chen et al., 2005; Goldberger et al., 2005; Globerson and Roweis, 2006; Weinberger et al., 2006; Yang et al., 2006a; Sugiyama, 2006; Yan et al., 2007; Zhang et al., 2007b; Torresani and Lee, 2007).

We note that the three theorems are more general than what is needed by the kernel trick. To apply the theorems to the usual kernel trick, we just substitute $\tilde{\psi}_i = \phi_i$ so that $\tilde{\varphi}_i$ becomes:

$$\tilde{\varphi}_i = \Phi^T \phi_i = (\langle \phi_1, \phi_i \rangle, \dots, \langle \phi_n, \phi_i \rangle)^T = (k(\mathbf{x}_1, \mathbf{x}_i), \dots, k(\mathbf{x}_n, \mathbf{x}_i))^T \equiv \mathbf{k}_i.$$

The notation \mathbf{k}_i is common in literatures and the matrix $(\mathbf{k}_1, \dots, \mathbf{k}_n)$ is the so-called *Gram matrix*. Now we have a computational shortcut for $\|\phi_i - \phi_j\|_M$:

$$\|\phi_i - \phi_j\|_M = \sqrt{(\phi_i - \phi_j)^T M (\phi_i - \phi_j)} = \sqrt{(\mathbf{k}_i - \mathbf{k}_j)^T G (\mathbf{k}_i - \mathbf{k}_j)}.$$

Finally, we note that applications of the three theorems are not limited to the kernel trick. When restricting $\{\tilde{\psi}_i\}$ to be an orthonormal set, Theorem 5 inspires the KCPA trick framework which has many advantages over the kernel trick framework on the tasks of metric learning as described in Chapter 3.

CHAPTER VIII

FUTURE DIRECTIONS

In the previous chapters, we present a general framework for kernelization and a spectral-based semi-supervised learning framework. These two frameworks can be applied together as shown in Chapter 5. Many extensions of the frameworks are already stated in each chapter. Here, we summarize them again before stating other possible directions.

- Kernelized learners can involve ill-conditioned computation as stated in Chapter 3. It is important to develop a numerically stable techniques in order to stabilize the kernelized learners so that the results gotten from the learners will make sense.

- The kernel alignment method presented in Chapter 4 cannot deals with a dataset where some class forming a shape of multi-modality. This limitation severely affects its usefulness to general real-world datasets. Hence, it is highly important to develop a new efficient kernel selection algorithm which can deals with multi-modality. Note that some kernel selection methods can deal with multi-modality, but they are not computationally efficient, e.g. those employing evolutionary algorithms. The survey of efficient kernel selection algorithms can be found in Goenen and AlpaydIn (2010).

- There are some supervised manifold learners which cannot be represented in our framework (Goldberger et al., 2005; Globerson and Roweis, 2006; Weinberger et al., 2006; Yang et al., 2006a; Torresani and Lee, 2007; Tao et al., 2009) because cost functions of these algorithms are not linear with respect to distances among examples. Extension of these algorithms to handle semi-supervised learning problems is an interesting future work as stated in Chapter 5.

- In Chapter 7, the class of regularized Mahalanobis distance learners are introduced. It is interesting to see practical prediction abilities of these learners when compared to non-regularized learners mainly concerned in this dissertation. Also, we note that regularization usually improves a condition number of a problem

so that it can also help to stabilize kernelized learners as mentioned above.

8.1 Learning a Mahalanobis Distance for SVM

Learning SVM is also based on Euclidean distance, and thus the use of the Euclidean distance may not be appropriate for some learning problems. Being able to learn the best (Mahalanobis) distance function with respect to a given training set can improve the performance of a learner in that problem.

In fact, for SVM, there is another explanation. Recall that the goal of SVM is to find an optimal-margin hyperplane. Let $\{\mathbf{x}_i, y_i\}_{i=1}^n$ be a given dataset. Note that a representation of $\{\mathbf{x}_i\} \subset \mathbb{R}^D$ can affect the margin of a hyperplane h . For example, any change of scales of some attributes of $\{\mathbf{x}_i\}$ will change a margin of h . More precisely, let A be a scaling matrix, the optimal-margin hyperplane with respect to $\{\mathbf{x}_i\}$ is not likely to be the same as the optimal-margin hyperplane with respect to $\{A\mathbf{x}_i\}$. There are a number of situations such that a dataset is recorded with some inappropriate scales because an observer does not have enough prior knowledge, for example, it is well known that the popular dataset named WINE of the UCI repository (Asuncion and Newman, 2007) is recorded with some inappropriate scales. More generally, besides a scaling matrix, A could be any types of matrices which change the representation of $\{\mathbf{x}_i\}$ to $\{A\mathbf{x}_i\}$, i.e. a dataset may be recorded as $\{A\mathbf{x}_i\}$ instead of its most appropriate form because of some misunderstanding or some errors in observation and recordation.

Therefore, given a dataset, it is beneficial to learn the best linear transformation A^* such that the margin of the optimal-margin hyperplane with respect to $\{A^*\mathbf{x}_i\}$ is largest when compared to other representations in the form of $\{A\mathbf{x}_i\}$ for all $A \in \mathbb{R}^{D \times D}$. It turns out that the problem of learning the best possible matrix A can also be formulated as the problem of learning the best possible Mahalanobis distance matrix $M \in \mathbb{S}_+^D$.

In order to learn the best linear transformation A , together with the optimal-margin hyperplane, a new formulation of SVM is required. One possible formulation is shown in Fig. 8.1:

$$\begin{array}{l}
\text{Minimize } C\|\mathbf{w}\|^2 + \sum_{i=1}^n \xi_i \\
\text{Subject to: } y_i(\langle \mathbf{w}, A\mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \\
\|A\mathbf{x}_i\|^2 \leq 1 \\
\xi_i \geq 0 \\
\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}, A \in \mathbb{R}^{D \times D}.
\end{array}$$

Figure 8.1: Our first formulation for SVM.

Using the property $M = A^T A$, we can reformulate the problem in another way as shown in Fig. 8.2.

$$\begin{array}{l}
\text{Minimize } C\mathbf{w}^T M \mathbf{w} + \sum_{i=1}^n \xi_i \\
\text{Subject to: } y_i(\langle \mathbf{w}, M\mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \\
\mathbf{x}_i^T M \mathbf{x}_i \leq 1 \\
\xi_i \geq 0 \\
\mathbf{w} \in \mathbb{R}^D, b \in \mathbb{R}, M \in \mathbb{S}_+^D.
\end{array}$$

Figure 8.2: Our second formulation for SVM.

Nevertheless, to our knowledge, it appears that the problem formulations shown in Fig. 8.1 and Fig. 8.2 do not belong to the class of convex programming. Hence, with these formulations the problem cannot be perfectly solved in the sense that a globally optimal solution is not guaranteed to be found in the polynomial running-time.

8.2 Other Directions

- In this dissertation, we consider only problems related to classification, clustering and the combination of both, i.e. semi-supervised learning. From the best of the author's knowledge, a Mahalanobis distance learner for tasks related to regression analysis have not been much concerned. Therefore, it is one interesting direction to develop a Mahalanobis distance learner to improve the performance of an existing regressor such as least square, in the same sense as a distance learner considered in this dissertation which improves the performance of the kNN algorithm.

- Up to now, the main contents of this dissertation is of theory and of modeling. What this dissertation lack of is about its applications to challenging real world problems, not just, say, a UCI dataset. It is highly interesting to apply our

methods to the high-dimensional real-world datasets such as those related to text categorization, speech recognition, computer vision and financial data prediction. Nevertheless, working on these real-world datasets is not straightforward. Some certain issues such as time series analysis, spatial analysis, sparsity, sensitivity and signal pre-processing are needed to be carefully concerned. For example, in a task of financial data prediction, the philosophy of the complexity-accuracy trade-off may be different from the other prediction tasks as accuracy in the financial prediction is very crucial, e.g. 0.1% of error can be unacceptable.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

References

- Asuncion A. and Newman D.J. UCI Machine Learning Repository [Online]. 2007.
Available from : www.ics.uci.edu/~mllearn/.
- Bach F.R. and Jordan M.I. Learning spectral clustering, with application to speech separation. Journal of Machine Learning Research (2006): 1963–2001.
- Belhumeur P.N., Hespanha J.P. and Kriegman D.J. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. Pattern Analysis and Machine Intelligence (1997): 711–720.
- Belkin M. and Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation (2003): 1373–1396.
- Bishop C.M. Pattern Recognition and Machine Learning. Heidelberg, Germany: Springer, 2006.
- Boyd S. and Vandenberghe L. Convex Optimization. New York, NY: Cambridge University Press, 2004.
- Breiman L. Arcing classifiers. Annals of Statistics (1998): 801–823.
- Cai D., He X. and Han J. Spectral regression for efficient regularized subspace learning. pp. 281–288, In International Conference on Computer Vision. Rio de Janeiro, Brazil: IEEE Press. 2007.
- Chapelle O. and Schölkopf B. Incorporating Invariances in Nonlinear SVMs. pp. 1–8, In Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press. 2001.
- Chapelle O., Schölkopf B. and Zien A., editors. Semi-Supervised Learning (Adaptive Computation and Machine Learning). Cambridge, MA: MIT Press, 2006.
- Chatpatanasiri R. and Kijisirikul B. A Unified Semi-Supervised Dimensionality Reduction Framework for Manifold Learning. Neurocomputing: Special Issue on Subspace Learning (2010): 11–20.
- Chatpatanasiri R., Korsrilabutr T., Tangchanachaianan P. and Kijisirikul B. A New Kernelization Framework for Mahalanobis Distance Learning Algorithms. Neurocomputing: Special Issue on Subspace Learning (2010): 1–10.

- Chatpatanasiri R., Pungprasertying P. and Kijirikul B. Zone analysis: a visualization framework for classification problems. Artificial Intelligence Review (2009): 1–18.
- Chen H.T., Chang H.W. and Liu T.L. Local discriminant embedding and its variants. pp. 846 – 853, In Computer Vision and Pattern Recognition, volume 2. Washington, DC: IEEE Press. 2005.
- Cheng J., Liu Q., Lu H. and Chen Y.W. A supervised nonlinear local embedding for face recognition. pp. 83 – 86, In International Conference on Image Processing, volume 1. Singapore: IEEE Press. 2004.
- Demmel J.W. Applied numerical linear algebra. Philadelphia, PA: SIAM, 1997.
- Friedman J.H. Regularized Discriminant Analysis. Journal of the American Statistical Association (1989): 165–175.
- Fukunaga K. Introduction to Statistical Pattern Recognition. Maryland, MO: Academic Press, 1990.
- Georghiades A.S., Belhumeur P.N. and Kriegman D.J. From few to many: Illumination cone models for face recognition under variable lighting and pose. Pattern Analysis and Machine Intelligence (2001): 643–660.
- Globerson A. and Roweis S. Metric learning by collapsing classes. pp. 1–8, In Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press. 2006.
- Goenen M. and Alpaydm E. Multiple kernel learning algorithms. Technical Report (2010): Submitted for Publication.
- Goldberger J., Roweis S., Hinton G. and Salakhutdinov R. Neighbourhood components analysis. pp. 1–8, In Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, Cambridge, MA. 2005.
- Guermeur Y., Lifchitz A. and Vert R. A kernel for protein secondary structure prediction. Kernel Methods in Computational Biology (2004): 193–206.
- Hastie T., Tibshirani R. and Friedman J.H. The Elements of Statistical Learning. Heidelberg, Germany: Springer, 2001.
- He X. and Niyogi P. Locality Preserving Projections. pp. 1–8, In Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press. 2004.

- Hoi S.C.H., Liu W., Lyu M.R. and Ma W.Y. Learning distance metrics with contextual constraints for image retrieval. pp. 2072–2078, In Computer Vision and Pattern Recognition. New York, NY: IEEE Press. 2006.
- Jain P., Kulis B., Davis J.V. and Dhillon I.S. Metric and Kernel Learning using a Linear Transformation. Arxiv preprint: arXiv:0910.5932 (2009): Submit for Publication.
- James G.M. Variance and bias for general loss functions. Machine Learning (2003): 115–135.
- Kimeldorf G.S. and Wahba G. Some Results on Tchebycheffian Spline Functions. Journal of Mathematical Analysis and Applications (1971): 82–95.
- Lanckriet G.R.G., Cristianini N., Bartlett P., Ghaoui L.E. and Jordan M.I. Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research (2004): 27–72.
- Lewkeeratiyutkul W. Lecture Notes on Real Analysis I-II. [Online]. 2006. Available from : <http://pioneer.netserv.chula.ac.th/~lwicharn/2301622/>.
- Li F., Yang J. and Wang J. A transductive framework of distance metric learning by spectral dimensionality reduction. pp. 513–520, In International Conference on Machine learning. New York, NY: ACM. 2007.
- Li H., Jiang T. and Zhang K. Efficient and Robust Feature Extraction by Maximum Margin Criterion. IEEE Transactions on Neural Networks (2006): 157–165.
- Li J., Li X. and Tao D. KPCA for semantic object extraction in images. Pattern Recognition (2008a): 3244–3250.
- Li X., Lin S., Yan S. and Xu D. Discriminant locally linear embedding with high-order tensor data. Systems, Man and Cybernetics-Part B (2008b): 342–352.
- Loefberg J. Yalmip : A toolbox for modeling and optimization in MATLAB. pp. 1–8, In Computer Aided Control System Design. Taipei, Taiwan: IEEE Press . 2004.
- Minka T.P. Old and new matrix algebra useful for statistics. [Online] . 1997. Available from : www.stat.cmu.edu/minka/papers/matrix.html.

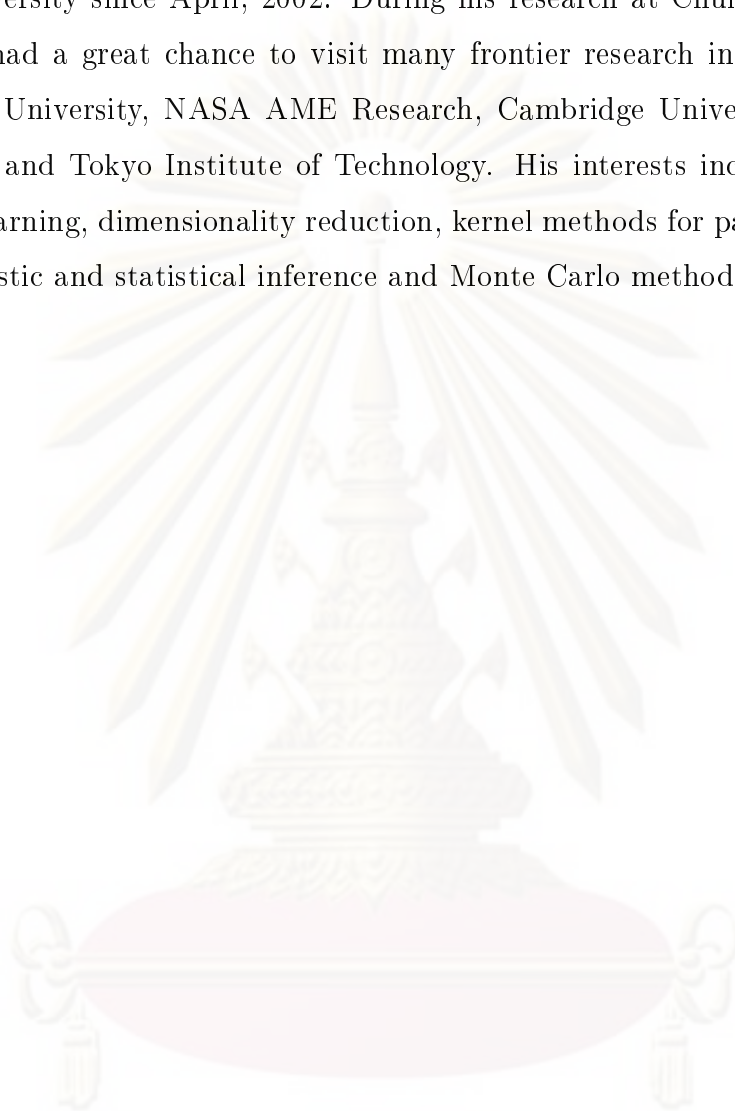
- Ng A., Zheng A.X. and Jordan M.I. Link analysis, eigenvectors and stability. pp. 903–910, In International Joint Conference on Artificial Intelligence. Washington, DC: Morgan Kaufmann. 2001.
- Pan V.Y., Qian G. and Zheng A.L. Preconditioning, Randomization, Solving Linear Systems, Eigen-Solving, and Root-Finding. pp. 5–6, In The 2009 conference on Symbolic Numeric Computation. Kyoto, Japan: ACM. 2009.
- Pang Y., Liu Z. and Yu N. A new nonlinear feature extraction method for face recognition. Neurocomputing (2006): 949–953.
- Pang Y., Wang L. and Yuan Y. Generalized KPCA by Adaptive Rules in Feature Space. International Journal of Computer Mathematics (2009): 1–8.
- Pang Y., Yuan Y. and Li X. Effective feature extraction in high-dimensional space. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics (2008): 1652–1656.
- Roweis S.T. and Saul L.K. Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science (2000): 2323–2326.
- Saul L.K., Weinberger K.Q., Ham J.H., Sha F. and Lee D.D. Spectral methods for dimensionality reduction. pp. 293–308, In Semisupervised Learning. Cambridge, MA: MIT Press. 2006.
- Schölkopf B., Herbrich R. and Smola A.J. A generalized representer theorem. pp. 416–426, In Conference on Learning Theory. Heidelberg, Germany: Springer. 2001.
- Schölkopf B. and Smola A.J. Learning with Kernels. Cambridge, MA: MIT Press, 2001.
- Schott J.R. Matrix Analysis for Statistics. Hoboken, NJ: Wiley, 2nd edition, 2005.
- Shawe-Taylor J. and Cristianini N. Kernel Methods for Pattern Analysis. Cambridge, UK: Cambridge University Press, 2004.
- Song Y., Nie F., Zhang C. and Xiang S. A Unified framework for semi-supervised dimensionality reduction. Pattern Recognition (2008): 2789–2799.
- Sugiyama M. Local fisher discriminant analysis for supervised dimensionality reduction. pp. 905–912, In International Conference on Machine Learning. New York, NY: ACM. 2006.

- Sugiyama M. Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis. The Journal of Machine Learning Research (2007): 1027–1061.
- Sugiyama M., Ide T., Nakajima S. and Sese J. Semi-Supervised Local Fisher Discriminant Analysis for Dimensionality Reduction. Pacific and Asia Conference on Knowledge Discovery and Data Mining (2008): 333–344.
- Tao D., Li X., Wu X. and Maybank S. Geometric mean for subspace selection. Pattern Analysis and Machine Intelligence (2009): 260–274.
- Tenenbaum J.B., de Silva V. and Langford J.C. A global geometric framework for nonlinear dimensionality reduction. Science (2000): 2319–2323.
- Torresani L. and Lee K. Large margin components analysis. pp. 1–8, In Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, Cambridge, MA. 2007.
- Vapnik V.N. The Nature of Statistical Learning Theory (Information Science and Statistics). Heidelberg, Germany: Springer, 1999.
- von Luxburg U. A tutorial on spectral clustering. Statistics and Computing (2007): 395–416.
- Weinberger K., Blitzer J. and Saul L. Distance metric learning for large margin nearest neighbor classification. pp. 1–8, In Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press. 2006.
- Xing E.P., Ng A.Y., Jordan M.I. and Russell S. Distance metric learning with application to clustering with side-information. pp. 1–8, In Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press. 2003.
- Xu L., White M. and Schuurmans D. Optimal Reverse Prediction. pp. 1137–1144, In International Conference on Machine Learning. New York, NY: ACM. 2009.
- Yan S., Xu D., Zhang B., Zhang H.J., Yang Q. and Lin S. Graph embedding and extensions: A general framework for dimensionality reduction. Pattern Analysis and Machine Intelligence (2007): 40–51.
- Yang J. and Yang J.Y. Why can LDA be performed in PCA transformed space? Pattern Recognition (2003): 563–566.

- Yang L., Jin R., Sukthankar R. and Liu Y. An efficient algorithm for local distance metric learning. In The Twenty-first National Conference on Artificial Intelligence. Boston, Massachusetts: AAAI Press. 2006a.
- Yang X., Fu H., Zha H. and Barlow J.L. Semi-supervised nonlinear dimensionality reduction. pp. 1065–1072, In International Conference on Machine Learning. New York, NY: ACM. 2006b.
- Yu H. and Yang J. A direct LDA algorithm for high-dimensional data - with application fo face recognition. Pattern Recognition (2001): 2067–2070.
- Zelnik-Manor L. and Perona P. Self-tuning spectral clustering. pp. 1601–1608, In Advances in Neural Information Processing Systems, volume 17. Cambridge, MA: MIT Press. 2004.
- Zhang C., Nie F. and Xiang S. A general kernelization framework for learning algorithms based on kernel PCA. Neurocomputing (2009a): To be published.
- Zhang D., Zhou Z.H. and Chen S. Semi-supervised dimensionality reduction. pp. 11–393, In the 7th SIAM International Conference on Data Mining. Philadelphia, PA: SIAM. 2007a.
- Zhang T., Li X., Tao D. and Yang J. Local Coordinates Alignment (LCA): A Novel Manifold Learning Approach. International Journal on Pattern Recognition and Artificial Intelligence (2008): 667–690.
- Zhang T., Li X., Tao D. and Yang J. Patch Alignment for Dimensionality Reduction. IEEE Transaction on Knowledge Data Engineering (2009b): 1299–1313.
- Zhang W., Xue X., Sun Z., Guo Y.F. and Lu H. Optimal dimensionality of metric space for classification. pp. 1135–1142, In International Conference on Machine Learning. Oregon: ACM. 2007b.
- Zhang Z. Learning Metrics via Discriminant Kernels and Multidimensional Scaling: Toward Expected Euclidean Representation. pp. 872–879, In International Conference on Machine Learning. New York, NY: ACM. 2003.
- Zhu X., Kandola J., Ghahramani Z. and Lafferty J. Nonparametric transforms of graph kernels for semi-supervised learning. pp. 1–8, In Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press. 2005.

Biography

Ratthachat Chatpatanasiri was born in Nakhon Ratchasima, Thailand, on December 13, 1981. He received Bachelor Degree of Engineering from Kasetsart University since April, 2002. During his research at Chulalongkorn University, he had a great chance to visit many frontier research institutes, including Freiburg University, NASA AME Research, Cambridge University, Max-Planck Institute and Tokyo Institute of Technology. His interests include Mahalanobis metric learning, dimensionality reduction, kernel methods for pattern recognition, probabilistic and statistical inference and Monte Carlo methods.



ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย